

Contents

Welcome to the Developer and IT-Pro Help for Dynamics 365 Business Central

Get started

Frequently asked questions

Features not implemented in on-premises deployments

Help and Support

Resources for Help and Support

Configuring technical support

Help system

User assistance model

Extend, Customize, and Collaborate on the Help

Configure Context-Sensitive Help

Deployment

Deployment Overview

Configuring the Help Experience

Online

Choosing Your Development Sandbox Environment

Embed App

Embed App Overview

Microsoft Responsibilities

Qualification and Onboarding

Managing in Microsoft Lifecycle Services

Components

Platform

Licensing

Customer Sign-up

AppSource

Sandbox

Ecosystem Features

On-Premises

System requirements

Software lifecycle policy and on-premises releases

Running a Container-Based Development Environment

Components

Planning

Deployment Topologies

Deployment Topologies Overview

Deploying Demonstration Environment

Deploying Single-Computer

Deploying on Two-Computers

Deploying on Three Computers

Installing Using Setup

Provisioning a Service Account

Securing Remote Connections Using Certificates

Business Central Web Server

Business Central Web Server Overview

Configuring Web Server Instances

Configure IIS

Configure SSL

Setting Up Multiple Web Server Instances

Migrating to Multitenancy

Database

Installation Considerations for SQL Server

Configuring Database Authentication

Creating and Altering Databases

Deploying to Azure SQL Database

Administration

Online

Administration of Business Central Online

Administration Center

Administration Center Overview

Managing Environments

Tenant Notifications

Environment Telemetry

Administration Center API

Automation

Introduction to Automation APIs

Automation API Overview

On-Premises

Intelligent Insights

Connect to the Intelligent Cloud from On-Premises

Replicating On-Premises Data

Managing your Intelligent Cloud Environment

Frequently Asked Questions about Connecting to the Intelligent Cloud

Server Administration Tool

Windows PowerShell Cmdlets

Windows PowerShell Cmdlets for Business Central

Administration Cmdlets

Administration Cmdlets for Extensions

Development Cmdlets

Development Cmdlets for Extensions

Authentication and Credential Types

Configuring Business Central Server

Configuring Business Central Web Server

Configuring Business Central Web Server Instances

Setting Up Multiple Web Server Instances

Configuring Database Authentication

Monitoring Business Central Server

Monitoring Performance Counters

Monitoring Server Events

Monitoring Server Events Overview

Trace Events List

Admin and Operational Events List

Using Event Viewer

Using Performance Monitor

Using PerView

Using LogMan

Using PowerShell

Turn Off or Limit Telemetry

Monitoring Long Running SQL Queries

SQL Server Performance

Optimizing SQL Server Performance

Compatibility Level

Data Access

Table Keys and Performance

Bulk Inserts

AL Database Methods

Query Objects

Troubleshooting: Analyzing Long Running SQL Queries Involving FlowFields

Troubleshooting: Using the Event Log to Monitor Long Running SQL Queries

Understanding Session Timeouts

Preparing Dynamics 365 for Sales for Integration

Development

Development in AL

Getting Started

Getting Started with AL

Choosing Your Development Sandbox Environment

Building Your First Sample Extension With Extension Objects, Install Code, and Upgrade Code

Using Designer

Keyboard Shortcuts

AL Formatter

AL Outline View

AL Code Navigation

AL Code Actions

Object Ranges

Adding Help Links from Pages, Reports, and XMLports

Working with Translation Files

Ready to Go

Getting Onboarded through Ready to Go

The "Ready to Go" Online Learning Catalog

Add-On Apps - Getting You Started

AppSource Validation

Marketing Validation Checklist

Technical Validation Checklist

How to Make Compelling Videos

How to Create an Effective Sales Landing Page

Getting Started with AL for On-Premises

Getting Started with C/SIDE and AL Side-by-Side for On-Premises

Running C/SIDE and AL Side-by-Side

Creating Runtime Packages for Business Central On-Premises

Manifest Files

JSON Files

Security Setting and IP Protection

Developing for Multiple Platform Versions

Debugging

Debugging in AL

RAD publishing in AL

Converting, Upgrading, and Installing Extensions

The Lifecycle of Apps and Extensions for Business Central

Converting from Extensions V1 to Extensions V2

The Txt2Al Conversion Tool

Generating Delta Files

Exporting Data for Extensions

Writing Extension Install Code

Upgrading Extensions

Publish and Install an Extension V2

Upgrading AppSource Apps in Production

Signing an App Package File

Deploying a Tenant Customization

Extending the Base Application

Extending Application Areas

Extending Item Charge Distribution Methods

Events

Events in AL

Event Types

Publishing Events

Raising Events

Subscribing to Events

Discover Events Using the Event Recorder

Notifications

Task Scheduler

Tables

Tables Overview

Table Object

Table Extension Object

Setting Relationships Between Tables

View Table Data

Insert, Modify, ModifyAll, Delete, and DeleteAll Methods

Get, Find, and Next Methods

Retaining Table Data after Publishing

Classifying Data

Enabling Sales Tables for Extension Development

Pages

Pages Overview

Page Object

Page Extension Object

Page Customization Object

API Page Type

Role Centers

Designing Role Centers

Sample Role Center

Adding links to the Navigation menu

Headlines

Cues and Action Tiles

Designing Pages

List Pages

Designing List Pages

Sample List Page

Displaying Data as Tiles

Views

Adding Filter Tokens

Card Pages

Designing Card Pages

Sample Card Page

FactBoxes

Adding Pages to Tell Me

Fields

Arranging Fields on a FastTab

Grid Control

Fixed Control

Field Groups

CalcFields, CalcSums, FieldError, FieldName, Init, TestField, and Validate Methods

Actions

Actions in AL

Adding Actions to a Page

Promoted Actions

Inspecting and Troubleshooting Pages

Control Add-in Style Guide

Reports

Reports Overview

Report Design Overview

Report Object

[Defining a Report Dataset](#)

[Request Pages](#)

[Adding Reports to Tell Me](#)

[Testing a Report](#)

[How to: Create a Word Layout Report](#)

[How to: Create an RDL Layout Report](#)

[Linking to the Web Client and App](#)

[Web Client URL](#)

[Business Central App URL](#)

[Working with Translation Files](#)

[Developing Connect Apps](#)

[Instrumenting for Telemetry](#)

[.NET in AL](#)

[Getting started with Microsoft .NET Interoperability from AL](#)

[.NET Control Add-Ins](#)

[Subscribing to Events in a .NET Framework Type](#)

[Serializing .NET Framework Types](#)

[Exporting Permission Sets](#)

[Creating and Interacting with an OData V4 Bound Action](#)

[AL Programming](#)

[AL Development Environment](#)

[AL Programming Guide](#)

[AL Simple Statements](#)

[FAQ for Developing in AL](#)

[Working with Multiple AL Project Folders within One Workspace](#)

[Code Analysis](#)

[Using the Code Analysis Tool](#)

[Ruleset for the Code Analysis Tool](#)

[Using the Code Analysis Tools with the Ruleset](#)

[AppSourceCop Analyzer Rules](#)

[CodeCop Analyzer Rules](#)

[PerTenantExtensionCop Analyzer Rules](#)

UICop Analyzer Rules

Isolated Storage

File Handling and Text Encoding

Flowfields

FlowFields and FlowFilters

Extensible Enums

Objects

Table Object

Table Extension Object

Table Keys

Page Object

Page Extension Object

Page Customization Object

Report Object

Profile Object

Codeunit Object

Query Object

XMLPort Object

Control Add-In Object

Methods

Methods Overview

Array Methods

Method Attributes

Procedure Overload

Action Option Type

Any Data Type

BigInteger Data Type

BigText Data Type

Blob Data Type

Boolean Data Type

Byte Data Type

Char Data Type

ClientType Option Type
Code Data Type
Codeunit Data Type
CodeunitInstance Data Type
CompanyProperty Data Type
Database Data Type
DataClassification Option Type
DataScope Option Type
Date Data Type
DateFormula Data Type
DateTime Data Type
Debugger Data Type
Decimal Data Type
DefaultLayout Option Type
Dialog Data Type
Dictionary Data Type
DotNet Data Type
Duration Data Type
ExecutionContext Option Type
ExecutionMode Option Type
FieldClass Option Type
FieldRef Data Type
FieldType Option Type
File Data Type
FilterPageBuilder Data Type
Guid Data Type
HttpClient Data Type
HttpContent Data Type
HttpHeaders Data Type
HttpRequestMessage Data Type
HttpResponseMessage Data Type
InStream Data Type

Integer Data Type
IsolatedStorage Data Type
JsonArray Data Type
JsonObject Data Type
JsonToken Data Type
JsonValue Data Type
KeyRef Data Type
Label Data Type
List Data Type
Media Data Type
MediaSet Data Type
ModuleDependencyInfo Data Type
ModuleInfo Data Type
NavApp Data Type
None Data Type
Notification Data Type
NotificationScope Option Type
ObjectType Option Type
Option Data Type
OutStream Data Type
Page Data Type
ProductName Data Type
Query Data Type
Record Data Type
RecordID Data Type
RecordRef Data Type
Report Data Type
ReportFormat Option Type
RequestPage Data Type
SecurityFilter Option Type
Session Data Type
SessionSettings Data Type

String Data Type

System Data Type

TableConnectionType Option Type

TaskScheduler Data Type

TestAction Data Type

TestField Data Type

TestFilter Data Type

TestFilterField Data Type

TestPage Data Type

TestPart Data Type

TestPermissions Option Type

TestRequestPage Data Type

Text Data Type

TextBuilder Data Type

TextConst Data Type

TextEncoding Option Type

Time Data Type

TransactionModel Option Type

TransactionType Option Type

Variant Data Type

Verbosity Option Type

Version Data Type

WebServiceActionContext Data Type

WebSeviceActionResultCode Option Type

XmlAttribute Data Type

XmlAttributeCollection Data Type

XmlCDATA Data Type

XmlComment Data Type

XmlDeclaration Data Type

XmlDocument Data Type

XmlDocumentType Data Type

XmlElement Data Type

[XmlNamespaceManager Data Type](#)

[XmlNameTable Data Type](#)

[XmlNode Data Type](#)

[XmlNodeList Data Type](#)

[XmlPort Data Type](#)

[XmlProcessingInstruction Data Type](#)

[XmlReadOptions Data Type](#)

[XmlText Data Type](#)

[XmlWriteOptions Data Type](#)

[Properties](#)

[Properties Overview](#)

[Table and Table Extension Properties](#)

[Page and Page Extension Properties](#)

[Codeunit Properties](#)

[Query Properties](#)

[Report Properties](#)

[XMLPort Properties](#)

[Control Add-In Properties](#)

[View Properties](#)

[Integrating with Dynamics 365 for Sales](#)

[Triggers](#)

[Triggers Overview](#)

[Table and Field Triggers](#)

[Page and Action Triggers](#)

[Codeunit Triggers](#)

[Report and Data Item Triggers](#)

[XMLPort Triggers](#)

[Query Triggers](#)

[Rules and Guidelines](#)

[Rules and Guidelines for AL Code](#)

[Best Practices for AL](#)

[Benefits and Guidelines for using a Prefix or Suffix](#)

Testing your Extension

User Scenario Documentation

Restrictions on UI for Objects Exposed as Web Services

Replacing OnBeforeCompanyOpen and OnAfterCompanyOpen

Building an Advanced Sample Extension

Testing the Advanced Sample Extension

Web Services

General

Publishing a Web Service

Handling UI Interaction

Managing Timezones

Working with Static Proxy

Authentication

Securing Remote Connections Using Certificates

Best Practices

SOAP

SOAP Service URIs

Basic Operations

Create

CreateMultiple

Delete

Delete_<part>

GetRecIdFromKey

IsUpdated

Read

ReadByRecId

ReadMultiple

Update

UpdateMultiple

Retrieving Companies

Indicating That a Value Exists in Field

OData

[Return or Obtain an AtomPub Document](#)

[Return or Obtain Service Metadata EDMX Document](#)

[Return or Obtain a JSON Document](#)

[Using Filter Expressions in OData URIs](#)

[Using FlowFilters in OData URIs](#)

[Server-Driven Paging](#)

[Containments and Associations](#)

[Using OData on Queries Set with Top Number of Rows](#)

[Using OData to Modify Data](#)

[Walkthrough: Creating and Interacting With an OData V4 Bound Action](#)

[Security](#)

[Security and Protection Overview](#)

[Application](#)

[Online](#)

[On-Premises](#)

[Upgrade](#)

[Upgrading to Business Central](#)

[Online](#)

[Importing Business Data from Other Finance Systems](#)

[The Dynamics GP Data Migration Extension](#)

[The QuickBooks Data Migration Extension](#)

[On-Premises](#)

[Transitioning From Codeunit 1](#)

[Technical Upgrade](#)

[Quick Reference](#)

[Upgrading the Application Code](#)

[Upgrading the Data: Single-Tenant Mode](#)

[Quick Reference](#)

[Upgrading the Data: Multitenant Mode](#)

[Quick Reference](#)

[Before You Upgrade](#)

[Important Information and Considerations for Before Upgrading](#)

[Deprecated Fields, and Fields Marked as Obsolete](#)

[Deprecated Features in the Austrian Version](#)

[Deprecated Features in the Belgian Version](#)

[Deprecated Features in the Canadian Version](#)

[Deprecated Features in the Dutch Version](#)

[Deprecated Features in the Finnish Version](#)

[Deprecated Features in the German Version](#)

[Deprecated Features in the Icelandic Version](#)

[Deprecated Features in the Italian Version](#)

[Deprecated Features in the Mexican Version](#)

[Deprecated Features in the Norwegian Version](#)

[Deprecated Features in the Swedish Version](#)

[Deprecated Features in the Swiss Version](#)

[Deprecated Features in the UK Version](#)

[Deprecated Features in the United States Version](#)

[Migrate Legacy Help to the Business Central Format](#)

[Dynamics 365 Business Central API](#)

Welcome to the Developer and IT-Pro Help for Dynamics 365 Business Central

3/31/2019 • 2 minutes to read

Dynamics 365 Business Central is a complete enterprise resource planning (ERP) software solution for mid-sized organizations that is fast to implement, easy to configure, and simple to use. Right from the start, simplicity has guided — and continues to guide — innovations in product design, development, implementation, and usability. In this section, you can find information about deployment and administration, and you can find information about developing for Dynamics 365 Business Central using the AL Language extension and Visual Studio Code.

TIP

If you are looking for the C/SIDE documentation, visit our [Dynamics NAV library](#).

See Also

[Welcome to Dynamics 365 Business Central](#)
[Dynamics 365 Business Central blog for partners](#)
[Dynamics NAV developer and ITpro content](#)

Frequently Asked Questions for Dynamics 365 Business Central Developer and ITPro Experiences

4/10/2019 • 2 minutes to read

This section contains answers to frequently asked questions about developing for and administering Dynamics 365 Business Central.

TIP

If you are looking for frequently asked questions about signing up for and using Business Central, see [Frequently Asked Questions](#) in the business functionality content for Business Central.

Is Business Central available in my country?

Business Central is available in a limited number of markets, but new countries are added through Microsoft-led localization or through partner-led localization on a quarterly basis. For more information, see [Countries and Translations Supported](#).

How often is Business Central updated?

Business Central online is governed by [Microsoft's Modern Lifecycle Policy](#). This means continuous service updates and a major update every 6 months. Stay tuned for more information.

For information about lifecycle support for Business Central on-premises, see [Software Lifecycle Policy and Dynamics 365 Business Central On-Premises Updates](#).

How often are production databases backed up?

Databases are protected by automatic backups. Full database backups are done weekly, differential database backups are done hourly, and transaction log backups are done every five minutes. Automatic backups are retained for 14 days.

For more information, see [Learn about automatic SQL Database backups](#).

Can I request a copy of the backup of my production database?

No, this is not currently supported.

Can I get training in Business Central?

Yes, you can. As a partner you have access to the Dynamics Learning Portal, where you can find eLearning courses for Business Central. For more information, see the [Microsoft Dynamics 365 training page](#).

How can I troubleshoot my customers' online tenants?

You can use the **Help and Support** page in your customers' tenants to find technical information, and they can use that page to contact you. For more information, see [Configuring Technical Support for Dynamics 365 Business Central](#).

See Also

[FAQ for Developing in AL](#)

[Features not implemented in on-premises deployments of Dynamics 365 Business Central](#)

[Software Lifecycle Policy and Dynamics 365 Business Central On-Premises Updates](#)

[Welcome to Dynamics 365 Business Central](#)

Features not implemented in on-premises deployments of Dynamics 365 Business Central

3/31/2019 • 2 minutes to read

This topic lists features that are available in Business Central but not in on-premises deployments. The topic is divided into two sections:

- The first section lists features that are available under very specific circumstances in on-premises deployments.
- The second section lists features that are not intended for use with on-premises deployments. There are no plans to implement these features.

Features that require specific circumstances

The following features are not available in all on-premises deployments because they require specific circumstances.

FEATURE	DESCRIPTION
Read/write data with Excel add-in	The Excel add-in that enables update of data requires Azure Active Directory as the authentication mechanism.
Excel financial reports	The Excel add-in that is used with the predefined Excel-based financial reports requires Azure Active Directory as the authentication mechanism.
Coversheets for contact management	The integration with Word to create the coversheets requires Azure Active Directory as the authentication mechanism.
Built-in Power BI reports and charts	The integration with Power BI requires Azure Active Directory as the authentication mechanism.
Built-in Microsoft Flow Management	You can use the built-in workflows in on-premises deployments of Business Central, provided that you connect to Microsoft Flow using Azure Active Directory as the authentication mechanism. This can be done using the Azure Active Directory Assisted Setup guide in Business Central. Microsoft Azure and Microsoft Flow require Azure Active Directory authentication; however, your Business Central on-premises deployment does not have to use Azure Active Directory as the general authentication mechanism.
Built-in web services	A number pages and queries are exposed as web services. However, the default endpoint must be manually updated before the web services can be consumed.
Outlook add-in	The Outlook add-in requires Dynamics NAV User/Password or Azure Active Directory as the authentication mechanism.

FEATURE	DESCRIPTION
Standard REST API	Business Central contains new standard REST APIs. However, on-premises deployments cannot be reached through Microsoft Graph or the common endpoint, <code>https://api.businesscentral.dynamics.com/v1.0/api/beta</code> . Instead, you must connect directly to the on-premises deployment, just as when you connect to web services.
Sales and Inventory Forecast	This functionality requires an Azure Machine Learning subscription.
Image Analyzer	This functionality requires an Computer Vision service.
Cortana Intelligence in Cash Flow Forecast	This functionality requires an Azure Machine Learning subscription.

Features not intended for use in on-premises deployments

The following features are not intended for use in on-premises deployments. There are no plans to implement these features in on-premises deployments.

FEATURE	DESCRIPTION
Inviting the external accountant	Integration with Dynamics 365 — Accountant Hub is not supported in on-premises deployments of Business Central.
Default Power BI reports	Automatic deployment and configuration of Power BI reports is not supported in on-premises deployments of Business Central.
Bulk Invoicing from Microsoft Bookings	Integration with the Bookings app in Office Business Premium is not supported.
Create workflow from Flow	Microsoft Flow does not integrate with on-premises workflow functionality. You cannot create new workflows based on existing Microsoft Flow templates in on-premises deployments of Business Central.
Sandbox environments	The sandbox environment that you can use to develop extensions against for the new developer experience cannot connect to an on-premises deployment. For more information, see Get started with the Container Sandbox Development Environment .
In-product search	In online deployments of Business Central, Tell Me, the in-product search, also searches in content on the docs.microsoft.com site. For on-premises deployments, this is not supported.
Late Payment Prediction	The Late Payment Prediction functionality is not supported in on-premises deployments of Business Central.

See Also

[System Requirements](#)

Resources for Help and Support for Dynamics 365 Business Central

6/25/2019 • 3 minutes to read

As a Business Central partner, you have access to resources that can help you support your customers the users of Business Central, and you have access to resources that can help you be more productive as a partner.

This page outlines the resources available to you.

Product Help

The functionality in the default version of Business Central is described on the Docs.microsoft.com site as described in the following table.

NAME	LOCATION	DESCRIPTION
Business Central docs	https://docs.microsoft.com/dynamics365/business-central	Use this library to learn about business functionality.
Business Central developer and ITpro docs	https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/	Use this content to learn how to extend, customize, and administrate Business Central.

Customize and extend the user assistance

When a prospect signs up for a trial of Business Central, they have access to user assistance according to the Business Central user assistance model. If you customize Business Central, you are expected to also customize the user assistance so that users will have access to content that can help them get started, get unblocked, and learn more. For more information, see [User Assistance Model](#) and [Configure the Help Experience](#).

Support

As a Business Central reselling partner, you are an administrator of your customers' Business Central tenants, and you are the first line of support. You can customize the support experience, and you have access to information that can help you troubleshoot any issues that your customers report.

For more information, see [Technical Support](#).

Get started with the "Ready to Go" program

The "Ready to Go" program is designed to support you in the journey of bringing offerings to market. The program contains learning, coaching, and tooling. For more information, see [The "Ready to Go" Program](#).

Get an overview of role-specific training material from Microsoft in the [The "Ready to Go" learning catalog](#).

Learn about current or upcoming capabilities

You can learn about current and coming capabilities through a number of different resources as outlined in the following table.

NAME	LOCATION	DESCRIPTION
------	----------	-------------

NAME	LOCATION	DESCRIPTION
Release plans	https://docs.microsoft.com/dynamics365/release-plans/	Get an overview of upcoming and recently released capabilities in Business Central and other Dynamics 365 apps.
Business Central docs	https://docs.microsoft.com/dynamics365/business-central	Use this content to learn about business functionality.
Business Central developer and ITpro docs	https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/	Use this content to learn how to extend, customize, and administrate Business Central.
The "Ready to Go" learning catalog	https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/readiness/readiness-learning-catalog	Get an overview of role-specific training material from Microsoft.

Share an idea about a new feature you'd like to have

On the [Dynamics 365 Ideas site](#), you can provide suggestions for new feature and capabilities. Your input goes directly to Business Central's engineering backlog for investigation and prioritization.

Make sure to search through the list of submitted suggestions, as chances are that someone already submitted something similar and might have already received votes. Vote if an idea already has been submitted to get it prioritized on the team's backlog.

Business Central Community

On the [Business Central Community site](#), you have access to a number of different resources as outlined in the following table.

NAME	LOCATION	DESCRIPTION
Business Central Forum	https://community.dynamics.com/business/f	Use this forum to submit a question and learn from other Business Central community members. MVPs, Partners, and Microsoft employees participate in the conversations.
Business Central on the Dynamics 365 Blog for users	https://cloudblogs.microsoft.com/dynamics365/users/product/business-central/	Use this to learn more about new Business Central and app releases, tips and tricks, as well as updates about new country releases.
Business Central on the Dynamics 365 Blog for partners	https://cloudblogs.microsoft.com/dynamics365/it/product/business-central/	Use this blog to learn about opportunities, processes, and tools for the Business Central partner community.

Summary of where to file bugs and issues

- For collaboration in preview or beta versions of the AL language in Visual Studio Code, use [GitHub](#)
- For Microsoft partners, for versions already in general availability and supported, file requests through [Microsoft PartnerSource](#)
- For Microsoft partners who have discovered a bug in a in preview or beta version, use [MS Collaborate](#)

Trials

Giving prospects access to a pre-configured trial of Business Central is an elegant way to introduce them to Business Central. You can use the standard trial provided by Microsoft, or you can prepare your own including relevant extensions.

For more information, see [Take prospects and customers online](#).

See also

[Technical Support](#)

[Configuring the Help Experience](#)

[Migrate Legacy Help](#)

[The "Ready to Go" Program](#)

[The Business Central Administration Center](#)

[Blog post: Find the right resources and provide feedback](#)

Configuring Technical Support for Dynamics 365 Business Central

5/3/2019 • 4 minutes to read

As a Business Central reselling partner, you are an administrator of your customers' Business Central tenants, and you are the first line of support. This means that you will get requests for support from your customers that you must triage, investigate, and either resolve or escalate to Microsoft.

In this section, you can learn about the tools that are available to you to help you troubleshoot your customers' Business Central.

Configuring the support experience

In your customers' Business Central tenants, the [Help and Support](#) page gives them access to resources that can help unblock them. You can customize the page to include the email address that your customers must use to contact you.

To set up this support email address, run page 9165 in your customer's tenant, and then choose if you want to use the email that you are logged in with, or if you want to specify a different contact email address. To use the email account that you are logged in with, choose the *Use my authentication email* link; otherwise, enter the relevant email address.

For more information on how to run a page, see [Web Client URL](#).

On-premises deployments

In on-premises deployments of Business Central, the **Help and Support** page does not contain the section for contacting technical support. Instead, you can enter an agreement with your customer's administrator about how and when to contact you.

There are two other links in the **Help and Support** page that you can customize:

- Blog
 - Specifies a link to where your customers can access a blog about their solution
- Coming soon
 - Specifies a link to where your customers can access a roadmap for future changes

If you choose to not modify these settings, then the links go to Microsoft's blog and release notes.

For more information, see [Configuring Business Central Web Server Instances](#).

NOTE

The **Help and Support** page is available only in the browser.

Finding technical information

The **Help and Support** page is a powerful tool for you to find technical information about your customers' Business Central, both online and on-premises. It gives easy access to the most recent error message, and it has a link to inspect pages for further troubleshooting. This is also where you can find information about which version

of Business Central, your online customers are on.

You can use the Business Central Administration center to easily navigate to your customers' tenants, and you can create sandbox environments that can help troubleshoot any issues reported by your customers. For more information, see [The Business Central Administration Center](#).

Azure Active Directory tenant

If you have configured the support email address, and your customer uses that to contact you, then the **Help and Support** page encourages them to include information about their Azure Active Directory tenant ID in the email. This information is shown at the bottom of the **Help and Support** page. You can use that to identify the tenant in the administration center, and you can use it to log into their tenant.

Version

You can use the information about which version the tenant is on to help you troubleshoot the issue that the customer has reported, for example. This information is listed in the **Troubleshooting** section of the **Help and Support** page in the following format:

VERSION	EXAMPLE	DESCRIPTION
Platform <major>.<minor>.<build>.<revision>	14.0.29537.0	Specifies the full platform version, which includes client and server components.
Application <build> (<country> <name> <major>.<minor>)	29537 (US Dynamics NAV 14.0)	Specifies the build number for the application, including the major version number.

However, if you use the online administration center, the version information is rendered differently:

VERSION	EXAMPLE	DESCRIPTION
Platform	14.0	Specifies the platform version, which includes client and server components.
Application <major>.<minor>.<build>.<revision>	14.0.29537.0	Specifies the full version number for the application.

The numbers are updated based on Microsoft's builds. In the default version of Business Central online, platform and application have the same major version number but different build numbers. If you perform a technical upgrade of Business Central on-premises, then platform and application will have different versions. The October'18 update was major update 13, and the April'19 update is major version 14.

For more information about build versions, see the blog post by our technical evangelist at [Business Central Build Numbers](#).

Last known error

The link behind the sentence *View the last known error* will find and show the most recent error message that was generated by the application code. This includes errors from field validation, posting routines, and other code behind business functionality.

The link cannot open errors that were generated by the platform. So if you suspect that the issue is caused by the platform, you can try to reproduce the error in a sandbox environment before you contact Microsoft for support. For more information, see [Create a sandbox environment](#).

See Also

[Inspecting and Troubleshooting Pages](#)

[The Business Central Administration Center](#)

[Deployment](#)

[Working with Administration Tools](#)

[Blog post: Business Central Build Numbers](#)

Dynamics 365 Business Central User Assistance Model

3/31/2019 • 6 minutes to read

The Business Central user assistance model is based on the following principles:

- Get started

Default values and setup wizards makes it easy to start using Business Central with your own data, in-product videos give new users a quick introduction to how the product works, and Home pages give easy access to key tasks so each user can easily get started with their work every day.

- Get unblocked

Embedded user assistance implemented as tooltips answers most immediate questions about what fields and actions do.

- Learn more

The Help menu and the tooltips provide context-sensitive links to Help articles with more information.

Apps, extensions, and customizations are expected to follow the same model by applying tooltips to controls on page objects, and by providing links to Help for their functionality. For more information about customizing and extending the user assistance, see [Configure the Help Experience](#).

In this article, we'll talk about the user assistance model itself and what it does.

Help users get started

The user assistance concept of *Get Started* is not just about getting started with Business Central on the first day. It's also about getting started all the other days, and about getting started with infrequent and unfamiliar tasks.

Assistance in the shape of wizards is very helpful for setting things up, or filling in data for a complicated report, for example. Designing Home pages that are truly designed for that particular role or job is also very useful in helping users get started with their work every day – they can easily get to their most important tasks, and that means that Business Central helps them get their work done more efficiently.

Help users get unblocked

Even the best designed user interface can still be confusing to some. It can be difficult to predict what users will find confusing, and that is why the base application includes tooltips for all controls and actions. In combination with descriptive captions and instructional text, the tooltips are our current implementation of *embedded user assistance*, which is an important principle in today's world of software design.

The tooltips help users unblock themselves by providing an answer to the most likely questions the users might have, such as "What data can I input here?" or "What is the data used for?". Keep that in mind when you develop the user interface of your solution.

Guidelines for tooltip text

The Microsoft user assistance model requires a tooltip for all controls of type Action and Field that exist on page objects. Follow these guidelines:

- If the control is a field, begin with the verb *Specifies*.

- If the control is an action, begin with a verb in the imperative form , such as *Calculate* or *View*.
- Include the most valuable information that users need to perform the task(s) that the field or action supports.
- Ensure relevance.
For example, for the Post action, do not write *Post the document*. Write, for example, *Update ledgers with the amounts and quantities on the document or journal lines.*
- Describe complex options in tooltips for option fields.
Use a colon to call out the option name and its description. See example 3 below.
- Try to not exceed 200 characters including spaces.
This makes the tooltip easier to scan so the user can get unblocked quickly. However, the UI will render longer tooltip text if you want to provide more detailed user assistance.
- Do not make line breaks in the tooltip text.
The UI cannot render formatting or line breaks in tooltips.

Examples:

CONTROL NAME	TOOLTIP
Password field	Specifies your company's password to the service that converts bank data. The password that you enter in this field must be the same as on the service provider's sign-on page. (175 characters including spaces)
Entries action	View the history of transactions that have been posted for the customer. (72 characters including spaces)
Account Type field	Specifies the purpose of the account. Total: Used to total a series of balances on accounts from many different account groupings. To use Total, leave this field blank. Begin-Total: A marker for the beginning of a series of accounts to be totaled that ends with an End-Total account. End-Total: A total of a series of accounts that starts with the preceding Begin-Total account. The total is defined in the Totaling field. (522 characters including spaces)

Help users learn more

The content that Microsoft publishes under the user assistance concept of *Learn more* is in part intended to answer those questions that the user interface (including the tooltips) cannot answer, such as where that page fits into the bigger workflow, or what comes next, or what would be the alternative, and so on.

The base version of Business Central uses content that is published to an online library ([Docs.microsoft.com/dynamics365/business-central](https://docs.microsoft.com/dynamics365/business-central)) so that it can also serve as onboarding material and as feature overviews that you can share with prospects. The content is written in Markdown, and our source files are available in a [public GitHub repo](#) so that you can extend and customize it for your customers.

There are different repos in GitHub for the source content and each of the languages that Microsoft translates to. For more information, see [Extend, Customize, and Collaborate on the Help](#).

Feedback and contributions

On docs.microsoft.com, each article has two buttons at the end of the article. The *Product feedback* button sends you to the Ideas site, and the *Sign in to give documentation feedback* button lets you submit feedback about the content through GitHub. In both cases, you must create an account if you do not already have one. For *product feedback*, you must sign in with your work or organizational email account. For *access to GitHub*, you can use any email address when you create an account.

We welcome your contributions, both as pull requests with suggestions or corrections to the content, and as GitHub Issues with bugs or questions. But please be mindful that feedback and contributions to the *dynamics365smb-docs* repo is about the content, not about the product.

IMPORTANT

Microsoft accepts pull requests to the *dynamics365smb-docs* repo only, not the language-specific repos. If you have feedback about translations, you can report a GitHub issue in the relevant repo.

Microsoft also accepts contributions and feedback about the developer and ITpro content through the [dynamics365smb-devitpro-pb](#). This repo does not have translation repos associated with it, but other than that, the same rules apply as for the *dynamics365smb-docs* repo.

For more information, see [Extend, Customize, and Collaborate on the Help](#).

Working in Markdown

If you fork one of our repos, you will be authoring in something called Markdown. We recommend that you learn the basics by referring to the Docs contributor guide. For more information, see [How to use Markdown for writing Docs](#).

The team that built the Docs.microsoft.com site have also developed an extension for Visual Studio Code that helps with Markdown validation, for example. For more information, see [Docs Authoring Pack for VS Code](#). However, you can also use other text editors.

For other tips and tricks, see [Blog post: Collaborate on content for Business Central](#).

Translate the Help

If you want to deliver a [localization app](#), or if you want to deliver your functionality in more than one country, you will want to translate the Help. To help you do that, we suggest that you take a look at the [Microsoft Dynamics 365 Translation Service](#), which is available as preview in the Microsoft Dynamics Life Cycle Services. For more information, see [Translate documentation files](#).

The user assistance in the shape of tooltips and other user interface text is translated as part of the application. For more information, see [Working with Translation Files](#).

See Also

[Configure the Help Experience](#)

[Adding Help Links from Pages, Reports, and XMLports](#)

[ToolTip Property](#)

[InstructionalText Property](#)

[Development of a Localization Solution](#)

[Translate documentation files](#)

[Resources for Help and Support](#)

[Blog post: Extending and customizing the Help](#)

[Blog post: Collaborate on content for Business Central](#)

[Docs Contributor Guide](#)

[Docs Authoring Pack for Visual Studio Code](#)

[Style Guide for Microsoft Dynamics NAV \(requires login\)](#)

Extend, Customize, and Collaborate on the Help for Dynamics 365 Business Central

3/31/2019 • 16 minutes to read

The Help for the base application is available in a public GitHub repo so that you can easily extend and customize for your customers. In this section, you can learn about working with the GitHub repos and Markdown files.

GitHub repos

There are different repos in GitHub for the source content and each of the languages that Microsoft translates to. The [dynamics365smb-docs](#) repo contains the content in English (US). If you want access to the content in other languages, navigate to the relevant repo - the names follow this pattern: `dynamics365smb-docs-pr.<language>-<country>`, such as [dynamics365smb-docs-pr.da-DK](#) for the Danish version.

NOTE

Microsoft accepts pull requests to the *dynamics365smb-docs* repo only, not the language-specific repos. If you have feedback about translations, you can report a GitHub issue in the relevant repo.

When Microsoft publishes an update to the content, the *live* branch in the corresponding GitHub repo is updated. The source repo is updated monthly, and the related language-specific repos are updated less frequently as new translations are made available. You can choose to update your fork with updates from the Microsoft repo on a monthly or less frequent basis depending on your preferred work processes. The GitHub platform and tooling will help you manage any potential merge conflicts if you have made changes to the same files as Microsoft has. For more information, see [Fork a repo](#).

TIP

You are not required to make your GitHub repos public. When you fork a public repo, you can specify in the settings for the new repo if the repo is public, private, or available only to specific GitHub accounts.

For guidance about the Microsoft-provided content for Business Central, see [User Assistance Model](#).

Get started with GitHub

1. Fork the right repo

You cannot work directly in the Business Central repos in the MicrosoftDocs GitHub org, such as the `dynamics365smb-docs` repo, so the first thing you need to do is create a fork of the repo under your GitHub account. A fork basically is copy of this repo that lets you work freely on the content without affecting the MicrosoftDocs/dynamics365smb-docs repo. For more information, see [Fork a Repo](#).

2. Install GitHub Desktop (optional) and clone your forked repo.

GitHub Desktop makes is easy to work and collaborate with repos locally from your own desktop. For more information, see [GitHub Desktop](#).

3. Get hold of your favorite Markdown editor, and start making changes.

The help content is stored in the *business-central* folder of the repo. Articles use a syntax for formatting text called GitHub Flavored Markdown, which is widely popular in the Markdown community. To learn more

about working with markdown, see [Getting started with writing and formatting on GitHub](#).

If you want to work locally, you can edit using any text editor. Just save the file as a .md type. Here are a couple good tools that provide you with some nice features, such as Preview:

- [Visual Studio Code](#) with the [Docs Authoring Pack for Visual Studio Code](#)
- [Atom](#) (this has spell check and is good for managing many files)

You can also find guidance for how to get started with Markdown in the [Docs Contributor Guide](#), which is published by the team that built the Docs.microsoft.com site where the Business Central team publishes their docs.

Get updates from Microsoft

Microsoft makes frequent changes to the Business Central content, and those changes show up in the public GitHub repos. The base repo, MicrosoftDocs/dynamics365smb-docs, is updated weekly, and the translations are updated monthly. But you can choose to get updates monthly, twice a year, or once a year, for example. That is entirely up to you.

When you decide it is time to get the latest version of the content from Microsoft, you can do that using GitBash or GitHub Desktop. In the Help for GitHub, you can see [an example of how this works in GitBash](#), but in GitHub Desktop, you simply use the *Merge into current branch* menu item to pull changes from the origin into your fork.

However, if your solution is available in more than one country, then you are likely to want to make content available in multiple languages. Microsoft has a GitHub repo for each supported language, but the configuration files are only available in the English (US) source repo, MicrosoftDocs/dynamics365smb-docs. To help you get the content that you need, you might want to run a PowerShell script that picks up content from the various GitHub repos.

The following example is based on a script that a Danish partner developed in order to get the Microsoft source for a number of languages, and then build HTML files for the legacy Dynamics NAV Help Server. The script is provided in agreement with the partner without further support.

```
$languages = $("da-dk","de-ch","de-de")
$git = "C:\Program Files\Git\cmd\git.exe"
$docfx = "C:\GitHub\DocFx\docfx.exe"
$365docs = "C:\GitHub\MSFT\dynamics365smb-docs"
$langDir = "c:\GitHub\MSFT\dynamics365smb-docs-pr."

Start-Process -FilePath $git -ArgumentList "clone --single-branch --branch live
https://github.com/MicrosoftDocs/dynamics365smb-docs.git" -WorkingDirectory "C:\working\Help" -Wait
foreach ($language in $languages)
{
    $arguments = $("clone --single-branch --branch live https://github.com/MicrosoftDocs/dynamics365smb-docs-
pr." + $language + ".git")
    Start-Process -FilePath $git -ArgumentList $arguments -WorkingDirectory "C:\working\Help" -Wait
    Copy-Item $($365docs + "\business-central\NAVdocfx.json") $($langDir + $language + "\business-central")
    Copy-Item $($365docs + "\business-central\media") $($langDir + $language + "\business-central") -Recurse -
Force
    Copy-Item $($365docs + "\business-central\LocalFunctionality") $($langDir + $language + "\business-
central") -Recurse -Force
    Copy-Item $($365docs + "\Templates") $($langDir + $language) -Recurse -Force
    Set-Content -Path $($langDir + $language + "\business-central\NAVdocfx.json") -Value (get-content -Path
$($365docs + "\business-central\NAVdocfx.json") | Select-String -Pattern 'dest': "c:\working\output", ' -
NotMatch)
    Start-Process -FilePath $docfx -ArgumentList $("C:\working\help\dynamics365smb-docs-pr." + $language +
"\business-central\NAVdocfx.json" + " --output c:\working\output\" + $language)
```

Because the Microsoft repos are public, you do not need a valid GitHub account in order to get the content. However, we recommend that your organization has a system account with access to GitHub at a minimum.

Contributing

A benefit of GitHub is the ability for you to contribute to the core content that the Microsoft team provides in the `dynamics365smb-docs` repo. For example, you might have a new article that you think would be beneficial or you might have a correction to an existing article. If you would like to contribute to the `MicrosoftDocs/dynamics365smb-docs` repo, you create what is called a *pull request* from your repo to the `MicrosoftDocs/dynamics365smb-docs` repo. The Microsoft team will then review the request and include the changes as appropriate.

For example, to create a pull request to the `MicrosoftDocs/dynamics365smb-docs` repo by using GitHub Desktop, do the following:

1. Commit the changes to your repo that you want to include in the pull request. Here is the command for Git Shell:

```
git add -u
git commit -m "update doc"
git push
```

2. Choose **Sync** to push the changes up to your repo on GitHub.
3. When the sync is completed, choose **Pull Request**, make sure that the pull request points at the *live* branch, and then choose send **Pull Request**.

Building HTML files

For publishing to your own website, you can use tools such as [DocFx](#). For example, if you want to preview your content locally, or if you want to publish to the legacy Microsoft Dynamics NAV Help Server. DocFX is an open source tool for converting markdown files. This section provides some guidance on how you can use DocFX to publish HTML files for the Dynamics NAV Hep Server.

1. Install DocFX on your computer.

For more information, see [DocFx](#).

2. Specify the output folder in which to store the generated HTML files.

By default the files will be saved in the folder `c:/output`. The output folder is set in the `NAVdocfx.json` file. If you want to change this folder, do the following:

- a. In the folder where your local clone is, such as `C:\GitHub\MSFT\dynamics365smb-docs\business-central`, open the `NAVdocfx.json` file in your preferred editor.
- b. Set the **"dest:"** parameter to your output folder, and save the changes.

3. Go to your desktop and open a command prompt.
4. Go to the docfx installation folder.
5. Run the following command:

```
docfx "c:\GitHub\MSFT\dynamics365smb-docs\business-central\NAVdocfx.json"
```

The files are generated as `.html` files and stored in the specified output.

NOTE

The root of the MicrosoftDocs repos contain files that are related to internal Microsoft processes, such as .openpublishing.build.ps1. These scripts are used to validate and preview content, but they rely on internal Microsoft resources that are not publicly available.

Authoring in Markdown

The content is styled using a Markdown syntax as described below.

Headings

Use `#` for headings.

Examples: `#` Heading 1, looks like:

Heading 1

`##` Heading 2, looks like:

Heading 2

`###` Heading 3, looks like:

Heading 3

Bulleted lists


Use `-` to create bullets, for example:

The following options are available:

- first option
- second option
- third option

Ordered lists

Use numbers for ordered lists. No space between the lines, we'll let the template take care of that.

1. Choose the  that opens the Tell Me feature(media/ui-search/search_small.png "Tell me what you want to do") icon, enter **Payment Journal**, and then choose the related link.
2. In the **Payment Journal** window, on the first journal line, enter the relevant information about the payment entry.
3. To apply a single vendor ledger entry:
4. In the **Applies-to Doc. No.** field, choose the field to open the **Apply Vendor Entries** window.

Bold and italics syntax

Use `**bold**` and `*italics*`

Tables

For tables in the body, use the markdown syntax.

```
| To | See |
|-----|
|<text>|<link>|
| | |
| | |
```

For nested tables in ordered and unordered lists use HTML-syntax. Markdown does not support tables very well. If you use the markdown syntax the list will be broken, the table will align left and list will be renumbered.

Comment syntax

Useful for sections that are not ready and will not pass the build check.

```
<!-- Comments -->
```

Examples

```
<!-- [Managing Payables](payables-manage-payables.md)-->
<!-- This is a paragraph that spans more lines and I can just put the comment tag
at the beginning and end of it -->
```

Links

Ordinary link to a different topic in the same folder

These links have the format `[link text](filename.md)`.

Example: `[Managing Payables](payables-manage-payables.md)`

Link to a topic in a subfolder of the source topic

These links have the format `[link text](subfolder/filename.md)`.

For example, you want to link to `payables-manage-payables.md` from `ui-work-general-journals.md`, where the folder structure is as follows:

- articles
 - `ui-work-general-journals.md`
- ManagePayables
 - `payables-manage-payables.md`

Here is the link: `[Manage Payables](ManagePayables/payables-manage-payables.md)`

Link to a topic in a different folder than source topic

These links have the format `[link text](../folder/filename.md)`.

For example, you want to link to `payables-manage-payables.md` from `receivables-manage-receivables.md`, where the folder structure is as follows:

- articles
 - ManageReceivables
 - `receivables-manage-receivables.md`
 - `ui-work-general-journals.md`
 - ManagePayables
 - `payables-manage-payables.md`

Here is the link: `[Manage Payables](../ManagePayables/payables-manage-payables.md)`

Link to a place in the same article

From within an article, you can create a link to a specific heading in the same article. You can create the link like other links except with the following format:

```
[link text](#target-heading)
```

target-heading is the text of the heading that you want to link to, except it is all lowercase and spaces between words are replaced with hyphens. For example, here is the link: `[How Autoscaling Works](#how-autoscaling-works)`

To the heading: `## How Autoscaling Works`

Link to a place in a different article

From an article, you can create a link to a specific heading in another article. You can create the link like other links except with the following format:

```
[link text](targetarticlename#target-heading)
```

targetarticlename is the file name of the article, including the .md file type. target-heading is the text of the heading that you want to link to, except it is all lowercase and spaces between words are replaced with hyphens.

For example, to link to the heading "How Autoscaling Works" in the article Autoscaling.md", add the following code: `[link text](Autoscaling.md#how-autoscaling-works)`

Line breaks (soft return)

In the editor, add two blank spaces at the end of the sentence and hit return. This is used in the See Also list. (See Also must be heading 2.)

Continue steps after a non-step para

Enter four spaces in front of the non-step para. Otherwise, the non-step para will restart the step sequence.

TOC

The TOC structure of the TOC file is as follows:

```
#[Overview](overview.md)
##[Topic 1](topic-1.md)
##[Topic 2](topic-2.md)
##[Topic 3](topic-3.md)
##[Topic 4](topic-4.md)
```

Standard Phrases

All fields in Business Central have tooltips. Therefore, do not document fields in Help. To refer readers to the tooltips, use this standard phrase where relevant:

"Choose a field to read a short description of the field or link to more information." For more information, see [Dynamics 365 Business Central User Assistance Model](#).

Topic Titles

- Use imperative verb form for step-based topics ("Pay vendors").
- Use gerund verb form for conceptual, non-step topics. ("Paying Vendors")
- Use nouns for highest-level topics. ("Sales")

File Naming

Rules

- No spaces or punctuation characters. Use hyphens to separate the words in the file name.
- Use all lowercase letters
- No more than 80 characters - this is a publishing system limit

- Use action verbs that are specific such as develop, buy, build, troubleshoot. No -ing words.
- No small words - don't include a, and, the, in, or, etc.
- All files must be in markdown and use the .md file extension.

Examples

TOPIC TITLE	NAMING
Select a Company	ui-how-select-company.md
Enter Criteria in Filters	ui-enter-criteria-filters.md
Troubleshooting: Record Locked by Another User	ui-troubleshoot-record-locked-another-user.md
Changing Basic Settings	ui-change-basic-settings.md
Sales	sales-manage-sales.md
Set Up Currencies	finance-setup-currencies.md
Set Up Purchasers	purchases-how-setup-purchasers.md
Understanding Session Timeouts	admin-understand-session-timeouts.md
Manage Data Encryption	admin-manage-data-encryption.md

Country-specific content

To simplify content localization and translation, country-specific articles live in country-specific folders. The TOC entries live under the "Local Functionality" parent node.

See also

[Business Central User Assistance Model](#)

[Configuring the Help Experience](#)

[Docs Contributor Guide](#)

[Docs Authoring Pack for Visual Studio Code](#)

[Getting started with writing and formatting on GitHub](#)

[Visual Studio Code](#)

[Atom](#)

[DocFx](#)

Configure Context-Sensitive Help

3/31/2019 • 4 minutes to read

A key pillar of helping users help themselves is to give them access to Help for the particular corner of Business Central that they are working in.

App-level configuration

At an app level, you can specify where the Help for your functionality is published in the app.json file. For example, if you publish your content to `https://mysite.com/en-us/mysolution`, then you would specify that in the `contextSensitiveHelpUrl` property as shown in the following example:

```
"contextSensitiveHelpUrl": "https://mysite.com/documentation",
```

In this example, the `contextSensitiveHelpUrl` property specifies that the links to the Help must go to the *mysite.com* site when the user is using your app's functionality across all locales. When the user is using functionality from the base application, then the Help calls will go to the default location on the *docs.microsoft.com* site.

If your app only supports a limited number of locales, you can specify that as well as shown in the following example:

```
"contextSensitiveHelpUrl": "https://mysite.com/{0}/documentation",  
"supportedLocales": [  
  "en-GB", "en-IE"  
],
```

Localization apps

Specifically for localization apps that bring Business Central to new markets, the properties in the app.json file can be set to take over the links to Help for specific languages as shown in the following example:

```
"helpBaseUrl": "https://mysite.com/{0}/documentation",  
"supportedLocales": [  
  "ca-es"  
],
```

In this example, the `helpBaseUrl` and `supportedLocales` properties specify that the links to the Help must go to the *mysite.com* site when the user is using the product in either English (Ireland) or English (United Kingdom). If the user switches the application language to English (US), then the Help calls will go to the default location on the *docs.microsoft.com* site.

Page-level configuration

Your target website is expected to have a default page that will display if nothing else is specified. But for each page or page extension, and for each field or field group on those pages, you can then specify the exact Help page that describes this page or field. You can do that using the `ContextSensitiveHelpPage` property as shown in the following example:

```
page 50101 "Reward Card"
{
    PageType = Card;
    SourceTable = Reward;
    ContextSensitiveHelpPage = 'sales-rewards';
}
```

In this example, the app contains a page object that is mapped to the *sales-rewards* Help file on the website that the app.json specifies. As a result, the *Learn more* link in the tooltips for this page will go to the equivalent of <https://mysite.com/documentation/sales-rewards>.

You can use the *ContextSensitiveHelpPage* property to direct all Help calls to the same article, or to group the Help calls based on individual features or workflows. For example, Microsoft has chosen to group the context-sensitive links depending on the granularity of the Help for specific area in the base application. If the Help for a specific area is made more granular, then the context-sensitive Help mapping is updated accordingly.

You can set the *ContextSensitiveHelpPage* property on all pages, or only on those that you don't want to get the default Help page for your website. For page extensions, the value of the *ContextSensitiveHelpPage* property will apply only to the controls that the page extension adds to the extended page objects. For example, if your page extension adds two new controls to the base application's Customer Card page, then the *Learn more* links in the tooltips for those two controls will go to the Help page that you have specified, and the *Learn more* links in the rest of the controls will go to the default Help that is specified in the base application. This way, multiple apps can extend the same page object and each apply their own content-sensitive Help link without overwriting the context-sensitive links for other apps.

In contrast, the app.json file also contains a *help* property, but this specifies the link that describes the app or solution itself and is used by AppSource.

How it works for the base application

In the current version of Business Central, the context-sensitive links to Help for the base application works in a different way that is based on a UI-to-Help mapping that is stored in table 2000000198 **Page Documentation**. In this table, all page objects in the default version of Business Central are listed, and have a target Help article associated with each of them. This means that multiple page objects can be associated with the same Help article, such as when a specific workflow involves multiple pages.

The table associates page IDs with target articles, but the URL to where to find the target article is specified at the application level that defaults to the <https://docs.microsoft.com/dynamics365/business-central/> site. In an extension, you can overrule this URL so that all calls for Help go to your site instead, for example. This is especially important for localization apps where all context-sensitive Help calls for that app's language must go to that app provider's website.

See also

[User Assistance Model](#)

[Resources for Help and Support for Dynamics 365 Business Central](#)

[Adding Help Links from Pages, Reports, and XMLports](#)

[Migrate Legacy Help to the Business Central Format](#)

[Development of a Localization Solution](#)

[Blog post: Extending and customizing the Help](#)

[Blog post: Collaborate on content for Business Central](#)

[Docs Contributor Guide](#)

[Docs Authoring Pack for Visual Studio Code](#)

Deployment of Dynamics 365 Business Central

3/31/2019 • 2 minutes to read

The topics in the Deployment section are intended to help an administrator configure a Business Central solution online or on-premises.

Take prospects and customers online

You can give prospects a quick introduction to Business Central by asking them to get a [free trial](#), and by showing them the apps in [AppSource](#), for example. With Business Central online, data is stored in the Microsoft cloud, removing the need to install SQL Server locally, for example.

You must enroll in the Cloud Solution Provider program in order to service Business Central online. For more information, see [Cloud Solution Provider program - selling in-demand cloud solutions](#). In the Microsoft Partner Center documentation, you can also learn how to [add a customer](#), [assign licenses to users](#), and [create new subscriptions](#). Business Central is one of the subscriptions that you can create, and there are Business Central-specific license types that you can assign to users.

For more information about reseller readiness for Business Central, see [Build Your Business on Dynamics 365 Business Central](#).

When to choose on-premises deployment

There can be many reasons to prefer to deploy Business Central on-premises rather than using the cloud solution.

Key Features of Setup for On-Premises Deployments

With Business Central Setup, you can:

- Install different components on different computers.
- Choose from a selection of predefined installation options, or create your own custom list of components and options to install.
- Preconfigure components before installation.
- Create, save, or load Setup configuration files that capture your selection of components and configuration information.

You use Setup to install software and to create custom deployments that you can distribute to different users across a company.

Installation Notes

- Before installing Business Central components on a computer, you must remove (uninstall) any previous versions.
- All components must be from the same version and build of Business Central for the software to run correctly.
- If you have either SQL Server 2000 or Microsoft SQL Server Desktop Engine (MSDE) installed on a computer where you want to install Business Central, then you must remove it before you begin installing. The presence of either of these database products causes a Setup error.

Configuring the Help Experience

Part of your configuration is to specify where to look up the Help for the solution. For on-premises deployments, you can choose to install the legacy Help Server, for example. For more information, see [Configuring the Help Experience](#).

See Also

[Upgrading to Business Central](#)

[System Requirements](#)

Configuring the Help Experience for Dynamics 365 Business Central

4/29/2019 • 4 minutes to read

The default version of Business Central comes with conceptual overviews and other articles that publish to the <https://docs.microsoft.com/dynamics365/business-central/> site. This location is then accessible from the Help menu and through the Learn More links in all tooltips. Each extension that you add will include its own tooltips and links to Help. But what if you want to deploy Business Central locally? Or if you have a vertical solution so that you want to refer your customers to your own website for Help? Or if you have a legacy Help collection based on the Dynamics NAV Help Server?

These and other scenarios are also supported in Business Central. But the options and possibilities are different, depending on your deployment scenario.

Apps for online tenants

When you build an app for inBusiness Central using the AL developer experience, you are expected to comply with the Business Central user assistance model. This includes tooltips and context-sensitive links to Help.

For more information, see [User Assistance Model](#) and [Configure Context-Sensitive Help](#).

On-premises deployments

For deploying Business Central on-premises, you must choose between using the legacy Dynamics NAV Help Server or an online website. Help Server is a simple website that requires your Help to be in a specific format (HTML files), and the online website can host any content that you want to make available. Your choice depends on the concrete needs of your solution and your users.

IMPORTANT

You can configure each client to use either an online library or Help Server. If you add configuration for an online library, you must remove the settings for Help Server.

Online library

If you want to use a website that is not based on Help Server, then you must specify the URL in the settings for the Web client and the Windows client, if your company uses this legacy client. The website does not have to be publicly accessible, but it must be accessible to all users of the solution that it support.

For the Web client, which is accessed by users from a browser or from the mobile apps, the navsettings.json file must contain the following settings:

```
//BaseHelpUrl": "The location of Help for this application.",
"BaseHelpUrl": "https://mysite.com/{0}/documentation/",
"//BaseHelpSearchUrl": "The URL to use if Help is included in the Search functionality in Business
Central.",
"BaseHelpSearchUrl": "https://docs.microsoft.com/{0}/search/index?search={1}&scope=BusinessCentral",
"//DefaultRelativeHelpPath": "The Help article to look up if no other article can be found.",
"DefaultRelativeHelpPath": "index",
```

For users who use the legacy Windows client connected to Business Central, the ClientUserSettings.config file

must contain the following settings:

```
<add key="BaseHelpUrl" value="https://mysite.com/{0}/documentation/" />
<add key="DefaultRelativeHelpPath" value="index" />
```

NOTE

Replace the value of the BaseHelpUrl key with the URL for your own website, such as

`https://mysite.com/{0}/documentation/`. The parameter, {0}, represents the locale of the browser that the user is using, such as en-us or da-dk, and is set automatically at runtime.

Help Server

If you want to use Help Server, then you must specify the server and port in the installation options. The Help Server website can also serve as a starting point for adding a library to your existing website, for example.

For the Web client, which is accessed by users from a browser or from the mobile apps, the navsettings.json file must contain the following settings:

```
"//HelpServer": [
  "Name of the Dynamics NAV Help Server to connect to."
],
"HelpServer": "https://myserver.com",
"//HelpServerPort": "The listening TCP port for the Dynamics NAV Help Server. Valid range: 1-65535",
"HelpServerPort": "49000",
```

For users who use the legacy Windows client connected to Business Central, the ClientUserSettings.config file must contain the following settings:

```
<add key="HelpServer" value="https://myserver.com" />
<add key="HelpServerPort" value="49000" />
```

In both examples, <https://myserver.com> represents the URL to the Help Server instance. For more information, see [Configuring Microsoft Dynamics NAV Help Server](#) in the developer and ITpro content for Dynamics NAV.

IMPORTANT

If you use Help Server, the UI-to-Help mapping functionality that is described in [Configure Context-Sensitive Help](#) does not work. Instead, you must rely on the legacy Help lookup mechanism that hinges on .HTM files with filenames that reflect the object IDs, such as N_123.htm for the page object with the ID 123. For more information, see [Working with Dynamics NAV Help Server](#).

For guidance about how to generate HTML files, see the [Readme.md in the public source repo for the business functionality content](#). Optionally, you can choose to reuse the HTML and .HTM files that you used for Dynamics NAV in your online library or Help Server deployment.

Fork the Microsoft repos

If you want to customize or extend the Microsoft Help, you can fork our public repo for either the source repo in English (US) at <https://github.com/MicrosoftDocs/dynamics365smb-docs>, or one of the related repos with translations into the supported languages. For more information, see [Extend, Customize, and Collaborate on the Help](#).

See Also

[User Assistance Model](#)

[Adding Help Links from Pages, Reports, and XMLports](#)

[Working with Dynamics NAV Help Server](#)

[Configuring Microsoft Dynamics NAV Help Server](#)

[Migrate Legacy Help to the Business Central Format](#)

[Development of a Localization Solution](#)

[System Requirements](#)

[Resources for Help and Support](#)

[Blog post: Extending and customizing the Help](#)

[Blog post: Collaborate on content for Business Central](#)

[Docs Contributor Guide](#)

[Docs Authoring Pack for Visual Studio Code](#)

Choosing Your Dynamics 365 Business Central Development Sandbox Environment

3/31/2019 • 2 minutes to read

To get started developing for Dynamics 365 Business Central it is important to understand the different options you have at hand. You can either choose to run a sandbox environment deployed as a Dynamics 365 Business Central service, or you can run a container-based image either hosted as an Azure VM or locally. Both options provide the AL development tools; the container-based sandbox additionally provides access to the C/SIDE development tools. You can also choose to run a sandbox environment with production data using the **Business Central Admin Center**. For more information, see [Business Central Admin Center](#).

NOTE

When you publish an app to the online sandbox for testing, it is published within the scope of the service node that is hosting the sandbox. Upgrading the sandbox to a new version means that the sandbox is moved to another node that is running the new version. All apps are removed before the sandbox is moved because they will not be available on the new node. However, the data of the app is not removed, so you only have to re-publish and install the app to make it available. Apps that are published to the production environment are published within a global scope and downloaded to the service node and installed during the upgrade, which means that they will not disappear.

Sandbox Overview

The following topic outlines the most important capabilities on the offered development sandbox environments for Dynamics 365 Business Central.

CAPABILITY	ONLINE SANDBOX	CONTAINER SANDBOX
Deployment	Dynamics 365 Cloud Service managed by Microsoft	Azure VM or on-premises managed by ISV/VAR
Production data	Manually uploaded using Rapid Start packages. Or, available through the Business Central Admin Center .	Manually uploaded using Rapid Start packages
Production services	Manually configured	Not available
Cost	Part of the Business Central subscription	Locally hosted - free, Azure-hosted - cost incurred
Development	Full capabilities of the development environment. Designer functionality, such as: Add/Remove components, Move components, Set/clear Freeze pane, Edit captions	Full capabilities of the development environment. Designer functionality, such as: Add/Remove components, Move components, Set/clear Freeze pane, Edit captions
Tools	Visual Studio Code, Designer	Visual Studio Code, Designer, on-premise tools such as SQL Server Management Studio, and C/SIDE.

CAPABILITY	ONLINE SANDBOX	CONTAINER SANDBOX
Debugging	Enabled	Enabled
Database access	No	Yes
Extensions	Must be manually installed.	Must be manually installed.
From AppSource	Available.	Not available.
From File	Not available.	Available.
From Visual Studio Code	Available.	Available.

Getting Started

Based on the overview above and the requirements for your development environment, you can get started with a sandbox by following the links below:

- [Online Sandbox with Demo Data](#)
- [Online Sandbox with Production Data](#)
- [Container Sandbox](#)

See Also

[Getting Started with AL](#)

[Keyboard Shortcuts](#)

[AL Development Environment](#)

Embed App is in limited preview, and is still under development. We are currently not onboarding additional partners. This content is provided for informational purposes so you can learn more about Embed App offering. Stay tuned for upcoming announcements in early 2019.

What is an Embed App?

Embed App is a term that defines an end-to-end solution meeting the specific needs of a vertical or micro-vertical industry.

Dynamics 365 Business Central plays a vital role in the Embed App, as Business Central is embedded as an integral part of the overall solution.

Some examples of an Embed App include:

- A Dentist solution
- A Real Estate Agent solution
- A Food Processing solution

An Embed App refers to what is being provided to a given customer segment, unrelated to how the solution is being implemented or architected. An Embed App can be built using AL, in other words extension, code-customization (C/AL), and a combination of extensions and code-customization.

App + Platform Vision for Dynamics 365 Business Central

Packaged applications are the fastest way to transform your business:
e. g. **Dynamics 365 Business Central**, Marketing, Sales, Service, Talent and Finance and Operations apps

But no application is exactly what you need: every business needs to customize these applications to fit their processes and industry.

Dynamics 365 Business Central, Add-on and Connect Apps, Customizations (per-tenant apps)

And sometimes there is no "app for that": digital transformation depends on creating bespoke applications for the last mile

Embed Apps

Apps

Apps + Platform

Platform

See Also

[Microsoft Responsibilities Qualification and Onboarding](#)
[Qualification and Onboarding](#)
[Managing in Microsoft Lifecycle Services](#)
[Business Central Component](#)
[Business Central Platform](#)
[Business Central Licensing](#)
[Customer Signup](#)
[Appsource](#)

Embed App is in limited preview, and is still under development. We are currently not onboarding additional partners. This content is provided for informational purposes so you can learn more about Embed App offering. Stay tuned for upcoming announcements in early 2019.

Microsoft Responsibilities when Running an Embed App

Dynamics 365 Business Central is a cloud service for small to medium-sized businesses that is built on and for Microsoft Azure. It provides organizations with a business management solution that helps companies connect their financials, sales, service, and operations to streamline business processes, improve customer interactions and make better decisions.

The Dynamics 365 Business Central service brings together the business management solution, business intelligence, infrastructure, computing, and database services in a single offering that enables organizations to run industry-specific Embed Apps from Independent Software Vendors (ISVs), without the hassle of managing infrastructure.

The Dynamics 365 Business Central service model distinguishes specific roles and responsibilities for ISVs, Implementation Partners (VARs), and Microsoft throughout the life cycle of the service. Microsoft maintains the Dynamics 365 Business Central service by deploying, actively monitoring, and servicing the Embed App and customers' production tenants that are running on the service. This includes allocating the required system infrastructure to run the service and proactively communicating to customers about the service's health (which is done through the Service Health dashboard in the Office 365 Admin Portal).

Microsoft responsibilities in the Dynamics 365 Business Central service include:

AREA	RESPONSIBILITIES
Infrastructure	<ul style="list-style-type: none">• Storage and database capacity management• High availability and disaster recovery• Platform security and compliance• Infrastructure capacity, scaling in response to demand• Infrastructure management and deployment• Data center networking, power, and cooling
Application Platform	<ul style="list-style-type: none">• Availability and security• Diagnostics, patches, updates, hotfixes, and updates• Monitoring and first line support for ISVs
Lifecycle Services portal	<ul style="list-style-type: none">• Development, deployment, and support of the portal functionality• High availability and disaster recovery• Monitoring, updating and patching• First line support for ISVs

See Also

[Embed App Overview](#)
[Qualification and Onboarding](#)
[Qualification and Onboarding](#)
[Managing in Microsoft Lifecycle Services](#)
[Business Central Component](#)
[Business Central Platform](#)
[Business Central Licensing](#)
[Customer Signup](#)
[Appsource](#)
[Sandbox](#)
[Ecosystem Features](#)

Embed App is in limited preview, and is still under development. We are currently not onboarding additional partners. This content is provided for informational purposes so you can learn more about Embed App offering. Stay tuned for upcoming announcements in early 2019.

Embed App is in limited preview, and is still under development. We are currently not onboarding additional partners. This content is provided for informational purposes so you can learn more about Embed App offering. Stay tuned for upcoming announcements in early 2019.

Qualification and Onboarding of ISV Partners to Embed App

Embed App Qualification requirements are being finalized by the owners. The main criteria for Embed App Onboarding at this point are:

- Partner must provide all day, every day support. Their customers do not get to call Microsoft support. They always call the partner.
- Partner must live up to the standards for documentation that Microsoft outlined in the CFMD program. Good documentation is part of Embed App success and uptake, and it also lowers the work load on their support center.
- Partner is committed to stay on the supported platform.
- Partner provides an SLA for requests from Microsoft towards the partner (TBD)
- Partners must have support agreement with Microsoft

For Embed Apps coming with a (code-customized) Base App included will have additional volume (number of users) requirements associated with it. When an Embed App is qualified for onboarding into Dynamics 365 Business Central service, the ISV will need to provide the following information about the Embed App to the Microsoft's onboarding team:

- The name of the application to be used for the client and web service URL, for example, an ISV for the fabrikamapples app would provide the following information:
 - Client: <https://fabrikamapples.bc.dynamics.com>
 - Web Services: <https://fabrikamapples.api.bc.dynamics.com>
- Entitlements for their ISV objects, per license type. See the Licensing section for more detail.
The rest of the information will be included with the Embed App package that the partner deploys through LCS.

See Also

[What is an Embed App](#)
[Microsoft Responsibilities](#)

Embed App is in limited preview, and is still under development. We are currently not onboarding additional partners. This content is provided for informational purposes so you can learn more about Embed App offering. Stay tuned for upcoming announcements in early 2019.

Managing an Embed App in Microsoft Lifecycle Services

Microsoft is going to provide essential functionality and relevant services within [Microsoft Lifecycle Services](#) collaboration portal (LCS) to support qualified ISVs and their VARs in managing the Embed App in Dynamics 365 Business Central service.

In LCS, an ISV partner can create a project for each Embed App they would like to deploy to Dynamics 365 Business Central service. When creating a project, the partner will also select the country where the Embed App is expected to be available. The partner needs to create separate projects for each country, even if the Embed App is the same. Keeping different countries in different projects can help partner control access for other personas (CSS, VAR and other stakeholders) on the country level.

Inside of the LCS project, the partner then proceeds to upload the Embed App package into the Asset library and deploy the environment in Dynamics 365 Business Central service, based on the uploaded package. After the Environment is successfully provisioned, it is ready to accept customer signups, which can come from either CSP or from self-service (IW) signups.

Each business entity (tenant) that signed up for the Embed App is automatically added and displayed on the Tenant list page. On this page, the partner can find more details about the tenant, including the name and the URL to login into each one.

To upgrade the environment to a new version of the Embed App, the partner must first upload the updated Embed App package to the Asset library and then deploy it to the environment that is already running. During upgrade, the partner will be able to see the progress on the Tenant list page (version change) and more detailed logs, including detected errors, in the Environment monitoring section. VARs and customers will get notified about the scheduled upgrades and will be able to re-schedule those for more convenient time.

The ISV can define the date and time for the upgrade to start. The upgrade orchestration and execution is then performed by the Dynamics 365 Business Central service.

Before deploying the new version of the application into their Production environment, the partner will be able to deploy it into a Staging environment, where tenants (customers, VARs) that opted in can test the new version safely and provide feedback to the ISV before the ISV deploys the new version into Production environment.

The partner gets access to comprehensive logs for the activities that they perform in the portal:

- Deployment
- Tenant provisioning
- Tenant upgrade
- Runtime (platform)
- Runtime (application)

They can also observe service health metrics, the load on the service (user activity, telemetry), and get insights into the performance of their application and tenants.

In the upcoming updates, we will keep introducing more features and services for the partner to manage their

environment efficiently, as well as functionality for VARs who help ISVs manage their tenants.

See Also

[Embed App Overview](#)

[Microsoft Responsibilities](#)

[Qualification and Onboarding](#)

[Qualification and Onboarding](#)

[Managing in Microsoft Lifecycle Services](#)

[Business Central Component](#)

[Business Central Platform](#)

[Business Central Licensing](#)

[Customer Signup](#)

[Appsource](#)

[Sandbox](#)

[Ecosystem Features](#)

Embed App is in limited preview, and is still under development. We are currently not onboarding additional partners. This content is provided for informational purposes so you can learn more about Embed App offering. Stay tuned for upcoming announcements in early 2019.

Embed App Components and Capabilities

Because an Embed App should provide end-to-end experience for the customers, the partner designing this experience, must be able to affect and control more parts of the Dynamics 365 Business Central service experience.

Components

On a high level, an Embed App is a package that consists of the following parts:

- Library extensions

This is the functionality of the Embed App that is implemented by the ISV partner in a form of extensions.

- Third party extensions

These are add-on extensions coming from other ISVs that contribute to and enhance the Embed App. The extensions are validated to be compatible by the Embed App owner.

- Extended metadata

This includes additional Embed App properties that are specific to this type of app and not otherwise available for other types of apps (see the list below).

- Base application and tenant template (optional)

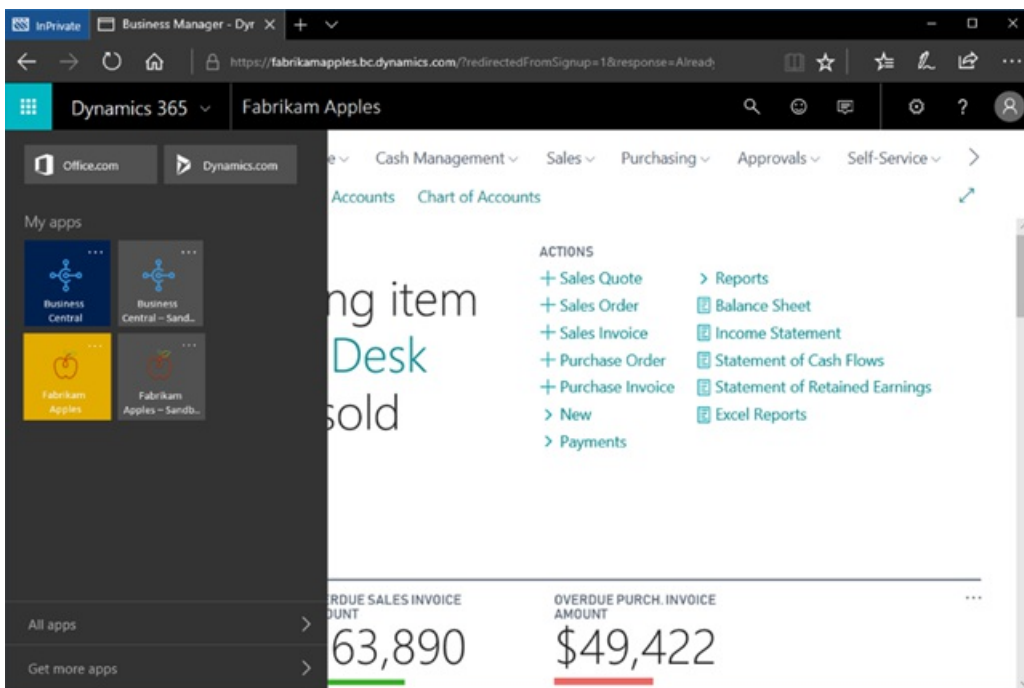
Capabilities

The following capabilities are only available for the Embed App and not for other types of Dynamics 365 Business Central apps (Connect and Add-on).

Partner Branding

The Embed App will promote the ISV brand in several places:

- Web Client and Web Service URLs
 - Client [https://\[application name\].bc.dynamics.com](https://[application name].bc.dynamics.com)
 - Web Services [https://\[application name\].api.bc.dynamics.com](https://[application name].api.bc.dynamics.com)
- Name, image, and icon on the “provisioning” page of the Fixed Client Endpoint
- Splash screen of the Web Client
- Title bar of the Web Client (for example, “Fabrikam Apples”)
- Dynamics Shell (<https://home.dynamics.com>) - a dedicated product tile, icon, and short marketing description
- In-product messages (such as pop-up errors, warnings, notifications)



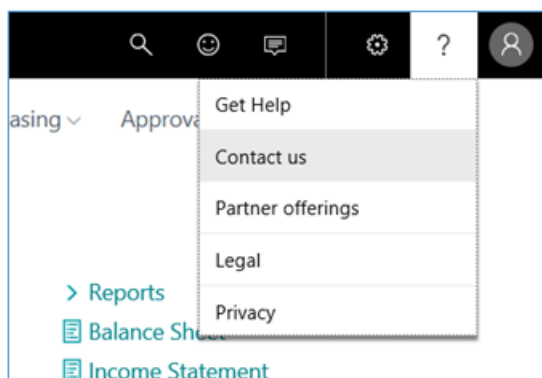
Exclusivity

The partner can control which third party apps can be installed for their Embed App.

- Whitelisting of the 3rd party apps - no other apps will be possible to install, except the ones explicitly approved by the ISV
- App install/uninstall controlled by the ISV
- The partner can choose to allow a customer to install other extensions from the AppSource, but this will be an explicit partner decision, not the default behavior

Additional settings (metadata):

- An Embed App is the property of the ISV partner, so the customers of the Embed App must be able to find the partner's own legal, privacy, contact, community and feedback links (not Microsoft links) when they work with the app:



- Whitelisted domains for embedding Embed App pages into other web sites, including SharePoint ("frame ancestors")
- Target version of Dynamics 365 Business Central platform
- Target version of Dynamics 365 Business Central base application (if not included with the Embed App)
- Azure KeyVault account for storing ISV application secrets (for example, accounts for connecting to 1-3rd party services)
- Base application + tenant template. This is an optional component of an Embed App. The partner can choose to include it or simply specify which version of the Dynamics 365 Business Central base application

the Embed App should use as a base application.

At this stage, within the extensions and base application, the ISV partner can work in their own Object ID range.

Base Application

The base application is the Dynamics 365 Business Central application which the partner obtained from Microsoft, customized and extended to fit the needs of the industry the Embed App supposed to cover. Major releases and cumulative updates (CUs) of the base application are publicly available on Microsoft Download and as images on Docker. Microsoft Download: <https://support.microsoft.com/en-us/help/4072483> Docker Hub: <https://hub.docker.com/r/microsoft/dynamics-nav/>

Pre-released versions of the base application are available for participants of the "Ready and Go" program via Microsoft Collaborate. Although we recommend always using the latest version of the base application, the ISV partner can choose any version they need. The only requirement is that the ISV partner makes sure that the base application version they include with the Embed App can run on a supported Dynamics 365 Business Central platform version.

The base application is an optional part of an Embed App package. If the partner has implemented all required functionality in their library extensions, they don't need to include the base application itself with the Embed App. Instead they should specify, in the metadata of the Embed App, which version of the Dynamics 365 Business Central base application they are targeting.

ISVs can choose to bring their own code-customized base application for several reasons:

- Shortening time-to-market ("lift and shift" approach).

The ISV's current solution is a significantly customized version of the Microsoft Dynamics NAV application and it will require substantial time and effort to migrate it into extensions. An ISV can lift their solution as-is (upgraded to a supported platform) to Dynamics 365 Business Central service and start offering it to their new and prospective customers. Then, they can gradually start moving their functionality into extensions to achieve the benefits that come with the extension model.

- Overcoming the limitations of the current extensions framework.

Extensions today can support many scenarios and the Microsoft team is working on extending these capabilities even more. However, some customization needs of the partners are not yet possible using the current version of the extensions framework, so partners can choose to do these changes directly in the base application using C/SIDE Development Environment.

- Availability of 3rd party add-ins as extensions.

Some of the add-ins required by the Embed App might not yet be available as extensions, for the reasons mentioned above. The partner can choose to import these add-ins as FOB files into the base application that they submit with the Embed App.

- Usage of .NET interoperability and custom assemblies.

ISV partners use .NET interoperability in their current application to address multiple business scenarios. Although extensions today allow a number of these scenarios to be implemented in AL, they don't and cannot cover for all possible scenarios of .NET usage. Therefore, the partner can choose to import the required .NET add-ins into the Add-ins table of the base application, and these add-ins will automatically be deployed into the environment where they will be running.

Microsoft recommends all ISV's to move towards a model where the code-customization of the base application is not used. In that future state, ISV's will be able to fully rely on AL and Extensions version 2.0 or later.

See Also

[Embed App Overview](#) [Microsoft Responsibilities](#)

Qualification and Onboarding

Managing in Microsoft Lifecycle Services Business Central Platform

Business Central Licensing

Customer Signup

Appsource

Sandbox

Ecosystem Features

Embed App is in limited preview, and is still under development. We are currently not onboarding additional partners. This content is provided for informational purposes so you can learn more about Embed App offering. Stay tuned for upcoming announcements in early 2019.

Dynamics 365 Business Central Platform

When deploying an Embed App to the Business Central service, the partner must ensure that it is compatible with a supported version of Business Central platform.

Minor updates

Microsoft will ship minor platform updates monthly and major platform updates every six months. Minor updates can include bug fixes and improvements which should not affect the compatibility of the platform with the previous version of the application. In rare situations, partners may be asked to recompile their solution to work with a minor update of a Business Central platform.

Major updates

Major updates will include changes that can require partners to perform a technical upgrade to make their application work with the new version of the platform.

Version availability

Microsoft is going to make new versions of the Business Central platform available to ISVs through the LCS portal. The partner will then have to pick the platform they want to use for deployment of their solution.

Deploying versions

When deploying a solution via the LCS portal, it will be possible to pick from the last 3 available versions of the platform (minor and major). Every newly released minor or major platform update will be added to the list and simultaneously one older version will be removed from that list.

The existing deployments, running on the platform versions, which are older than 3 updates will enter a grace period of 30 days and after that, if not upgraded, will be moved out of the standard SLA.

See Also

- [Embed App Overview](#)
- [Microsoft Responsibilities](#)
- [Qualification and Onboarding](#)
- [Qualification and Onboarding](#)
- [Managing in Microsoft Lifecycle Services](#)
- [Business Central Component](#)
- [Business Central Platform](#)
- [Business Central Licensing](#)
- [Customer Signup](#)
- [Appsource](#)

Embed App is in limited preview, and is still under development. We are currently not onboarding additional partners. This content is provided for informational purposes so you can learn more about Embed App offering. Stay tuned for upcoming announcements in early 2019.

Embed App Licensing

Embed Apps licenses can only be purchased through CSP. Microsoft offers several types of paid licenses (users):

- Essential
- Premium
- Team Member
- External Accountant

Customers can also subscribe for an evaluation version of an Embed App by using self-service signup (also known as IW or viral signup). This subscription comes with 10000 licenses and allows customers to evaluate the functionality of the Embed App using non-production companies.

In the Business Central service, the license files (*.flf) are not used, and a completely new way of defining the licensing permissions is implemented.

We define license permissions (per object) in the Entitlements table of the application database, Entitlements are grouped in the Entitlement Set table, then each Entitlement Set is linked to one of the Azure Active Directory (Azure AD) Service Plans that we offer:

- Team Member
- Essential
- Premium
- External Accountant

Therefore, when a user purchases, for example, an Essential license and tries to sign in to Business Central, we retrieve the user's Service Plan (in this case Essential) from Azure AD and then load its associated Entitlements as license permissions.

See Also

[Embed App Overview](#)

[Microsoft Responsibilities](#)

[Qualification and Onboarding](#)

[Qualification and Onboarding](#)

[Managing in Microsoft Lifecycle Services](#)

[Business Central Component](#)

[Business Central Platform](#)

[Business Central Licensing](#)

[Customer Signup](#)

[Appsource](#)

[Sandbox](#)

[Ecosystem Features](#)

Embed App is in limited preview, and is still under development. We are currently not onboarding additional partners. This content is provided for informational purposes so you can learn more about Embed App offering. Stay tuned for upcoming announcements in early 2019.

Embed AppCustomer Signup

A customer can sign up for any number of Embed Apps and for Business Central application using the same Org ID. These apps will run side-by-side with each other, will use different URLs and will be displayed as separate tiles on the home.dynamics.com portal. There are two ways for a customer (tenant, organization) to subscribe to an Embed App:

- Using the self-service IW signup – for acquiring a free evaluation version of the app.
- Through the Microsoft Partner Center Cloud Solution Provider (CSP) program by contacting the partner - for acquiring a paid production version of the Embed App.

Tenant provisioning is happening automatically (just-in-time) on the first attempt to login into the solution.

Navigating to <https://businesscentral.dynamics.com> will trigger provisioning of the Business Central tenant, while navigating to [https://\[application name\].bc.dynamics.com](https://[application name].bc.dynamics.com) will trigger provisioning of the tenant running on the "[application name]" application.

Self Service (IW) signup - evaluation

The partner can allow customers to use the self-service signup (also known as IW signup and viral signup) for their Embed App. To give customers self-service signup capability, the partner should prepare a signup URL that will redirect the Office signup flow to their application URL. The signup URL should have the following format:

```
https://signup.microsoft.com/signup?sku=6a4a1628-9b9a-424d-bed5-4118f0ede3fd&ru=https%3A%2F%2F[application name].bc.dynamics-TIE.com%2F%3FredirectedFromSignup%3D1
```

The partner can then pass the URL to their customers, either from the partner's own marketing page or in a welcome e-mail.

To work with an Embed App, the customers would use a URL that looks something like this:

- Client: [https://\[application name\].bc.dynamics.com](https://[application name].bc.dynamics.com)
- Web Services: [https://\[application name\].api.bc.dynamics.com](https://[application name].api.bc.dynamics.com)

To work with Business Central, they would use these URLs:

- Client: <https://businesscentral.dynamics.com>
- Web Services: <https://api.businesscentral.dynamics.com>

Partner initiated (CSP) signup – paid

In CSP, it is the Partner who defines the Partner-to-Customer price. Partners can use several options to charge their customers.

Option 1: Embed App price is added on top of Microsoft-to-Partner price:

Example (not actual prices):

	PARTNER-TO-CUSTOMER PRICE	PRICE
CSP	Essential	25+50=75 USD

Option 2: Embed App price is added as a 3rd party CSP offering (This functionality is still in development by the CSP team; tentative GA of this feature is December 2018):

Example (not actual prices):

	PARTNER-TO-CUSTOMER PRICE	PRICE
CSP	Essential	25
CSP	Fabrikam Apples (Essential)	50

Option 3: Business Central license + Embed App self-monetization

Example (not actual prices):

	PARTNER-TO-CUSTOMER PRICE	PRICE
CSP	Essential	25
External (e.g. www.stripe.com)	Fabrikam Apples (Essential)	50

In all three options, the ISV partner will be selling Business Central licenses in CSP.

See Also

[Embed App Overview](#)

[Microsoft Responsibilities](#)

[Qualification and Onboarding](#)

[Qualification and Onboarding](#)

[Managing in Microsoft Lifecycle Services](#)

[Business Central Component](#)

[Business Central Platform](#)

[Business Central Licensing](#)

[Appsource](#)

[Sandbox](#)

[Ecosystem Features](#)

Embed App is in limited preview, and is still under development. We are currently not onboarding additional partners. This content is provided for informational purposes so you can learn more about Embed App offering. Stay tuned for upcoming announcements in early 2019.

AppSource for Embed App

AppSource is a Market place where Embed App partners can provide marketing details (descriptions, whitepapers and videos) about their app.

Embed App partners can choose to promote themselves and their Embed App on the AppSource.

The Embed App itself (package) should not be uploaded into the AppSource, it is uploaded, deployed and tested via LCS. AppSource in this case is used for the marketing purposes, not as a repository of Apps.

Partner can submit the reference to their project in LCS in the AppSource, to make sure the Microsoft team, assessing the apps can access their environment (being invited via LCS) and verify that the necessary technical acceptance criteria are met.

See Also

[Embed App Overview](#)

[Microsoft Responsibilities](#)

[Qualification and Onboarding](#)

[Qualification and Onboarding](#)

[Managing in Microsoft Lifecycle Services](#)

[Business Central Component](#)

[Business Central Platform](#)

[Business Central Licensing](#)

[Customer Signup](#)

[Sandbox](#)

[Ecosystem Features](#)

Embed App is in limited preview, and is still under development. We are currently not onboarding additional partners. This content is provided for informational purposes so you can learn more about Embed App offering. Stay tuned for upcoming announcements in early 2019.

Embed App Sandbox

Embed App customers, just like Business Central customers, can choose to create a Sandbox environment in addition to their production environment and use it for developing and testing customizations, demos, trainings and similar non-production activities.

The sandbox feature is currently in preview in the Business Central service, so additional license fees can apply when it is made available officially.

For customizing their tenants, the partners and customers can use VS Code and 50000-50149 object ID range. Learn more about customizing the application at <http://aka.ms/BusinessCentralApps>

See Also

[Embed App Overview Microsoft Responsibilities](#)

[Qualification and Onboarding](#)

[Qualification and Onboarding](#)

[Managing in Microsoft Lifecycle Services](#)

[Business Central Component](#)

[Business Central Platform](#)

[Business Central Licensing](#)

[Customer Signup](#)

[Appsource](#)

[Ecosystem Features](#)

Embed App is in limited preview, and is still under development. We are currently not onboarding additional partners. This content is provided for informational purposes so you can learn more about Embed App offering. Stay tuned for upcoming announcements in early 2019.

Embed App Ecosystem Features

Business Central application is running in a rich ecosystem of other Microsoft and 3rd party services, which Embed App can decide to take advantage of.

The following integration capabilities of the Business Central should be considered:

- Microsoft Graph – [evaluating]
- Dynamics 365 API endpoint (aka native API) – available if the base application objects are unchanged
- Microsoft Office Outlook Add-in - available
- Microsoft Power BI – available (using customer own Power BI license)
- Microsoft Flow – available (using customer own Flow license)
- Microsoft PowerApps – available
- WalkMe (UI walkthroughs) – not available
- Microsoft Apps included with Business Central (Yodlee, Quick Books, OCR, AMC...) - available, but the partner needs to setup his own agreement with these service providers (if needed)
- Azure ML – available via partner's own Azure ML subscription
- CDS/CRM integration (with Dynamics 365 for Sales) – pending CDS/CRM decision (Microsoft)
- Accountant Hub - available (if shipped by Microsoft)

See Also

[Embed App Overview](#)

[Microsoft Responsibilities](#)

[Qualification and Onboarding](#)

[Qualification and Onboarding](#)

[Managing in Microsoft Lifecycle Services](#)

[Business Central Component](#)

[Business Central Platform](#)

[Business Central Licensing](#)

[Customer Signup](#)

[Appsource](#)

[Sandbox](#)

System Requirements for Dynamics 365 Business Central On-premises

4/10/2019 • 12 minutes to read

The following sections list the minimum hardware and software requirements to install and run Business Central on-premises. **Minimum** means that later versions (such as SP1, SP2, or R2 versions) of a required software product are also supported.

NOTE

Business Central Setup installs some software if it is not already present in the target computer. For more information, see the "Additional Information" section for each component.

Client Components

Web Client Requirements

The following table shows the minimum system requirements for the Business Central Web client on-premises.

Supported browsers	<ul style="list-style-type: none">• Microsoft Edge.• Internet Explorer 11.• Google Chrome 72.0 for Windows.• Mozilla Firefox 65.0 for Windows.• Safari 10.0 for macOS. <p>Cookies and JavaScript must be enabled in the browser.</p>
Business inbox in Outlook	<ul style="list-style-type: none">• Microsoft Office 365, Microsoft Office 2019, or Microsoft Office 2016.
Sending data to Excel	<ul style="list-style-type: none">• Microsoft Office 365, Microsoft Office 2019, or Microsoft Office 2016.
Editing in Excel using the Excel Add-in	<ul style="list-style-type: none">• Excel 2019, Excel 2016, or Excel Online. <p>For more information, see Exporting Your Business Data to Excel.</p>
SharePoint Online links	<ul style="list-style-type: none">• Microsoft Office 2019, Microsoft Office 2016, or Microsoft Office 365.
Printing reports to Excel or Word	<ul style="list-style-type: none">• Microsoft Office 2019, Microsoft Office 2016, or Microsoft Office 365.

Additional information	If you experience problems using the Business Central Web client, you can try to turn off browser tools, such as translator tools that may run in the background.
------------------------	---

Business Central Tablet Client and Phone Client (in a Browser) Requirements

The following table shows the minimum system requirements for the Business Central Tablet client and Business Central Phone client running in a browser when used for development and testing purposes.

Server component	Identical to the Business CentralWeb client.
Supported browsers	<p>The following desktop browsers are supported:</p> <ul style="list-style-type: none"> • Microsoft Edge • Internet Explorer 11 (build 11.0.9600.17239) for Windows 10. • Google Chrome 72.0 for Windows. • Mozilla Firefox 65.0 for Windows. • Safari 10.0 for macOS. <p>Cookies and JavaScript must be enabled in the browser.</p>

Business Central Universal App Requirements

The following table shows the minimum system requirements for the Business Central Universal App.

For the latest information, see the app in the Windows Store, App Store, or Google Play.



Supported operating systems	<ul style="list-style-type: none"> • Windows 10 S, Home, Pro, Enterprise, or Education (32-bit and 64-bit editions). • Android 6.0 or higher (tablet and phone). • iOS 10.0 or higher (iPad and iPhone).
Additional hardware	<ul style="list-style-type: none"> • 1 GB RAM for Android and Windows.
Additional software	<ul style="list-style-type: none"> • A third-party telephony or VoIP app such as Skype is required for placing calls from Business Central. • A third-party email program such as Outlook is required for sending emails from Business Central. • Microsoft Office 2019, Office 2016, or Office 365 is required for sending data to Microsoft Excel or to Microsoft Word.
Additional information	<ul style="list-style-type: none"> • Device diagonal screen size 7" for tablets. • Screen resolution 960 × 510 for tablets. • Device diagonal screen size 4" for phones. • Screen resolution 854 x 480 for phones.

Dynamics NAV Client connected to Business Central Requirements

The following table shows the minimum system requirements for using the Dynamics NAV Client connected to Business Central.

Supported operating systems	<ul style="list-style-type: none">Windows 10 Pro, Enterprise, or Education (32-bit and 64-bit editions). Important: Windows 10 S is not supported.Windows Server 2019 Standard, Essentials, or Datacenter.Windows Server 2016 Standard, Essentials, or Datacenter.Windows Server 2012 R2 Standard or Essentials (64-bit edition).
Hardware resources	<ul style="list-style-type: none">Hard disk space: 200 MB.Memory: 1 GB.
Reports	<ul style="list-style-type: none">For editing RDLC report layouts:<ul style="list-style-type: none">Report Builder for SQL Server 2016 or Visual Studio 2017 with Microsoft Rdlc Report Designer for Visual Studio installed.For editing Word layouts:<ul style="list-style-type: none">Microsoft Word 2016 or later
Outlook client integration and mail merge	<ul style="list-style-type: none">Microsoft Office 365, Microsoft Office 2019, or Microsoft Office 2016.
Import and export with Microsoft Excel and Office XML, and SharePoint links	<ul style="list-style-type: none">Microsoft Office 365, Microsoft Office 2019, or Microsoft Office 2016.
Editing in Excel using the Excel Add-in	<ul style="list-style-type: none">Excel 2019 or Excel 2016. For more information, see Exporting Your Business Data to Excel. For Business Central on-premises, see Setting up the Excel Add-In for Editing Data since the same steps apply to Business Central on-premises.
OneNote integration	<ul style="list-style-type: none">Microsoft Office 365, Microsoft Office 2019, or Microsoft Office 2016.
Email logging	<ul style="list-style-type: none">Active Directory and Microsoft Exchange Server 2019 or Exchange Server 2016.Microsoft Exchange Online, or Exchange Online as part of an Office 365 subscription.
Additional software	<ul style="list-style-type: none">Microsoft .NET Framework 4.7.2.

Additional information	<ul style="list-style-type: none"> • Business Central Setup installs the following software if it is not already present in the target computer: <ul style="list-style-type: none"> ◦ Microsoft .NET Framework 4.7.2. • The Dynamics NAV Client is available in a 32-bit version and 64-bit version. On a 32-bit Windows operating system, the 32-bit version is run. On a 64-bit Windows operating system, the 64-bit version is run by default; however, you can also run the 32-bit version if it is required. • Business Central Setup can only install the Excel Add-in if Excel is present on the target computer. • Outlook synchronization is not supported on 64-bit versions of Office.
------------------------	---

Dynamics NAV Development Environment Requirements

The following table shows the minimum system requirements for the Dynamics NAV Development Environment.

Supported operating systems	<ul style="list-style-type: none"> • Windows 10 Pro, Enterprise, or Education (32-bit and 64-bit editions). • Windows Server 2019 Standard, Essentials, or Datacenter. • Windows Server 2016 Standard, Essentials, or Datacenter. • Windows Server 2012 R2 Standard or Essentials (64-bit edition).
Hardware resources	<ul style="list-style-type: none"> • Hard disk space: 200 MB. • Memory: 1 GB.
Reports	<ul style="list-style-type: none"> • For creating and editing RDLC report layouts: <ul style="list-style-type: none"> ◦ Report Builder for SQL Server 2016, or ◦ One of the following versions of Visual Studio: <ul style="list-style-type: none"> ◦ Visual Studio 2017 with Microsoft Rdlc Report Designer for Visual Studio installed. ◦ Visual Studio 2015 Professional or Enterprise edition with SQL Server Data Tools installed. <p>Important: Before you install Visual Studio 2015, install Microsoft .NET Framework 4.7.2; otherwise, an error will occur when you compile or run RDLC reports. For more information, see Report error "Visual Basic Command Line Compiler has stopped working".</p> • For upgrading reports: <ul style="list-style-type: none"> ◦ Report Builder for SQL Server 2016 • For creating Word report layouts: <ul style="list-style-type: none"> ◦ Word 2016 or later

Additional software	<ul style="list-style-type: none"> • Microsoft .NET Framework 4.7.2.
Additional information	<ul style="list-style-type: none"> • Business Central Setup installs the following software if it is not already present in the target computer: <ul style="list-style-type: none"> ◦ Microsoft .NET Framework 4.7.2. ◦ SQL Server Native Client 11.0. ◦ Report Builder for SQL Server 2016. This is not installed if a version of SQL Server Report Builder or Microsoft Visual Studio is already present on the target computer • If the development environment and Business Central Server are on the same computer, then only a 64-bit operating system is supported.

Server Components

Business Central Server Requirements

The following table shows the minimum system requirements for Business Central Server.

Supported operating systems	<ul style="list-style-type: none"> • Windows 10 Pro, Enterprise, or Education (64-bit edition). • Windows Server 2019 Standard, Essentials, or Datacenter. • Windows Server 2016 Standard, Essentials, or Datacenter. • Windows Server 2012 R2 Standard or Essentials (64-bit edition).
Hardware resources	<ul style="list-style-type: none"> • Hard disk space: 500 MB. • Memory: 2 GB.
Dynamics 365 for Sales integration	<ul style="list-style-type: none"> • Windows Identity Framework. For a list of supported Dynamics 365 for Sales versions, see Microsoft Dynamics 365 for Sales Integration Requirements.
Additional software	<ul style="list-style-type: none"> • Microsoft .NET Framework 4.7.2. • Windows PowerShell 4.0.
Additional information	<ul style="list-style-type: none"> • Business Central Setup installs the following software if it is not already present on the target computer: <ul style="list-style-type: none"> ◦ Microsoft .NET Framework 4.7.2. ◦ Windows Identity Framework.

Business Central Web Server Components Requirements

Supported operating systems	<ul style="list-style-type: none"> • Windows 10 Pro, Enterprise, or Education (64-bit edition). • Windows Server 2019 Standard, Essentials, or Datacenter. • Windows Server 2016 Standard, Essentials, or Datacenter. • Windows Server 2012 R2 Standard or Essentials (64-bit edition).
Web server	<ul style="list-style-type: none"> • Internet Information Server 10, Internet Information Server 8.5, or Internet Information Server 8.0.
Additional software	<ul style="list-style-type: none"> • Microsoft .NET Framework 4.7.2. • Windows PowerShell 4.0.
Additional information	<ul style="list-style-type: none"> • Business Central Setup installs the following software if it is not already present on the target computer. <ul style="list-style-type: none"> ◦ Microsoft .NET Core 1.0 Windows Server Hosting. This is installed by Business Central Setup if not already present. ◦ Microsoft .NET Framework 4.7.2. ◦ Internet Information Server 10, Internet Information Server 8.5, or Internet Information Server 8.0, depending in the operating system, with the required features enabled. • For more information about configuring IIS, see Configuring IIS

Business Central Database Components for SQL Server Requirements

The following table shows the minimum system requirements for Business Central database components for SQL Server.

Supported operating systems	<ul style="list-style-type: none"> • Windows 10 Pro, Enterprise, or Education (64-bit edition). • Windows Server 2019 Standard, Essentials, or Datacenter. • Windows Server 2016 Standard, Essentials, or Datacenter. • Windows Server 2012 R2 Standard or Essentials (64-bit edition).
Hardware resources	For more information, see Hardware and Software Requirements for Installing SQL Server . From this page, you can also access requirements for other versions of SQL Server.

SQL Server	<ul style="list-style-type: none"> • Microsoft SQL Server 2017 Express, Standard or Enterprise. • Microsoft SQL Server 2016 Express, Standard or Enterprise. • Microsoft SQL Server 2014 Express, Standard or Enterprise. • Azure SQL Database Managed Instance, Elastic Pool, or Single Database.
Service Packs and Cumulative Updates	Unless explicitly stated, all released Service Packs and Cumulative Updates of the above Microsoft SQL Server versions are supported. It is recommended to always be on the latest released Service Pack and Cumulative Update.
Additional information	<p>Business Central Setup installs the following software if it is not already present on the target computer:</p> <ul style="list-style-type: none"> • SQL Server 2016 Express (64-bit edition). <p>If the operating system on the target computer does not support SQL Server 2016 Express, Setup displays a pre-requisite warning. In this case you should exit Setup and then update the operating system on the computer to one that does support SQL Server 2016 Express. Then run Setup again.</p>

Business Central Help Server Requirements

The following table shows the minimum system requirements for the Business Central Help Server.

Supported operating systems	<ul style="list-style-type: none"> • Windows 10 Pro, Enterprise, or Education (64-bit editions). • Windows Server 2019 Standard, Essentials, or Datacenter. • Windows Server 2016 Standard, Essentials, or Datacenter. • Windows Server 2012 R2 Standard or Essentials (64-bit edition).
Hardware resource	<ul style="list-style-type: none"> • Hard disk space: 500 MB. • Memory: 2 GB.
Web server	<ul style="list-style-type: none"> • Internet Information Server 10, Internet Information Server 8.5, or Internet Information Server 8.0.
Additional software	<ul style="list-style-type: none"> • Microsoft .NET Framework 4.7.2.

Additional information	<ul style="list-style-type: none"> • Business Central Setup installs the following software if it is not already present on the target computer. <ul style="list-style-type: none"> ◦ Microsoft .NET Framework 4.7.2. ◦ Internet Information Server 10, Internet Information Server 8.5, or Internet Information Server 8.0. depending on the operating system, with the required features enabled. • Windows Search must be enabled on the computer that you install the Business Central Help Server on. If you install on Windows Server 2012 R2, and Windows Search is not enabled as a file service, Business Central Setup adds the service. However, the changes do not take effect until the computer has restarted.
------------------------	---

Additional Components and Features

Automated Data Capture System Requirements

The following table shows the minimum system requirements for Automated Data Capture System (ADCS) for Business Central.

Additional software	<ul style="list-style-type: none"> • MSXML version 6.0. • Telnet or Microsoft Windows HyperTerminal. • VT100 Plug-in for each computer on which you install ADCS. • Microsoft Loopback Adapter.
Additional information	<ul style="list-style-type: none"> • HyperTerminal is no longer included with Windows. For more information, see What happened to HyperTerminal? in the Windows Help. • VT100 Plug-in acts as a virtual Telnet server.

Business Inbox in Microsoft Outlook Requirements

The following table shows the minimum system requirements for using Business Central as your business inbox in Outlook.

Supported Outlook Applications	<ul style="list-style-type: none"> • Outlook 2016 or later • Outlook Web App • OWA for iPad • OWA for iPhone • OWA for Android.
Supported Exchange Servers	<ul style="list-style-type: none"> • Exchange Online • Exchange Server 2019 • Exchange Server 2016 <p>In deployments that use Exchange Server, the Exchange PowerShell endpoint must be accessible by Business Central Server.</p>

Supported Authentication	<ul style="list-style-type: none"> The Business Central Server must be configured to run with NavUserPassword, ACS, or AAD Credentials Type. Also, the Business Central Web client must be configured for Secure Sockets Layer (SSL).
Supported Browsers	<ul style="list-style-type: none"> When using the Outlook Web App (OWA), your computer must be running a supported browser listed in the Business Central Web client Requirements.
Supported Operating Systems	<ul style="list-style-type: none"> When using OWA for iPad, OWA for iPad, or OWA for Android, your mobile device must use a supported Operating System listed in Business Central Universal App Requirements.

Microsoft Outlook Add-In Requirements

The following table shows the minimum system requirements for the Business Central Add-In for Outlook for synchronization with Outlook.

Supported Outlook Applications	<ul style="list-style-type: none"> Outlook 2019 Outlook 2016
Supported Exchange Servers	<ul style="list-style-type: none"> Exchange Server 2019 Exchange Server 2016 Exchange Online.

Microsoft Dynamics 365 for Sales Integration Requirements

The following table shows the product version requirements for integrating Business Central with Dynamics 365 for Sales, and the versions in which users can view the availability of items in Business Central from Dynamics 365 for Sales.

Sales/Dynamics NAV/Business Central	2015/Update 1/online	2016/Update 1/online	Sales Enterprise (v8.x)	Sales Enterprise and Sales Professional (v9.x)
Dynamics NAV 2016	Supported ***	Supported ***	Supported ***	Supported ***
Dynamics NAV 2017	Supported **	Supported *	Supported *	Supported *
Dynamics NAV 2018	Supported **	Supported *	Supported *	Supported *
Business Central (online)	Not supported **	Not supported **	Supported *	Supported *
Business Central (on-premises)	Supported **	Supported *	Supported *	Supported *

Legend:

- "*" item availability capability is supported.
- "***" integration solution can be installed from the Dynamics NAV 2016 DVD, but viewing item availability is not supported.
- "****" viewing item availability is not supported

NOTE

AD, IFD and Claims authentication types are supported for the 2015 and 2016 on-premises versions of Dynamics 365 for Sales. OAuth and Office 365 authentication are supported for the 2015, 2015 Update 1, and 2016 Update 1 online versions of Dynamics 365 for Sales. For more details on authentication types, see [Connection strings in XRM tooling to connect to Dynamics 365](#).

Business Central as an App for SharePoint Requirements

The following table shows the minimum system requirements for Business Central as an App for SharePoint.

Supported operating systems	<ul style="list-style-type: none">• Windows Server 2019 Standard, Essentials, or Datacenter.• Windows Server 2016 Standard, Essentials, or Datacenter.• Windows Server 2012 R2 Standard or Datacenter (64-bit edition).
Additional software	<ul style="list-style-type: none">• SharePoint 2013 Service Pack 1.• SharePoint Online.

See Also

[Welcome to the Developer and IT-Pro Help for Business Central](#)

[Product and Architecture Overview](#)

[Deployment](#)

Software Lifecycle Policy and Dynamics 365 Business Central On-Premises Updates

4/10/2019 • 2 minutes to read

This topic outlines the lifecycle and support policies for Dynamics 365 Business Central on-premises updates.

Fixed Lifecycle Policy

Dynamics 365 Business Central (on-premises) software is covered by the Fixed Lifecycle Policy.

Licensed customers must stay current with updates to the Dynamics 365 Business Central on-premises software in accordance with the following servicing and system requirements. This policy requires that the customer maintain Software Assurance (SA) or the Enhancement Plan, and that it deploy updates as noted later in this topic.

On-premises software update policies

The customer is in full control of its on-premises deployments and must follow this policy. The customer is in control of installing updates in its on-premises environments. Microsoft will support the Dynamics 365 Business Central (on-premises) software as indicated on the [Microsoft Lifecycle Policy for Business Central on-premises](#) page, but only if the customer keeps the deployed software current according to this policy.

Critical fixes and non-critical updates are handled in the following way:

- **Critical fixes** – Critical fixes include security fixes and any fixes that are required to support reliability and availability. Critical fixes will be made available in the latest platform update version.
- **Non-critical updates** – Customers must update to the most current Dynamics 365 Business Central to deploy non-critical updates.

RELEASE	VERSION	BUILD NUMBER	AVAILABILITY	MAINSTREAM SUPPORT ENDS
Dynamics 365 Business Central (on-premises)	October'18 Update	24630	November 1, 2018	**
Dynamics 365 Business Central (on-premises)	April '19 Update	29537	April 1, 2019	October 10, 2023
Dynamics 365 Business Central (on-premises)	October'19 Update			*

* Indicates that the expiration date will be added on this page when the next version is released.

** Indicates that the version is considered a service pack with a support end date of April 14, 2020.

The build number for the versions that are not yet available will be updated when the version is released.

See Also

[Microsoft Lifecycle Policy for Business Central on-premises](#)

[Configuring Technical Support](#)

[Welcome to the Developer and IT-Pro Help for Dynamics 365 Business Central](#)

[Upgrading to Dynamics 365 Business Central](#)

[Deployment of Dynamics 365 Business Central](#)

Running a Container-Based Development Environment

5/21/2019 • 3 minutes to read

Dynamics 365 Business Central is available as a container-based image, ready for running on a Windows system with Docker installed. The container-based approach is used when you need access to both the AL development environment and the C/SIDE development environment.

TIP

For more information on where to find Docker images, see [What Docker Image is Right for You?](#).

Install and configure Docker

Install Docker and configure it for Windows Containers.

1. Please choose the version of Docker that is appropriate for the host operating system.

- Use [Docker Community Edition](#) if the host operating system is Windows 10.
For more information, see [Install instructions](#).
- Use [Docker Enterprise Edition](#) if the host operating system is Windows Server.
For more information, see [Install instructions](#).

2. Switch Docker to use Windows containers. By default Docker uses Linux containers.

To switch to Windows containers, in the Taskbar, right-click the Docker icon , and then select **Switch to Windows Containers**. For more information, see [Switch between Windows and Linux containers](#).

Run the container-based image

Run the following command in a Command Prompt as Administrator to run a Docker image of Dynamics 365 Business Central:

```
docker run -e accept_eula=Y -m 4G microsoft/bcsandbox
```

NOTE

When you run the Docker run command, it will start downloading the image if it does not already exist. A container consists of multiple layers, only the needed layers are downloaded.

After starting the `docker run` command above, you will see log entries similar to the following:

```
Initializing...
Starting Container
Hostname is cdc633cdb0a2
...
Container IP Address: 172.20.203.209
Container Hostname : cdc633cdb0a2
Container Dns Name : cdc633cdb0a2
Web Client : https://cdc633cdb0a2/NAV/
NAV Admin Username : admin
NAV Admin Password : Biba4071
Files:
http://cdc633cdb0a2:8080/certificate.cer
Initialization took 83 seconds
Ready for connections!
```

At this point, you can open your internet browser and type in the Web client URL from the log. You will be prompted with a login dialog, where you can login with the NAV Admin Username/Password displayed.

NOTE

The container image uses a so called self-signed certificate for https communication. Because of that, your browser might warn you that the page you are requesting is unsafe. In those specific circumstances and only for test and development environments, it is safe to ignore this warning. If you want to solve this warning, you can install the certificate on your PC (see the link under "Files" in the log entries).

The NavContainerHelper module

To support the use of containers, optional PowerShell scripts are available, which support setup of development environments. Use the `NavContainerHelper` to work with containers. On a Windows 10 or Windows Server 2016 machine, start Powershell as an Administrator and type:

```
install-module navcontainerhelper -force
```

To see which functions are available in the NavContainerHelper module use the following command:

```
Write-NavContainerHelperWelcomeText
```

To get quickly get started, run the following command from the NavContainerHelper module:

```
new-navcontainer -accept_eula -containerName test -imageName microsoft/bcsandbox:<country> -usebestcontaineros
```

The `NavContainerHelper` will create a folder on the C:\ drive called DEMO and will place all files underneath that folder. The DEMO folder will be shared to the container for transfer of files etc. If you do not specify a username and a password, it will ask for your password and use the current Windows username. If you specify your windows password, the container setup will use Windows Authentication integrated with the host. The `NavContainerHelper` will also create shortcuts on the desktop for the Dynamics 365 Business Central Web client, a container prompt, and a container PowerShell prompt.

The `navcontainerhelper` module also allows you to add the `-includeCSide` switch in order to add the Dynamics 365 Business Central Windows client and C/SIDE to the desktop and export all objects to a folder underneath C:\DEMO\Extensions for the object handling functions from the module to work.

See Also

[Getting Started with AL](#)

[Get started with the Container Sandbox Development Environment](#)

[Keyboard Shortcuts](#)

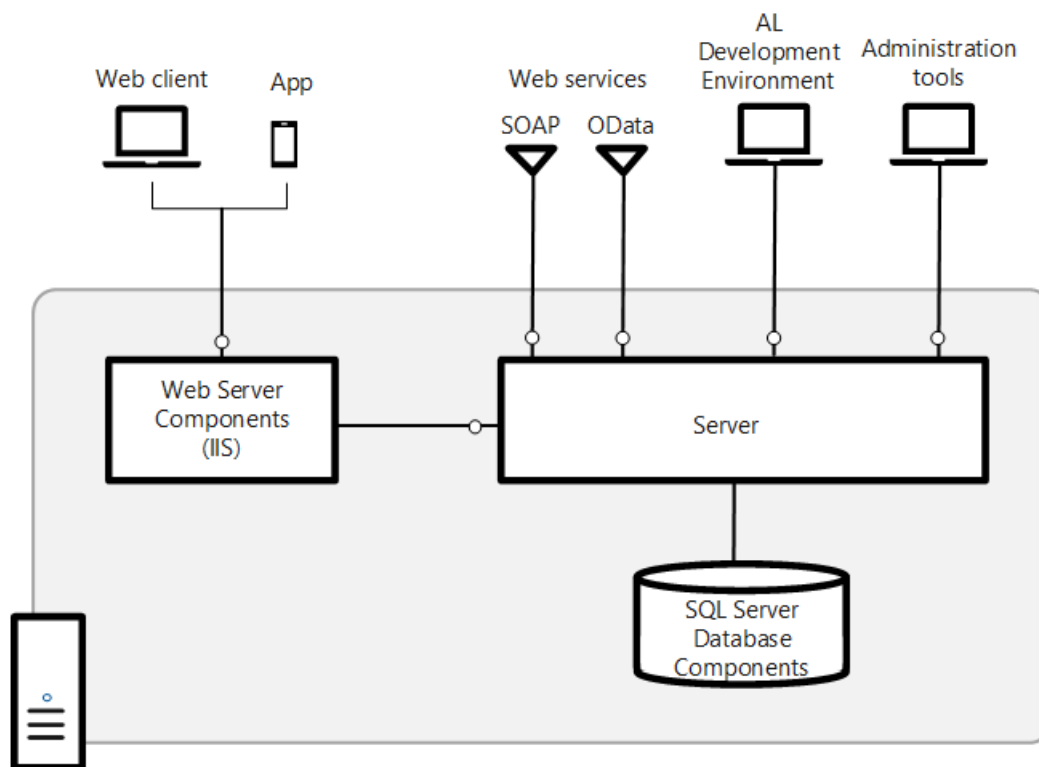
Business Central Component and System Topology

3/31/2019 • 3 minutes to read

The Business Central deployment comprises three core components for serving the application to users, plus various tools and components for managing, developing, extending, and testing the application.

Multi-Tier System Topology

To understand the components is useful to first look at the base topology of a Business Central deployment, as illustrated in the following diagram:



Components

Main components

Every deployment must include the core components: Web server, Server, and SQL Database.

COMPONENT	DESCRIPTION	MORE INFORMATION
SQL Database	An SQL Server or Azure SQL Database database that contains application object definitions and business data. In a multitenant deployment, the application and business data can be separated into different databases: the application database and the tenant, which is the a database that contains the business data. In this case, there can be one or more tenant for a single application database.	Creating Databases Deploy a Business Central Database to Azure SQL Database

COMPONENT	DESCRIPTION	MORE INFORMATION
Server	Business Central Server is a .NET-based Windows Service application that uses Windows Communication Framework to handle communication between clients and databases. It controls authentication, event logging, scheduled tasks, reporting and more.	Configuring Business Central Server
Web Server	An Internet Information Server (IIS) web site, provisioned with the Business Central Web Server components, that enables access from the Business Central Web client and mobile apps.	Business Central Web Server
Business Central App	A desktop, phone, and tablet app for Business Central.	Windows Store App Store Google Play
Web services	SOAP and OData Web Services for exposing application functionality to external systems and users. Developers can create and publish functionality as web services, where they expose pages, codeunits, or queries, and even enhance a page web service by using an extension codeunit.	Web Services

Development and administration components

COMPONENT	DESCRIPTION	MORE INFORMATION
AL development environment	An AL language extension for Visual Studio Code for developing applications and extensions.	Getting Started with C/SIDE and AL for On-Premises.
Business Central Server Administration tool	A Microsoft Management Console (MMC) for creating and configuring Business Central Server instances.	Business Central Server Administration Tool
Business Central Administration Shell	Windows PowerShell modules for managing the deployment, including tasks such adding and configuring Business Central Server and Web server instances, databases, and users, and administering extension packages.	Windows PowerShell Cmdlets for Business Central

Additional components

COMPONENT	DESCRIPTION	MORE INFORMATION
Demo Database	A database that contains application objects and sample business data for demonstration purposes.	

COMPONENT	DESCRIPTION	MORE INFORMATION
Dynamics NAV Development Environment	The C/SIDE client that was available in Dynamics NAV for developing applications using C/AL. In Business Central, this is only required for performing upgrades but you can still use it to develop applications.	Development in C/AL in the Dynamics NAV Developer and IT Pro Help.
Dynamics NAV Development Shell	Windows Powershell modules for merging and modifying application object files and creating extension packages. Installed with the Dynamics NAV Development Environment.	Windows PowerShell Cmdlets for Business Central
Dynamics NAV Client connected to Business Central	Windows Desktop application for accessing Business Central.	
Microsoft Outlook Integration	A Business Central Server component for integrating with Microsoft Outlook.	
Microsoft Outlook Add-in	A component to synchronize data, such as to-dos, contacts, and tasks, between Business Central and Outlook. The Outlook Add-In uses Business Central web services.	Setting Up the Office Add-Ins for Outlook Integration
Microsoft Excel Add-in	A component that enables users to export data from Business Central to Excel.	Setting up the Excel Add-In
Page Testability	A Business Central Server component for testing pages.	
Automated Data Capture System	A system that tracks the movement of items in a warehouse.	
ClickOnce Installer Tools	Tools for implementing ClickOnce installation for the Dynamics NAV Client connected to Business Central.	Deploying Microsoft Dynamics NAV Windows client Using ClickOnce in the Dynamics NAV Developer and IT Pro Help.
NAS Service	A server component that executes business logic without a user interface or user interaction. NAS services in Business Central Server support applications such as Microsoft Office Outlook Integration and the NAV Job Queue.	Instead of using NAS services, we recommend that you use the Task Scheduler (see Task Scheduler). If you decide to use NAS, and want to read more about its configuration, see Configuring NAS Services in the Dev and IT Pro Help for Microsoft Dynamics NAV 2018.

See Also

[Deployment](#)

[Installing Business Central Using Setup](#)

[Multitenant Deployment Architecture](#)

Planning Your Dynamics 365 Business Central Deployment

3/31/2019 • 2 minutes to read

This article outlines some of the things you should consider and decide on before you install Business Central.

Most of the topics discussed in this article can be changed at any time after the initial installation.

Network Topology

A Business Central deployment consists of various components that support the production, development, and testing. These components can be installed on various computers. The deployment process varies depending on the topology that you implement.

For more information, see [Deployment Topologies](#).

Single-tenancy and Multitenancy

By default, Business Central is installed as a single-tenant deployment. This means that the application and the business data is stored in the same database. You can also set up a multitenant deployment, where the application and business data reside in separate databases.

In a multitenant deployment, information about the Business Central application is stored in a separate application database. Your customers' data is stored in separate business databases, each of which is a *tenant* in your deployment. By separating application from data, you can deploy the same solution to many customers with centralized maintenance of the application and isolation of each tenant, which in turn, makes upgrading easier compared with a single-tenant deployment.

For more information, see [Multitenant Deployment Architecture](#).

User Authentication

Business Central supports several credential mechanisms for authorizing users trying to access data. By default, Windows authentication is used.

For more information, see [Authentication and Credential Types](#).

Business Central Server Service Account

The central component of a Business Central deployment is the Business Central Server, which handles all communication between the client and the databases. The Business Central Server requires a log on account, referred to as the service account. By default, the Network Service Account is used, which is acceptable in a test environment, but we recommend that you use a domain account your production environment.

For more information, see [Provisioning the Business Central Server Service Account](#).

Enhancing Connection Security

Business Central offers features that help secure connections over a wide area network (WAN), such as connections from the Business Central Web Server, Dynamics NAV Client connected to Business Central, and web services to the Business Central Server. The implementation of these security features require that you obtain a

certificate from a certification authority or trusted provider.

For more information, see:

[Using Security Certificates with Business Central On-Premises](#)

[Configuring SSL to Secure the Business Central Web Client Connection](#)

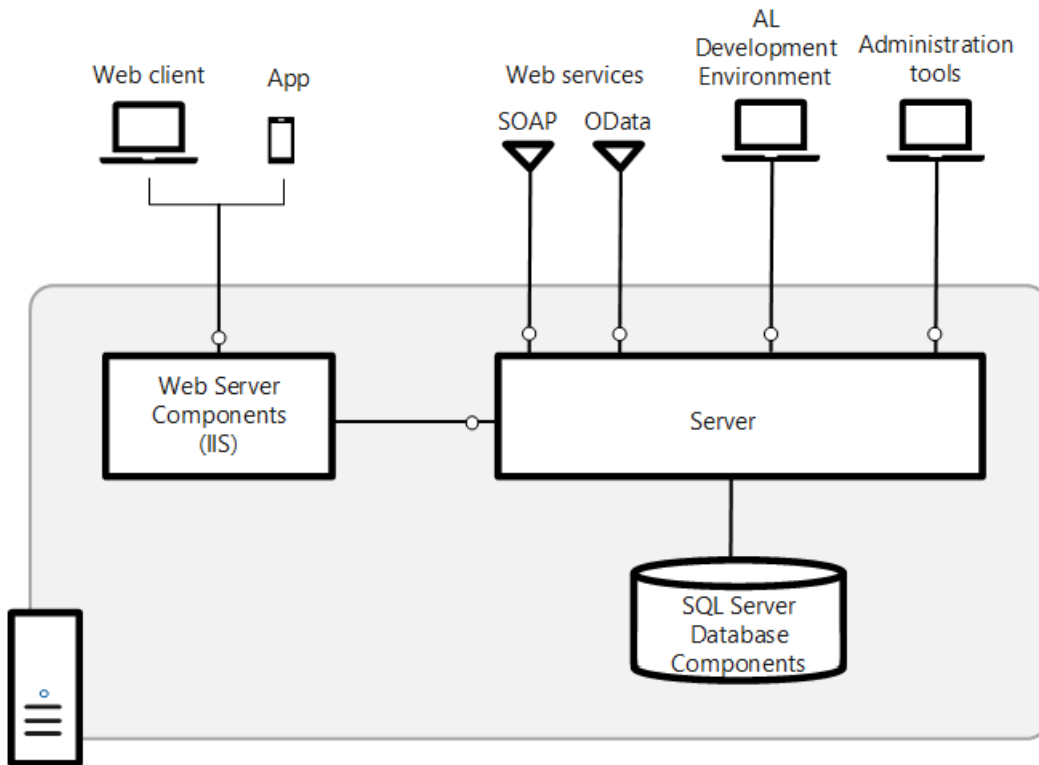
See Also

[Upgrading to Business Central Business Central Components](#)
[System Requirements](#)

Deployment Topologies

3/31/2019 • 2 minutes to read

A Business Central deployment consists of various components that support the production, development, and testing. These components can be installed on various computers. The deployment process varies depending on the topology that you implement. This article provides an overview of the supported topologies.



Deployment Scenarios

TOPOLOGY	DESCRIPTION	MORE INFORMATION
Demonstration	<p>Installs an end-to-end environment, complete with the base application and demonstration data for a single company, on a single computer. The installation enables access to Business Central from the Web client and App, and development.</p> <p>The deployment requires minimal hardware resources, preparation, and configuration.</p>	Deploying a Demonstration Environment
Single-computer	<p>Installs the Business Central Web Server components, Business Central Server, and the SQL Server database components on the same computer.</p>	Deploying in a Single Computer Environment

TOPOLOGY	DESCRIPTION	MORE INFORMATION
Two-computer	Installs the Business Central Web Server components on one computer and the Business Central Server and the SQL Server database components on another computer.	Deploying in a Two Computer Environment
Three-computer	Installs the Business Central Web Server components, Business Central Server, and the SQL Server database components on separate computers.	Deploying in a Three Computer Environment

See Also

[Install Business Central Using Setup](#)
[Business Central Web Server Overview](#)

Deploying the Business Central Demonstration Environment

3/31/2019 • 2 minutes to read

This deployment scenario installs the major Business Central components on a single computer, complete with a base application and database with demonstration data. After the installation, you will have an end-to-end environment, where you can access Business Central data from the Web client. The installation requires minimal hardware resources, preparation, and configuration.

Installed Components and Configuration

Components

This scenario installs the following components:

- Business Central Web Server components
- Internet Information Services

If IIS is already installed, then the setup will enable any required features that are not currently enabled.

- Business Central Server
- SQL Server Database Components, including CRONUS International Ltd. demonstration database and demo license.

For information about what you can do with this license, see [Properties of the Demo License](#).

- Business Central Server Administration tool
- AI Development Environment

Configuration

This scenario uses the default setting of Business Central Setup, which includes the following:

- Business Central Web Server components
 - Port: 8080 (inbound rule automatically added to Windows Firewall)
 - Protocol: HTTP
- Windows authentication for authenticating users.
- Business Central Server configuration:
 - Service instance: BC140
 - Client service port: 7046
 - SOAP web services port: 7047
 - OData web services port: 7048
- Business Central database components configuration:
 - Service instance: NAVDEMO
 - Database: Demo Database NAV (13-0)

- NETWORK SERVICE account is used as the service account for Business Central Server and database.

Prepare for the Business Central Web client installation

1. Get access to the Business Central installation media. For example, this could be a DVD or network drive that contains the Business Central installation files.
2. Make sure that the computer meets the hardware and software requirements.

For more information, see [System Requirements](#).

Run Business Central Setup

1. From the Business Central installation media, run the setup.exe file to start the Business Central Setup.
2. Follow the setup until you get to the **Dynamics 365 Business Central** page, then choose **Advanced installation option > Install Demo**.

The installation starts. This can take several minutes.

Open the Business Central Web client

- To open the Business Central Web client from the computer where you installed Business Central, on the **Start** menu, choose **All Programs**, and then choose **Business Central Web Client**.
- To open the Business Central Web client from other devices on the network, open an Internet browser, and type the following URL in the address box:

```
http://ComputerName:8080/[!INCLUDE[serverinstance](../developer/includes/serverinstance.md)]
```

Substitute **ComputerName** with the name of the computer where you installed Business Central. If you are working on the computer where you installed Business Central, then you can use **localhost**.

For example:

```
http://localhost:8080/BC130
```

NOTE

If you get an error and the Business Central Web client does not open, then see [Troubleshooting Web Server and Web Client Installation](#) in the Dev and IT Pro help for Dynamics NAV to try to resolve the problem.

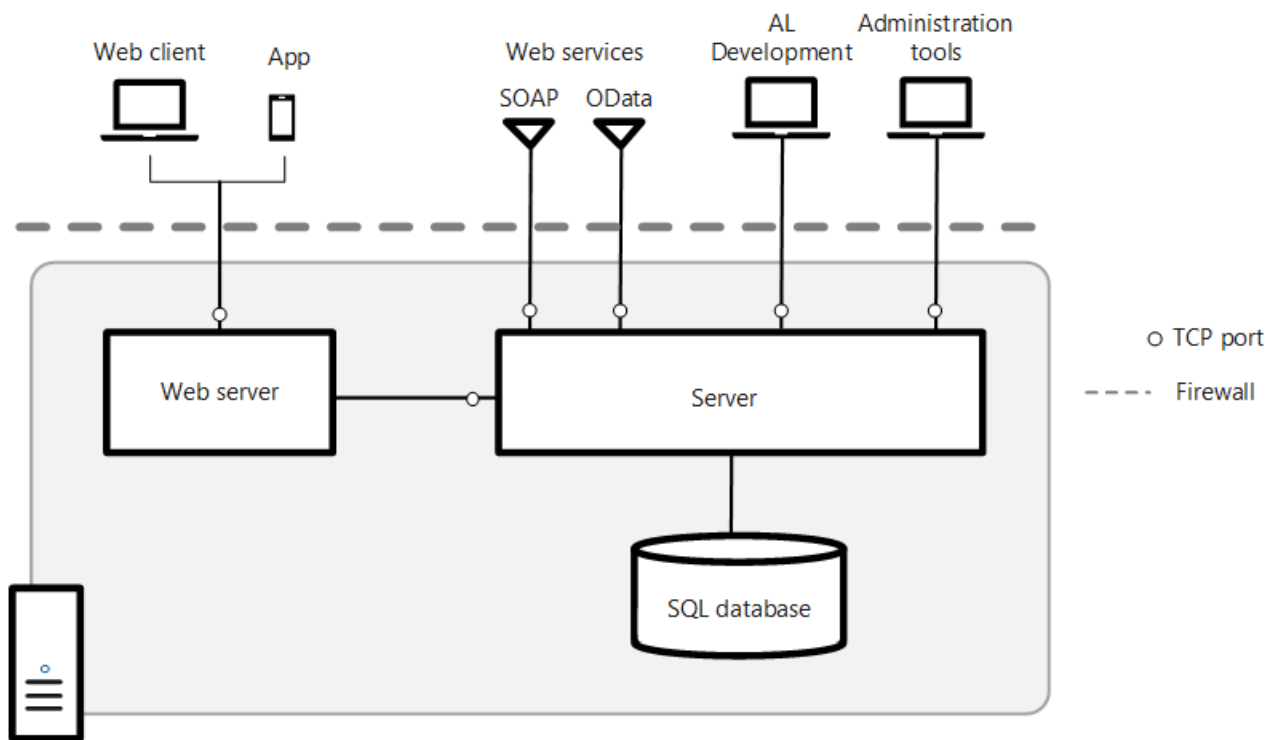
See Also

[Business Central Web Server Overview](#)

Deploying Business Central in a Single-Computer Topology

3/31/2019 • 4 minutes to read

In this scenario, you install the Business Central Web Server components, Business Central Server, and the SQL Server database components on the same computer.



Pre-Installation Tasks

The following table includes tasks to perform before you install.

TASK	DESCRIPTION	FOR MORE INFORMATION, SEE
Make sure that system requirements are met.	Verify that the computer has the required hardware and software installed.	System Requirements
Install Internet Information Services.	<p>When you install the Business Central Web Server components, Business Central Setup creates a website for the Business Central Web client on IIS. If IIS is already installed, then make sure that the required features are enabled.</p> <p>Note: This step is optional because instead of installing and configuring IIS manually, you can use Business Central Setup to install IIS and enable the required features by setting the Install IIS Prerequisites option to Yes.</p>	Configure Internet Information Services

TASK	DESCRIPTION	FOR MORE INFORMATION, SEE
Determine the TCP ports for the Business Central Web client, client services, and SOAP/OData web services (optional) and allow communication on the port through Windows Firewall.	<p>Business Central Setup creates a website on IIS. During Setup, you will have to choose the port to use for the site. The default port is port 8080.</p> <p>The default client services port is 7046.</p> <p>If you will enable SOAP and OData web services, you will also need to specify a port for each. The default ports are 7047 and 7048.</p> <p>If you choose to do so, Business Central Setup will automatically create an inbound rule in Windows Firewall that allows communication on the ports. Otherwise, you will have to do this manually.</p>	Create an Inbound Port Rule in the Windows documentation.
Set up the service account for Business Central Server and the SQL Server database.	Optional. When you install Business Central Server, you can specify a user account that will be used to log on to the Business Central Server instance and Business Central database. The default service account is Network Service. If you want to use Network Service, then no action is required for this task.	Provisioning a Service Account
Obtain and install an SSL certificate.	<p>Optional. If you want to configure SSL on the connection to Business Central Web client, then complete the following procedures:</p> <ul style="list-style-type: none"> - Obtain an SSL certificate. - Import the certificate into the local computer store of the computer on which you will install the Business Central Web Server components. - Obtain the certificate's thumbprint. <p>Note: You can also configure SSL after you have installed the Business Central Web client. For more information, see Post-installation Tasks.</p>	Configure SSL to Secure the Web Client Connection

Installation Tasks

The following table includes tasks for installing the Business Central components.

TASK	DESCRIPTION	FOR MORE INFORMATION, SEE
------	-------------	---------------------------

TASK	DESCRIPTION	FOR MORE INFORMATION, SEE
Install Business Central Web Server components, Business Central Server, and SQL Server database components.	Run the Business Central Setup setup.exe file, choose the Advanced installation options > Choose an installation option > Custom , and then choose the Server , SQL Server Database Components , Server , and Web Server Components options.	Install Business Central Using Setup

Post-installation Tasks

The following table includes tasks that configure the Business Central Web Server components after installation. These tasks are optional depending on your organizational and network requirements.

TASK	DESCRIPTION	FOR MORE INFORMATION, SEE
Change the user authentication method.	The Business Central supports the following authentication methods: Windows, UserName, NavUserPassword, and AccessControlService. By default, Windows authentication is used.	Authentication and User Credential Type
Secure the connection to the Business Central Web client with SSL.	You can help secure Business Central data that is transmitted over the Internet by enabling Secure Sockets Layer (SSL) on the connection to the Business Central Web client.	Configure SSL to Secure the Web Client Connection
Change the configuration of the Business Central Web Server.	There are several parameters in the navsettings.json configuration file for the Business Central Web Server that you can modify to change the behavior of the Business Central Web client. Some of the more common parameters include the Business Central Server instance, company, language, time zone, regional settings, session time out, and online Help URL.	Configuring Business Central Web Server
Set up multiple Business Central Web client applications.	You can set up multiple web server instances for the Business Central Web client on the existing website. The web server instances will use the same address (URL) except with an alias that specifies the specific application.	Creating and Managing Business Central Web Server Instances Using PowerShell
Configure web browsers on devices.	The Business Central Web client supports several different web browsers. To access the Business Central Web client, the web browser must be enabled on a device with cookies and JavaScript.	Web Client Requirements

See Also

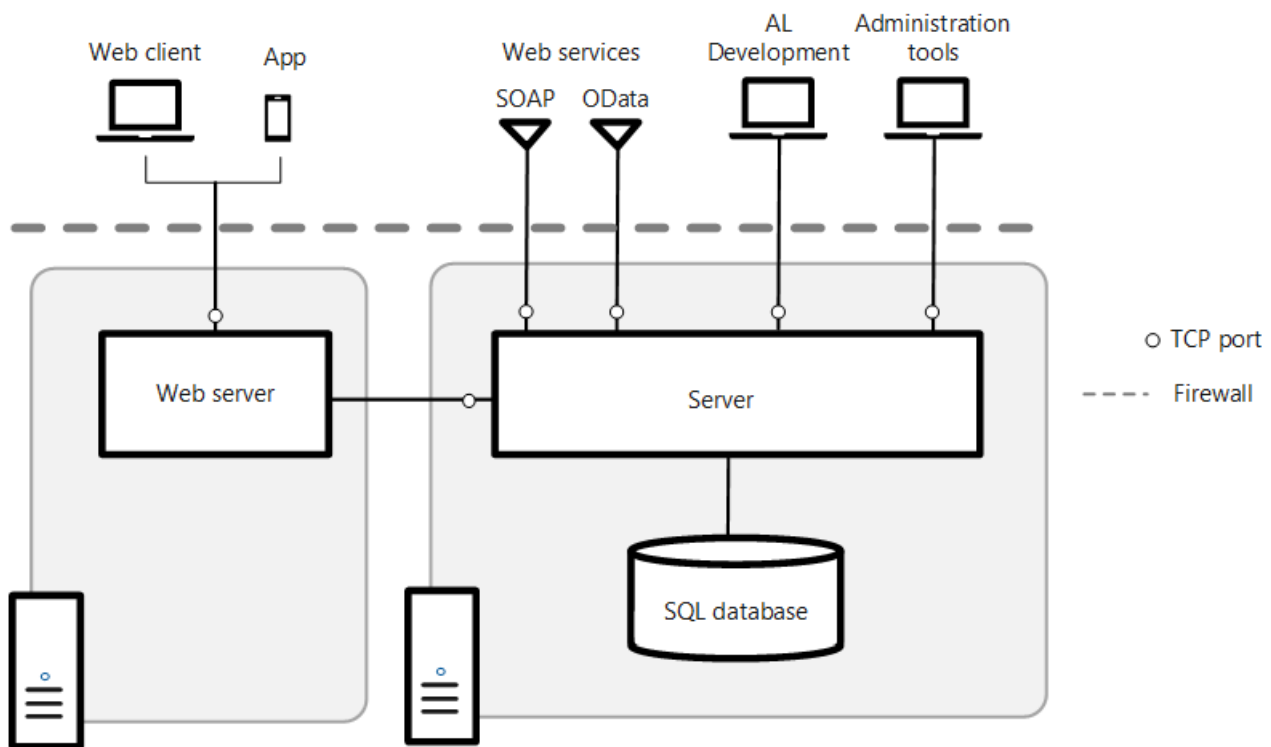
[Business Central Web Server Overview](#)

Installing Business Central in a Two Computer Environment
Installing Business Central in a Three Computer Environment

Deploying Business Central in a Two-Computer Topology

3/31/2019 • 4 minutes to read

In this scenario, you install the Business Central Web Server components on a computer separate than Business Central Server and the SQL Server database components.



Pre-Installation Tasks

The following table includes tasks to perform before you install the Business Central Web Server components.

TASK	DESCRIPTION	FOR MORE INFORMATION, SEE
Make sure that system requirements are met.	Verify that the computer has the required hardware and software installed.	System Requirements
Install Internet Information Services.	<p>When you install the Business Central Web Server components, Business Central Setup creates a website for the Business Central Web client on IIS. If IIS is already installed, then make sure that the required features are enabled.</p> <p>Note: This step is optional because instead of installing and configuring IIS manually, you can use Business Central Setup to install IIS and enable the required features by setting the Install IIS Prerequisites option to Yes.</p>	Configure Internet Information Services

TASK	DESCRIPTION	FOR MORE INFORMATION, SEE
Determine the TCP ports for the Business Central Web client, client services, and SOAP/OData web services (optional) and allow communication on the port through Windows Firewall.	<p>Business Central Setup creates a website on IIS. During Setup, you will have to choose the port to use for the site. The default port is port 8080.</p> <p>The default client services port is 7046.</p> <p>If you will enable SOAP and OData web services, you will also need to specify a port for each. The default ports are 7047 and 7048.</p> <p>If you choose to do so, Business Central Setup will automatically create an inbound rule in Windows Firewall that allows communication on the ports. Otherwise, you will have to do this manually.</p>	Create an Inbound Port Rule in the Windows documentation.
Set up the service account for Business Central Server and the SQL Server database.	Optional. When you install Business Central Server, you can specify a user account that will be used to log on to the Business Central Server instance and Business Central database. The default service account is Network Service. If you want to use Network Service, then no action is required for this task.	Provisioning a Service Account
Obtain and install an SSL certificate.	<p>Optional. If you want to configure SSL on the connection to Business Central Web client, then complete the following procedures:</p> <ul style="list-style-type: none"> - Obtain an SSL certificate. - Import the certificate into the local computer store of the computer on which you will install the Business Central Web Server components. - Obtain the certificate's thumbprint. <p>Note: You can also configure SSL after you have installed the Business Central Web client. For more information, see Post-installation Tasks.</p>	Configure SSL to Secure the Web Client Connection

Installation Tasks

The following table includes tasks for installing the Business Central Web Server components.

TASK	DESCRIPTION	FOR MORE INFORMATION, SEE
------	-------------	---------------------------

TASK	DESCRIPTION	FOR MORE INFORMATION, SEE
On one computer, install Business Central Server and SQL Server Database Components on one computer	Run the Business Central Setup setup.exe file, choose Advanced installation options > Choose an installation option > Custom , and then choose the Server and SQL Server Database Components options.	Install Business Central Using Setup
On the other computer, install the Business Central Web Server components.	Run Business Central Setup, choose Advanced installation options > Choose an installation option > Custom , and then the Web Server Components option.	Install Business Central Using Setup Business Central Web Server Overview
Configure delegation from the web server to Business Central Server.	Because Business Central Server is running on a different computer than the Business Central Web Server components, you must configure the computer that is running Business Central Web Server components to delegate its access to Business Central Server on behalf of the device trying to access from the Business Central Web client.	Configure Delegation for Business Central Web Server

Post-installation Tasks

The following table includes tasks to configure the Business Central Web Server components after installation. These tasks are optional depending on your organizational and network requirements.

TASK	DESCRIPTION	FOR MORE INFORMATION, SEE
Change the user authentication method.	The Business Central supports the following authentication methods: Windows, UserName, NavUserPassword, and AccessControlService. By default, Windows authentication is used.	Authentication and User Credential Type
Secure the connection to the Business Central Web client with SSL.	You can help secure Business Central data that is transmitted over the Internet by enabling Secure Sockets Layer (SSL) on the connection to the Business Central Web client.	Configure SSL to Secure the Web Client Connection
Change the configuration of the Business Central Web Server.	There are several parameters in the navsettings.json configuration file for the Business Central Web Server that you can modify to change the behavior of the Business Central Web client. Some of the more common parameters include the Business Central Server instance, company, language, time zone, regional settings, session time out, and online Help URL.	Configuring Business Central Web Server

TASK	DESCRIPTION	FOR MORE INFORMATION, SEE
Set up multiple Business Central Web client applications on a website.	You can set up multiple web server instances for the Business Central Web client on the existing website. The web server instances will use the same address (URL) except with an alias that specifies the specific application.	Creating and Managing Business Central Web Server Instances Using PowerShell
Configure web browsers on devices.	The Business Central Web client supports several different web browsers. To access the Business Central Web client, the web browser must be enabled on a device with cookies and JavaScript.	Web Client Requirements

See Also

[Business Central Web Server Overview](#)

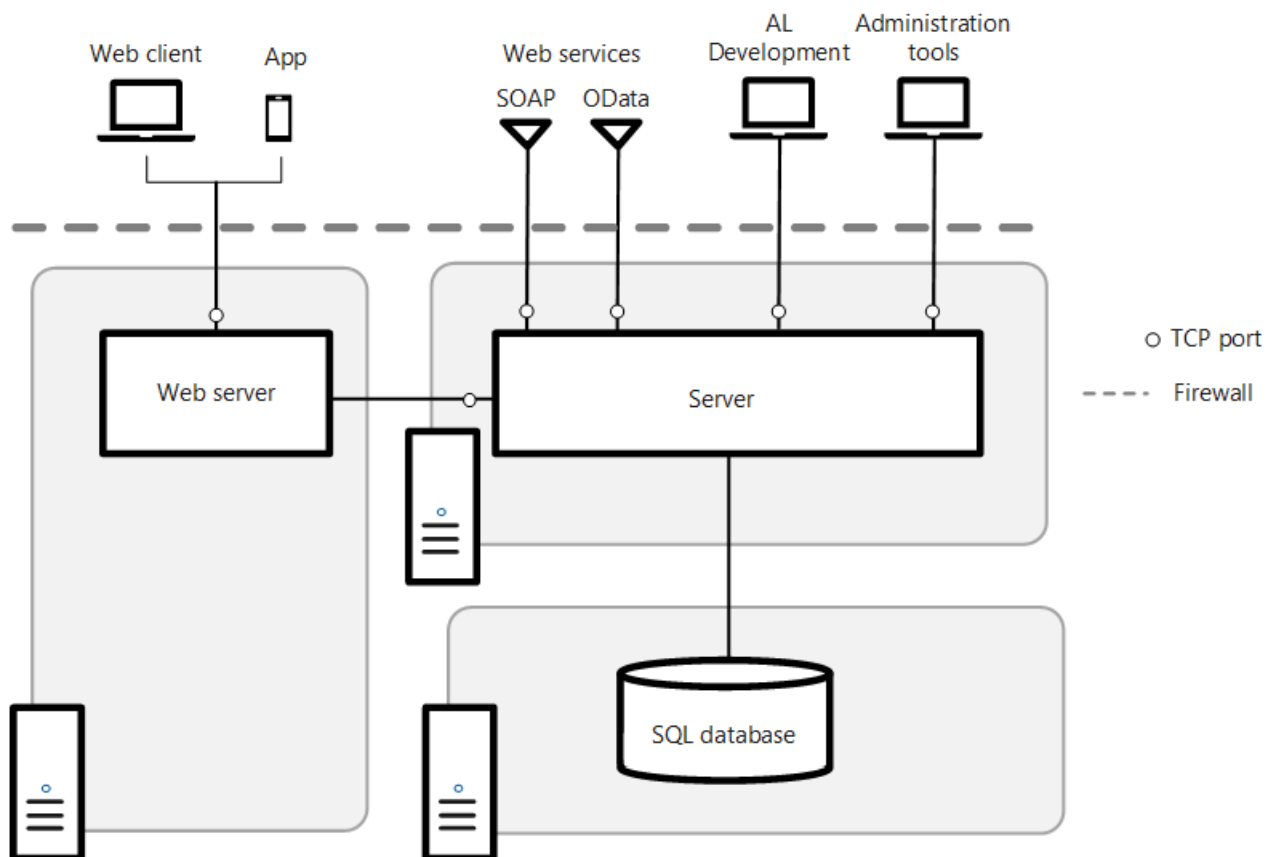
[Installing Business Central in a Single Computer Environment](#)

[Installing Business Central in a Three Computer Environment](#)

Deploying Business Central in a Three-Computer Topology

3/31/2019 • 5 minutes to read

In this scenario, you install the Business Central Web Server components, Business Central Server, and the SQL Server database components on separate computers.



This article also applies to deploying the Business Central Phone client and Business Central Tablet client.

Pre-Installation Tasks

The following table includes tasks to perform before you install the Business Central Web Server components.

TASK	DESCRIPTION	FOR MORE INFORMATION, SEE
Make sure that system requirements are met.	Verify that the computer has the required hardware and software installed.	System Requirements

TASK	DESCRIPTION	FOR MORE INFORMATION, SEE
Install Internet Information Services.	<p>When you install the Business Central Web Server components, Business Central Setup creates a website for the Business Central Web client on IIS. If IIS is already installed, then make sure that the required features are enabled.</p> <p>Note: This step is optional because instead of installing and configuring IIS manually, you can use Business Central Setup to install IIS and enable the required features by setting the Install IIS Prerequisites option to Yes.</p>	Configure Internet Information Services
Determine the TCP ports for the Business Central Web client, client services, and SOAP/OData web services (optional) and allow communication on the port through Windows Firewall.	<p>Business Central Setup creates a website on IIS. During Setup, you will have to choose the port to use for the site. The default port is port 8080.</p> <p>The default client services port is 7046.</p> <p>If you will enable SOAP and OData web services, you will also need to specify a port for each. The default ports are 7047 and 7048.</p> <p>If you choose to do so, Business Central Setup will automatically create an inbound rule in Windows Firewall that allows communication on the ports. Otherwise, you will have to do this manually.</p>	Create an Inbound Port Rule in the Windows documentation.
Set up the service account for Business Central Server and the SQL Server database.	Optional. When you install Business Central Server, you can specify a user account that will be used to log on to the Business Central Server instance and Business Central database. The default service account is Network Service. If you want to use Network Service, then no action is required for this task.	Provisioning a Service Account
Obtain and install an SSL certificate.	<p>Optional. If you want to configure SSL on the connection to Business Central Web client, then complete the following procedures:</p> <ul style="list-style-type: none"> - Obtain an SSL certificate. - Import the certificate into the local computer store of the computer on which you will install the Business Central Web Server components. - Obtain the certificate's thumbprint. <p>Note: You can also configure SSL after you have installed the Business Central Web client. For more information, see Post-installation Tasks.</p>	Configure SSL to Secure the Web Client Connection

Installation Tasks

The following table includes tasks for installing the Business Central Web client.

TASK	DESCRIPTION	FOR MORE INFORMATION, SEE
On the first computer, install the Business Central database components.	Run the Business Central Setup setup.exe file, choose Advanced installation options > Choose an installation option > Custom , and then choose the SQL Server Database Components option.	Install Business Central Using Setup
Start the SQL Server Browser Service on the SQL Server computer.	This task is only required if you are using a named database instance for Business Central. By default, Business Central uses the database instance named NAVDEMO. The SQL Server Browser Service is required so that the database instance can be discovered by the Business Central Server instance, which in this scenario, is another computer. To start the SQL Server, use SQL Server Configuration Manager.	Start, Stop, Pause, Resume, Restart SQL Server Services
On second computer, install Business Central Server.	Run the Business Central Setup setup.exe file, choose Advanced installation options > Choose an installation option > Custom , and then choose the Server option.	Install Business Central Using Setup
On the third computer, install the Business Central Web Server components.	Run the Business Central Setup setup.exe file, choose Advanced installation options > Choose an installation option > Custom , and then choose Web Server Components option.	Install Business Central Using Setup
Configure delegation from the web server to Business Central Server.	Because Business Central Server is running on a different computer than the Business Central Web Server components, you must configure computer running Business Central Web Server components to delegate its access to Business Central Server on behalf of the device trying to access from the Business Central Web client.	Configure Delegation for Business Central Web Server

Post-installation Tasks

The following table includes tasks that configure the Business Central Web Server components after installation. These tasks are optional depending on your organizational and network requirements.

TASK	DESCRIPTION	FOR MORE INFORMATION, SEE
------	-------------	---------------------------

TASK	DESCRIPTION	FOR MORE INFORMATION, SEE
Change the user authentication method.	The Business Central supports the following authentication methods: Windows, UserName, NavUserPassword, and AccessControlService. By default, Windows authentication is used.	Authentication and User Credential Type
Secure the connection to the Business Central Web client with SSL.	You can help secure Business Central data that is transmitted over the Internet by enabling Secure Sockets Layer (SSL) on the connection to the Business Central Web client.	Configure SSL to Secure the Web Client Connection
Change the configuration of the Business Central Web Server.	There are several parameters in the navsettings.json configuration file for the Business Central Web Server that you can modify to change the behavior of the Business Central Web client. Some of the more common parameters include the Business Central Server instance, company, language, time zone, regional settings, session time out, and online Help URL.	Configuring Business Central Web Server
Set up multiple Business Central Web client applications on a website.	You can set up multiple web server instances for the Business Central Web client on the existing website. The web server instances will use the same address (URL) except with an alias that specifies the specific application.	Creating and Managing Business Central Web Server Instances Using PowerShell
Configure web browsers on devices.	The Business Central Web client supports several different web browsers. To access the Business Central Web client, the web browser must be enabled on a device with cookies and JavaScript.	Web Client Requirements

See Also

[Business Central Web Server Overview](#)

[Installing Business Central in a Single Computer Environment](#)

[Installing Business Central in a Two Computer Environment](#)

Installing Business Central Using Setup

3/31/2019 • 5 minutes to read

You use Business Central Setup to install the different components that comprise a Business Central production, demonstration, or development environment. For a list of components, see [Components and Topology](#).

About Setup

Setup is available on the installation media (DVD) in the setup.exe file. When run, Setup leads you through installation process, where you can install individual components or select predefined options that install a logical collection of components.

Configuration settings

Throughout Setup, you are presented with various configuration settings. Some settings require that you set them, and other settings have a default value. In many cases, the default value is sufficient for the initial installation. After you run Setup, you can change the configuration settings by using other tools such as the Business Central Server Administration tool and Business Central Administration Shell.

Prerequisite Installations by Setup

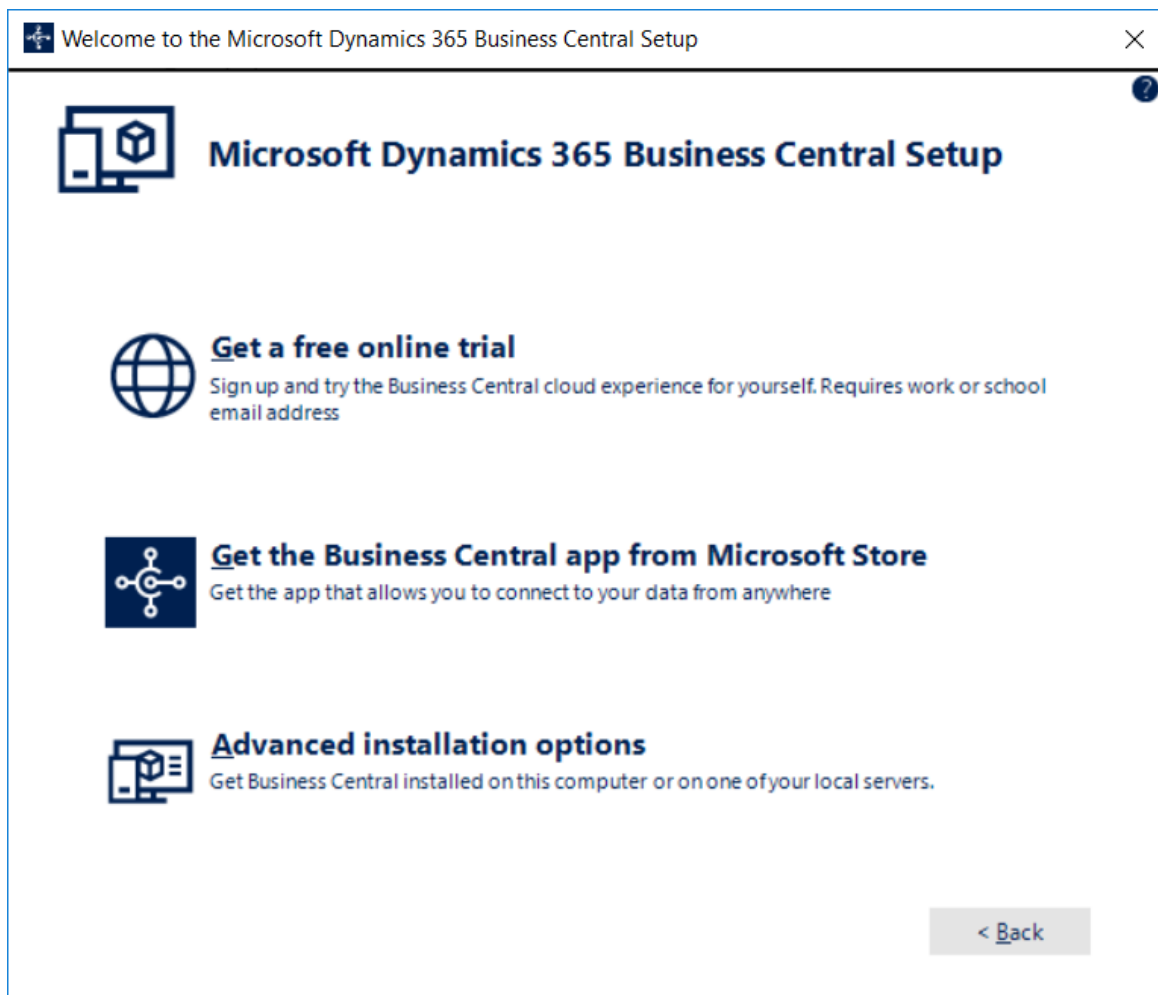
There are some components that require other software in order run, for example the database requires SQL Server and the Web client requires IIS. Setup will install several of these prerequisites, like installing SQL Server Express and enabling IIS. You can see which prerequisites Setup installs in the [System Requirements](#).

Before you run Setup

1. Plan you deployment and identify the components that you want to install.
2. Verify that the target computer where you will install components meets the hardware and software requirements for the components that you want to install. For more information, see [System Requirements](#).
3. Make sure that you are an administrator on the computer where you run Setup.

Run Setup

1. In the installation media (DVD) folder, double-click the setup.exe.
2. Follow Setup until you get to the **Dynamics 365 Business Central** page.



- Choose **Get a free online trial to sign-up** if you interested in hearing about and trying the cloud experience.
- Choose **Get the Business Central app from the Microsoft Store** to download a companion app that mimics that Web client but has the same look-and-feel as the mobile apps.
- Choose **Advance installation options** to install a demonstration environment or one more components. Then, follow the on-screen instructions to complete the installation.

Cancel Setup

Setup does not provide a **Cancel** button on all pages, but you can cancel an installation from any page by choosing the **Close** button in the upper-right corner. All Business Central components are removed from the computer. The only software that Setup cannot remove are:

- Database files, such as the Demo database.
- Prerequisites for Business Central components that Setup can install, such as the .NET Framework.

Run Setup from a command prompt

You can run Business Central Setup from a command prompt, either by pointing to its location on the installation media, or after the initial installation, from the location where the Setup.exe is automatically stored on your computer. The default location is:

```
C:\Program Files (x86)\Common Files\Microsoft Dynamics 365 Business Central\<Version number>\Setup
```

You can use the following options with Setup.exe.

SETUP OPTION	DESCRIPTION
/config <Setup config file>	Specifies path and file name information for a Setup configuration file to load. This is the only required option.
/help	Displays Help about Setup.exe options.
/log <log path>	Specifies path and file name information for a Setup log file to be created by Setup. The file must not exist before you run Setup.
/quiet	Specifies that Setup does not display anything on the screen. All configuration information is taken from the specified configuration file.
/repair	Repairs the current installation of Business Central.
/uninstall	Removes the current installation Business Central.

Save, Edit, and Load a Setup Configuration File

During Setup, you can save the configuration settings to a file before you finish and exit Setup. Then later, you can load use Setup to load the configuration file to make it faster to replicate the same configuration for another deployment.

Save to a Setup configuration file

1. Choose **Save** on the **Specify parameters** page in Setup. This page is available when you run Setup unless you select **Install Demo**, which skips all other Setup pages.
2. Type a file name for the configuration file. An .xml extension is added automatically.
3. Choose **Save**.

You now return to the **Specify parameters** page, where you can continue with installing software. You can also close Setup if you only have to create a Setup configuration file.

Edit a Setup configuration file

You edit the file using an XML editor or text editor. Setup configuration files contain two types of settings.

SETTING TYPE	PURPOSE
Component	<p>For each component, there are three separate values, all displayed on a single line:</p> <ul style="list-style-type: none"> - ShowOptionNode Specifies whether the component should be displayed in Setup. For silent installs, this parameter is not relevant. - State There are two possible values: Local, indicates that the component is included in the install, and Absent indicates that the component is not included. - Id Identifies the component <p>You can change value for State or ShowOptionNode, but not for Id. Also, you cannot add or remove a component.</p>

SETTING TYPE	PURPOSE
Parameter	These settings contain configuration information for components. As with Components, you can modify a parameter's Value , but not its Id .

Load a Setup configuration file

The option to load a Setup configuration file is on the **Choose an installation option** page in Setup.

IMPORTANT

A Setup configuration file contains information about which components to install and which settings to apply to each component. Therefore, you should not customize the list of components or configure components in Setup before you load a Setup configuration file because loading the configuration overwrites all prior customization and configuration.

1. On the **Choose an installation option** page, choose **Load Configuration**.

This option is located under **Custom Components**.

2. In the **Open** dialog box, select or browse to the Setup configuration file that you want to open, and then double-click the file.

Setup now shows the **Customize the installation** page that has been modified according to the component selection in the loaded Setup configuration file.

3. Modify the list of components to install or choose **Next** to proceed to the **Specify parameters** page, where settings from the Setup configuration file are shown.
4. Configure these settings or choose **Apply** to accept these values and continue.

See Also

[Components](#)

[Deployment](#)

Provisioning the Business Central Server Service Account

3/31/2019 • 9 minutes to read

The Business Central Server account is used by Business Central clients to log on to the Business Central Server instance. The Business Central Server then uses the service account to log on to the Business Central database. When you install Business Central Server, you identify an Active Directory account to provide credentials for the server. By default, Setup runs Business Central Server under the Network Service account, a predefined local account used by the service control manager. This account has minimum privileges on the local computer and acts as the computer on the network.

We recommend that you create a domain user account for running Business Central Server. The Network Service account is considered less secure because it is a shared account that can be used by other unrelated network services. Any users who have rights to this account have rights to all services that are running on this account. If you create a domain user account to run Business Central Server, you can use the same account to run SQL Server, whether or not SQL Server is on the same computer.

NOTE

Because Business Central Setup and the New-NavDatabase cmdlet configure the required permissions for the Business Central Server account, you will typically use the procedures in this topic when you change the Business Central Server account for an existing installation.

Prerequisite

Delete the **Dynamics 365 Business Central** folder in the **ProgramData** folder of your system drive, for example, `C:\ProgramData\Microsoft\Microsoft Dynamics NAV\` or

`C:\ProgramData\Microsoft\Microsoft Dynamics 365 Business Central`.

The **ProgramData** folder is typically hidden, so you might have to change the folder options for your system drive to show hidden files, folders, and drives.

Provisioning a Domain User Account

If you are running the Business Central Server under a domain user account, you must:

- Enable the account to log in as a service
- Enable the account to register an SPN on itself
- Add the account to the SMSvcHost.exe.config file
- Give the account necessary database privileges in SQL Server

Enabling the account to log in as a service

Depending on various factors, the account may already have this ability to log in as a service. For example, if you have already installed SQL Server and configured it to run under the same account, SQL Server will have modified the account to log in as a service.

When this permission is lacking, Business Central Server instances may not be able to start.

For instructions for enabling an account to log in as a service, see [Manage User Accounts in Windows Server](#).

Enabling the account to register an SPN on itself

To enable secure mutual authentication between clients and Business Central Server, you must configure the Business Central Server account to self-register Service Principal Names (SPNs). Mutual authentication is recommended in a production environment but may not be necessary in a testing or staging environment. This is done by modifying the account in Active Directory.

For more information, see [Service Principal Names](#) in the Active Directory documentation.

Add the account to the SMSvcHost.exe.config file

Business Central uses Net.TCP Port Sharing Service, which is managed by SMSvcHost.exe. The SMSvcHost.exe.config contains information about the identities (or accounts) that can use the service. These accounts are specified as security identifiers (SIDs) in the section of the SMSvcHost.exe.config file. By default, permission is implicitly granted to system accounts, such as NetworkService. For other accounts, you must explicitly add the SID for the account to the SMSvcHost.exe.config file as follows:

1. Get the SID of the user account.

The SID is an alphanumeric character string, such as S-1-5-20 or S-1-5-32-544. There are different ways to get the SID, such using Windows Management Instrumentation Control Command-line (WMIC) or the computer's registry.

- To use WMIC, open a command prompt, and run the following command:

```
wmic useraccount get name,sid
```

This will display a list of user accounts and their SIDs.

- To use the registry, run regedit, and then go to the *HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList* folder. This folder list the SIDs for each user account. To find the SID that corresponds to the user account that you want, look at the *ProfileImagePath* key data.

2. Using a text editor, open the SMSvcHost.exe.config file.

You will find the SMSvcHost.exe.config file in the installation folder for the latest .NET Framework version on the Business Central Server computer; for example, `C:\Windows\Microsoft.NET\Framework64\v4.0.30319\`.

3. Add the SID to the element as follows, and then save the file:

```
<system.serviceModel.activation>
  <net.tcp listenBacklog="10" maxPendingConnections="100" maxPendingAccepts="2"
receiveTimeout="00:00:10" teredoEnabled="false">
    <allowAccounts>
      <!-- Your NAV Server account -->
      <add securityIdentifier="N-N-N-N"/>
    </allowAccounts>
  </net.tcp>
```

For more information about SMSvcHost.exe and the SMSvcHost.exe.config file, see [Configuring the Net.TCP Port Sharing Service](#).

Giving the account necessary database privileges in SQL Server

The Business Central Server account needs two privileges on SQL Server instance used for Business Central:

1. To create databases on the instance, it must have the dbcreator server-level role. This privilege is only

needed during database creation.

2. To serve client requests and read/write to the Business Central database, it must be member of the db_owner database role on the Business Central database.

When you install the Business Central database by using Business Central Setup or the [New-NAVDatabase](#) cmdlet, you can specify the Business Central Server account. In these cases, the server account that you specify should already have the necessary privileges in SQL Server. If you change the Business Central Server account for an existing installation, then you should verify the account has the required privileges in SQL Server.

To verify server-level and database-level privileges on SQL Server after you create your Business Central database, use SQL Server Management Studio and, if necessary, modify privileges. If you have installed SQL Server with the guidelines in [Installation Considerations for Microsoft SQL Server](#), then SQL Server Management Studio is already installed on your computer. Otherwise, update your SQL Server installation to include the **Management Tools - Complete option for SQL Server** (for SQL Server 2012/2014.) For SQL Server 2016, SQL Server Management Studio can be downloaded and installed as a standalone application.

Assign necessary SQL Server privileges for the Business Central Server account

1. Start SQL Server Management Studio and connect to the instance where the Business Central database is installed.
2. Create a login for the Business Central Server account.
 - a. Navigate the tree view: **Security, Logins**.
 - b. Right-click **Logins** and choose **New Login**.
 - c. Choose **Search**, and use the **Select User or Group** dialog box to identify the Business Central Server account.
 - d. Choose **OK** to exit the New Login dialog box.
3. (optional) Grant the login **Alter any event session** and **View server state** permissions.

This step is only required if you want to log SQL Server deadlocks in the Windows Event log for the Business Central Server instance. For more information, see [Monitoring SQL Database Deadlocks](#).

- a. Navigate the tree view: **Security, Logins**.
 - b. Right-click the login that you created, and then choose **Properties**.
 - c. Under **Select a page**, choose **Securables**.
 - d. On the **Explicit** tab, select the **Alter any event session** and **View server state** check boxes in the **Grant** column.
 - e. Choose **OK**.
4. Grant the login the server-level role **dbcreator**
 - a. Navigate the tree view: **Security, Logins**.
 - b. Right-click the Business Central Server account, and then choose **Properties**.
 - c. Click on **Server Roles**.
 - d. Check the **dbcreator** box.
 - e. Choose **OK**.
 5. Add the login as a user on the master database.
 - a. Navigate the tree view: **Databases, System Databases, master, Security, Users**.
 - b. Right-click **Users** and choose **New User**.
 - c. Choose the ellipse button at the far right of the second line in the **Database User – New** dialog box.

- d. In the **Select Login** dialog box, enter or browse for the login you created for the Business Central Server account.
 - e. Enter a name in the **User name** field (the first line in the **Database User - New** dialog box).
 - f. Choose **OK** to exit the **Database User - New** dialog box.
6. Grant the Business Central Server login permissions on the master database. In the tree view, right-click **master** and choose **Properties**. Then do the following in the **Database Properties – master** dialog box.
 - a. Under **Select a Page**, choose **Permissions**.
 - b. Under **Name**, choose the login you created for the Business Central Server account name.
 - c. Under **Permissions for <username>**, on the **Explicit** tab, scroll down to down to the **Select** line, and select the check box in the **Grant** column.
 - d. Choose **OK** to exit the **Database Properties – master** dialog box.
 - e. Navigate the tree view: **Databases, System Databases, master, Tables, System Tables**.
 - f. Right-click the **dbo.\$ndo\$srvproperty** table and choose **Properties**.
 - g. Under **Select a Page**, choose **Permissions**.
 - h. Choose **Search**, and use the **Select User or Group** dialog box to identify the login for the Business Central Server account.
 - i. Under **Permissions for <username>**, on the **Explicit** tab, scroll down to down to the **Select** line, and select the check box in the **Grant** column.
 - j. Choose **OK** to exit the **Table Properties – dbo.\$ndo\$srvproperty** dialog box.
 7. Grant the login the necessary database roles on the Business Central database.
 - a. Navigate the tree view: **Databases, <your Microsoft Dynamics NAV database>, Security, Users**.
 - b. Right-click **Users** and choose **New User**.
 - c. In the **Database User – New** dialog box, choose the ellipse button at the far right of the second line.
 - d. Select the login you created for the Business Central Server account name and choose **OK**.
 - e. Under **Database role membership**, select the **db_owner** check box.
 - f. Choose **OK** to exit the **Database User – New** dialog box.
 - g. Right-click your Business Central database and choose **Properties**.
 - h. Under **Select a Page**, choose **Permissions**.
 - i. Choose **Search**, and use the **Select User or Group** dialog box to identify login you created for the Business Central Server account.
 - j. Under **Permissions for <username>**, on the **Explicit** tab, scroll down to down to the **View database state** line, and select the check box in the **Grant** column.
 - k. Choose **OK** to exit the Database Properties dialog box for your Business Central database.

Alternatively, you can script these steps in SQL Server Management Studio, as shown in the following example:

```
USE [master]
GO
CREATE LOGIN [domain\accountname] FROM WINDOWS
CREATE USER [domain\accountname] FOR LOGIN [domain\accountname]
GRANT SELECT ON [master].[dbo].[$ndo$srvproperty] TO [domain\accountname]
ALTER SERVER ROLE [dbcreator] ADD MEMBER [domain\accountname]
GRANT VIEW SERVER STATE TO [domain\accountname]
GRANT ALTER ANY EVENT SESSION TO [domain\accountname]

GO
USE [Microsoft Dynamics NAV Database]
GO
CREATE USER [domain\accountname] FOR LOGIN [domain\accountname]
ALTER ROLE [db_owner] ADD MEMBER [domain\accountname]
GRANT VIEW DATABASE STATE TO [domain\accountname]
```

Provisioning the Network Service Account

The only circumstance where it is necessary to take any action regarding the Network Service account is when change the Business Central Server account on an existing installation from a domain account to the Network Service. In this situation you must verify that the account has the necessary database privileges in SQL Server, as per [Giving the account necessary database privileges in SQL Server](#), above.

Using Security Certificates with Business Central On-Premises

3/31/2019 • 8 minutes to read

You use certificates to help secure connections over a wide area network (WAN), such as connections from the Business Central Web Server, Dynamics NAV Client connected to Business Central, and web services to the Business Central Server. Implementing security certificates on your deployment environment requires modifications to various components, like the Business Central Server, Business Central Web Server, and clients.

About Security Certificates

A certificate is a file that Business Central Server uses to prove its identity and establish a trusted connection with the client that is trying to connect. Business Central can support the following configurations:

- *Chain trust*, which specifies that each certificate must belong to a hierarchy of certificates that ends in a root authority at the top of the chain.
- *Peer trust*, which specifies that both self-issued certificates and certificates in a trusted chain are accepted.

The implementation in this section describes the chain trust configuration, which is the more secure option.

NOTE

An instance of Business Central Server that has been configured for secure WAN communication always prompts users for authentication when they start the client, even when the client computer is in the same domain as Business Central Server.

Certificates for Production

In a production environment, you should obtain a certificate from a certification authority or trusted provider. Some large organizations may have their own certification authorities, and other organizations can request a certificate from a third-party organization.

Obtaining Certificates

You implement chain trust by obtaining X.509 service certificates from a trusted provider. These certificates and their root certification authority (CA) certificates must be installed in the certificates store on the computer that is running Dynamics NAV Server. The CA certificate must also be installed in the certificate store on computers that are running the Business Central Web Server and Dynamics NAV Client connected to Business Central so that clients can validate the server.

Most enterprises and hosting providers have their own infrastructure for issuing and managing certificates. You can also use these certificate infrastructures. The only requirement is that the service certificates must be set up for key exchange and therefore must contain both private and public keys. Additionally, the service certificates that are installed on Business Central Server instances must have the Service Authentication and Client Authentication certificate purposes enabled.

IMPORTANT

Microsoft recommends against using wildcard SSL certificates in Business Central installations. Wildcard certificates pose security risks because if one server or sub-domain is compromised, all sub-domains may be compromised. Wildcard certificates also introduce a new style of impersonation attack. In this attack, the victim is lured to a fraudulent resource in the certified domain through phishing. Conventional certificates detect this attack, because the user's browser checks that the private key is hosted on a server whose name matches the one displayed in the browser's address window.

Run the Certificates Snap-in for Microsoft Management Console

Some of the following procedures use the Certificates snap-in for Microsoft Management Console (MMC). If you do not already have this snap-in installed, you can add it to the MMC. For information see [Add the Certificates Snap-in to an MMC](#).

Install and Configure the Certificates

You install the security certificates on the computers running Business Central Server, Business Central Web Server, and Dynamics NAV Client connected to Business Central. The root CA certificate and the service certificate are used in the configuration, but client certificates are not.

Install Certificates on components

1. Follow the installation instructions that are available from your certificate provider to install the root CA and service certificates on the following computers:
 - Install the root CA on the computer that is running Business Central Server and all computers that are running Business Central Web Server instances and Dynamics NAV Client connected to Business Central.
 - Install the service certificate on the computer that is running Business Central Server only.
2. Make sure that the **Server Authentication** and **Client Authentication** certificate purposes are enabled for the service certificate.

A certificate can be enabled for several different purposes. The **Server Authentication** and **Client Authentication** purposes must be enabled. You can enable or disable other purposes to suit your requirements.

You enable certificate purposes by using the Certificates Snap-in for MMC. For more information, see [Modify the Properties of a Certificate](#).

Grant access to the Business Central Server service account

After you have installed the root CA and the service certificate on the computer running Business Central Server, you must grant access to the service account that is associated with the server so that the service account can access the service certificate's private key.

1. In the left pane of MMC, expand the **Certificates (Local Computer)** node, expand the **Personal** node, and then select the **Certificates** subfolder.
2. In the right pane, right-click the certificate, select **All Tasks**, and then choose **Manage Private Keys**.
3. In the **Permissions** dialog box for the certificate, choose **Add**.
4. In the **Select Users, Computers, Service Accounts, or Groups** dialog box, enter the name of the dedicated domain user account that is associated with Business Central Server, and then choose the **OK** button.
5. In the **Full Control** field, select **Allow**, and then choose the **OK** button.

6. In the right pane, select the certificate.
7. In the **Certificate** dialog box, choose the **Details** tab, and then select the **Thumbprint** field.
8. Copy the value of **Thumbprint** field.

For example, copy the hexadecimal characters to text editor, such as Notepad. Delete all spaces from the thumbprint string. If the thumbprint is `c0 d0 f2 70 95 b0 3d 43 17 e2 19 84 10 24 32 8c ef 24 87 79`, then change it to `c0d0f27095b03d4317e219841024328cef248779`.

TIP

It is important that the thumbprint does not contain any invisible extra characters; otherwise you will experience problems when using it later. To avoid this, see [Certificate thumbprint displayed in MMC certificate snap-in has extra invisible unicode character](#).

Configure the Business Central Server instance

The Business Central Server instance configuration includes several settings for certificates and enabling remote logins. You can modify a server instance by using Business Central Server Administration tool or Business Central Administration Shell. For details about how to modify a server instance, see [Configuring Business Central Server](#).

1. Run the Business Central Server Administration tool.
2. Under **General**, change the following settings for the Business Central Server instance.

SETTING	NEW VALUE	DESCRIPTION
Credential Type	<code>NavUserPassword</code> , <code>Username</code> , or <code>AccessControlService</code>	The default value is <code>Windows</code> . When you change it to <code>NavUserPassword</code> , <code>Username</code> , or <code>AccessControlService</code> , client users who connect to the server are prompted for user name and password credentials.
Certificate Thumbprint	Value of the Thumbprint field in the previous procedure.	Remove any leading or trailing spaces in the thumbprint.

3. If you want to use secure web services, then under **SOAP Services** and **OData Services**, select the **Enable SSL** check box.
4. Save and the new values for the server instance.
5. Restart the Business Central Server instance.

If there is a problem, see Windows Event Viewer.

Configure the Business Central Web Server and Dynamics NAV Client connected to Business Central

The chain trust configuration allows client users to log on to one or more instances of Business Central Server as long as their login credentials have been associated with user accounts in Business Central. The client validates that the server certificate is signed with the root CA.

After you have installed the root CA on the computer running the Business Central Web Server or Dynamics NAV Client connected to Business Central, you must modify the client configuration file.

Modify the Business Central Web client configuration file

1. On the computer that is installed the Business Central Web Server, open the [navsetting.json configuration file](#) in a text editor, such as Notepad.
2. Change the following settings:

KEY	NEW VALUE	DESCRIPTION
ClientServicesCredentialType	NavUserPassword , Username , or AccessControlService	The default value is Windows . When you change it to NavUserPassword , Username , or AccessControlService , client users who connect to the server are prompted for user name and password credentials.
DnsIdentity	The subject name of the service certificate	The default value is <identity> . Replace this with the subject name or common name (CN) of the certificate that is used on the computer that is running Business Central Server.

3. Save the navsettings.json configuration file.

Modify the Dynamics NAV Client connected to Business Central configuration file

1. Open the ClientUserSettings.config configuration file.

The location of this file is *Users\<username>\AppData\RoamingLocal\Microsoft\Dynamics 365 Business Central*.

By default, this file is hidden. Therefore, you may have to change your folder options in Windows Explorer to view hidden files.

NOTE

If you want to change default Dynamics NAV Client connected to Business Central settings for all future users, edit the default ClientUserSettings.config file — that is, the one in C:\Program Files (x86)\Microsoft Dynamics 365 Business Central\140. Be sure that you run your text editor with Administrator privileges when you do so.

2. Modify the following settings.

KEY	NEW VALUE	DESCRIPTION
ClientServicesCredentialType	NavUserPassword , Username , or AccessControlService	The default value is Windows . When you change it to NavUserPassword , Username , or AccessControlService , client users are prompted for user name and password credentials.
DnsIdentity	The subject name of the service certificate.	The default value is <identity> . Replace this with the subject name or common name (CN) of the certificate that is used on the computer that is running Business Central Server.

3. Save and close the ClientUserSettings.config file.

When starting the Dynamics NAV Client connected to Business Central, users are prompted for a valid user name and password.

See Also

[Authentication and User Credential Types](#)

Business Central Web Server Overview

3/31/2019 • 4 minutes to read

Giving users access to data from the Business Central Web client, companion app, and Outlook add-in requires a Internet Information Services (IIS) website as part of your deployment. The website, which we refer to as Business Central Web Server instance, hosts the files that provide content and services to client users over the Internet. This article highlights several factors to consider to help you set up Business Central Web Server instances that suit your deployment requirements.

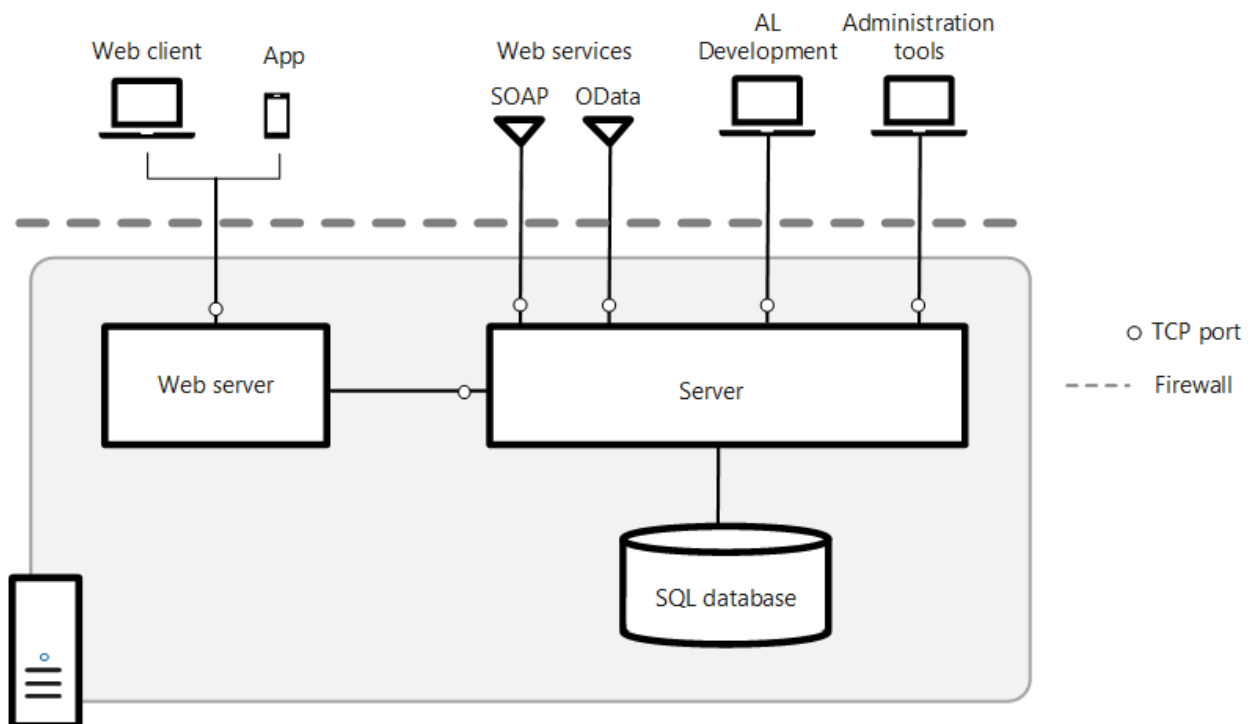
If you just want to get started installing the Business Central Web Server components, see [Install Business Central Using Setup](#).

ASP .NET Core on IIS

Business Central Web Server instances run on ASP.NET Core on IIS, which in part dictates the directory structure of the instances. For more information about ASP .NET Core, see [Introduction to ASP.NET Core](#).

Network Topology

The following illustration shows the component infrastructure that supports Business Central Web Server instances on your network.



Each Business Central Web Server instance must connect to a Business Central Server, which in turn connects to the database that contains the application and business data. Multiple Business Central Web Server instances can connect to the same Business Central Server. You can deploy these components on one computer or on separate computers. For example, you can install the Business Central Web Server instance on one computer and the Business Central Server and SQL Server database on another computer. The topology that you choose depends on the network resources and the infrastructure of the Business Central components. The installation and configuration process is different for each scenario.

For information about the common deployment scenarios, see [Deployment Topologies](#).

Creating a Business Central Web Server instance

There are two ways to create a Business Central Web Server instance. You can use the Business Central Setup Setup or the Business Central Web Server PowerShell cmdlets.

Using Business Central Setup Setup

Setup is the quickest way to get a web server instance up and running, and is typically how you install the first Business Central Web Server instance in your deployment.

- Setup installs the Business Central Web Server components, which does the following:
 - Installs and configure IIS with the required prerequisites, including Microsoft .NET Core - Windows Server Hosting
 - Installs a web server instance on IIS.
 - Installs components and files in a **WebPublish** folder that enables you to add additional web server instances without having to use the Business Central installation media (DVD).
- You can only use Setup to install a single Business Central Web Server instance.
- Setup does not let you choose the site deployment type for the web server instance. By default, it creates a SubSite instance. For more information, see [Site Deployment Types](#).
For information about how to install the Business Central Web Server components, see [Install Business Central Using Setup](#).

Using Business Central Web Server PowerShell cmdlets

There are several PowerShell cmdlets that enable you to create, configure, and remove Business Central Web Server instances from a command line interface. To create a web server instance, you use the [New-NAVWebServerInstance](#) cmdlet, which has the following advantages over Setup:

- You can create multiple web server instances.
- You have more flexibility regarding the [site deployment type](#) of the Business Central Web Server instances on IIS. For example, you can create a root-level website instance or a subsite application instance under a container website.

IMPORTANT

Using New-NAVWebServerInstance cmdlet requires that Microsoft .NET Core Windows Server Hosting is installed and IIS is installed and configured with the prerequisites. So unless you have previously installed the Business Central Web Server components by using Setup, you will have to install and configure the prerequisites manually. For more information about the prerequisites, see [Configure Internet Information Services](#).

For information about how to create a Business Central Web Server instance by using the New-NAVWebServerInstance cmdlet, see [Creating and Managing Business Central Web Server Instances Using PowerShell](#).

Deployment Phases

Typically, you will deploy the Business Central Web client in phases, which can influence the network topology and security settings that you deploy. For example, in the development phase, you develop, test, and fine-tune the application. In this phase, you might consider deploying the Business Central Web client in a single-computer scenario. When you move to the production phase, you deploy the Business Central Web client in the full network infrastructure.

Security

User Authentication

Business Central supports four methods for authenticating users who try to access the Business Central Web client: Windows, UserName, NavUserPassword, and AccessControlService. Windows authentication is configured by default. For more information, see [Users and Credential Types](#) and [Authentication and User Credential Type](#).

Service Account for Business Central Server and Business Central Database Access

When you install Business Central Server and Business Central database components, you must identify an Active Directory account to provide credentials for the servers. By default, Setup runs Business Central Server and the Business Central database under the Network Service account, a predefined local account that is used by the service control manager.

TIP

We recommend that you create and use a domain user account for running Business Central Server and accessing the Business Central database. The Network Service account is considered less secure because it is a shared account that can be used by other unrelated network services.

For more information, see [Provisioning a Service Account](#).

Securing the Connection to Microsoft Dynamics NAV Web Client With SSL

You can help secure Business Central data that is transmitted over the Internet by enabling Secure Sockets Layer (SSL) on the connection to the Business Central Web client. You can configure SSL when you install the Business Central Web Server components or after the installation.

For more information, see [Configure SSL to Secure the Web Client Connection](#).

See Also

[Configure Internet Information Services](#)

[Configuring the Business Central Web Server](#)

Configuring Business Central Web Server Instances

6/17/2019 • 11 minutes to read

Accessing Business Central from the Business Central Web client or App requires a Business Central Web Server instance on IIS. You create a Business Central Web Server instance for the Business Central Web Server by using the Business Central Setup to install the Business Central Web Server or by running the [New-NAVWebServerInstance cmdlet](#). When you set up a web server instance, you are configuring the connection from the Business Central Web Server to the Business Central Server instance. The connection settings, along with several other configuration settings, are saved in a configuration file for the web server instance.

About the configuration file

The configuration file for the web server instances is a .json file type called **navsettings.json**. The navsettings.json file is a Java Script Object Notification file type that is similar to files that use the XML file format.

After installation, you can change the configuration by modifying the navsettings.json. There are two ways to modify this file: directly or using PowerShell.

Where to find the navsettings.json file

The navsettings.json file is stored in the physical path of the web server instance, which is by default is `%systemroot%\inetpub\wwwroot\[WebServerInstanceName]`.

`[WebServerInstanceName]` corresponds to the name (alias) of the web server instance in IIS, for example, `c:\inetpub\wwwroot\BC140`.

Modify the navsettings.json file directly

1. Open the navsettings.json in any text or code editor, such as Notepad or Visual Studio Code.

Each setting is defined by a key-value pair.

- In the navsettings.json file, a setting has the format:

```
"keyname": "keyvalue",
```

The `keyname` is the name of the configuration setting and the `keyvalue` is the value.

For example, the configuration setting that specifies the Windows credential type for authenticating users is:

```
"ClientServicesCredentialType": "Windows",
```

Include values in double quotes.

2. Find the configuration settings that you want to change, and then change the values.

See the [Settings](#) section for a description of each setting.

3. When you are done making changes, save the file.
4. Restart the Business Central Web Server instance for the changes to take effect.

For example, in IIS Manager, in the **Connections** pane, select website node for Business Central Web

Server, and then in the **Actions** pane, choose **Restart**. Or, from your desktop, run `iisreset`.

Modify the navsettings.json file by using the Set-NAVWebServerInstanceConfiguration cmdlet

The PowerShell script module **NAVWebClientManagement.psm1** includes the [Set-NAVWebServerInstanceConfiguration cmdlet](#) that enables you to configure a web server instance.

1. Depending on your installation, run the Dynamics NAV Development Shell or Windows PowerShell as an administrator.

For more information, see [Get started with the Business Central Web Server cmdlets](#).

2. For each setting that you want to change, at the command prompt, run the following command:

```
Set-NAVWebServerInstanceConfiguration -Server [MyComputer] -ServerInstance [ServerInstanceName] -WebServerInstance [MyBCWebServerInstance] -KeyName [Setting] -KeyValue [Value]
```

Replace:

- `[MyComputer]` with the name of the computer that is running the Business Central Server
- `[ServerInstanceName]` with the name of the server instance, such as **BC140**.
- `[MyBCWebServerInstance]` with the name of the web server instance for the Business Central Web Server.
- `[KeyName]` with the name of the setting. Refer to the next section in this article.
- `[KeyValue]` with the new value of the setting.

Settings in the navsettings.json

The navsettings.json has the following structure, where settings are included under one of two root elements:

`NAVWebSettings` and `ApplicationIdSettings`:

```
{
  "NAVWebSettings": {
    "//ServerInstance": "Name of the Business Central Server instance to connect to (for client) or listen on (for server).",
    "ServerInstance": "BC140",
    [...more keys]
  },
  "ApplicationIdSettings": {
    "BlogLink": "https://myBCBlog.com",
    [...more keys]
  }
}
```

`//` indicates a comment that provides help for the setting, and has no affect on the Web Server instance.

The following table describes the settings that are available in the navsettings.json for each root element. If you do not see a setting in the file, this is because some settings are not automatically included as a key in the file. For these settings, you can add the key manually. If you do not add the key, the default value of the setting is used.


IMPORTANT

If modifying the file directly, place values in double quotes `" "`.

`NAVWebSettings` **element settings**

SETTING/KEYNAME	DESCRIPTION
AllowedFrameAncestors	<p>Specifies the host name of any web sites in which the Business Central Web client or parts of are embedded. By default, the Business Central Web Server will not allow a website to display it inside an iframe unless the website is hosted on the same web server. This value of this setting is a comma-separated list of host names (URIs). Wildcard names are accepted. For example:</p> <pre>https://mysite.sharepoint.com, https://*.myportal.com</pre>
GlobalEndpoints	<p>Specifies the comma-separated list of global endpoints that are allowed to call this website. The values must include http scheme and fully qualified domain name (FDQN), such as <code>https://financials.microsoft.com</code>.</p> <p>Default value: none</p>
LoadScriptsFromCdn	<p>Specifies whether to load scripts from Content Distribution Networks (CDNs). This only applies to scripts that are available from a CDN, like jQuery.</p> <p>If set to <code>false</code>, scripts will be loaded from the Web server, which is useful in, for example, an intranet scenario where there is no internet access.</p> <p>Default value: <code>true</code></p>
AllowNtlm	<p>Specifies whether NT LAN Manager (NTLM) fallback is permitted for authentication.</p> <p>To require Kerberos authentication, set this value to <code>false</code>.</p> <p>Values: <code>true</code>, <code>false</code></p> <p>Default value: <code>true</code></p>
ClientServicesChunkSize	<p>Sets the maximum size, in kilobytes, of a data chunk that is transmitted between Business Central Web Server and Business Central Server. Data that is transmitted between Business Central Web Server and Business Central Server is broken down into smaller units called chunks, and then reassembled when it reaches its destination.</p> <p>Values: From 4 to 80.</p> <p>Default value: 28</p>
ClientServicesCompressionThreshold	<p>Sets the threshold in memory consumption at which Business Central Web Server starts compressing data sets. This limits amount of consumed memory. The value is in kilobytes.</p> <p>Default value: 64</p>

SETTING/KEYNAME	DESCRIPTION
ClientServicesProtectionLevel	<p>Specifies the security services used to protect the data stream between the Business Central Web Server and Business Central Server. This value must match the value that is specified in the Business Central Server configuration file. For more information, see Configuring Business Central Server.</p> <p>Values: EncryptAndSign, Sign, None</p> <p>Default value: EncryptAndSign</p>
Server	<p>Specifies the name of the computer that is running the Business Central Server.</p> <p>Default value: localhost</p>
ServerInstance	<p>Specifies the name of the Business Central Server instance that the Business Central Web Server connects to.</p> <p>Default value: BC140</p>
ClientServicesCredentialType	<p>Specifies the authorization mechanism that is used to authenticate users who try to connect to the Business Central Web Server. For more information, see Authentication and User Credential Type.</p> <p>The credential type must match the credential type configured for the Business Central Server instance that the Business Central Web Server connects to. For information about how to set up the credential type on Business Central Server, see Configuring Business Central Server.</p> <p>Values: Windows, UserName, NavUserPassword, AccessControlService</p> <p>Default value: Windows</p>
ClientServicesPort	<p>Specifies the TCP port for the Business Central Server. This is part of the Business Central Server's URL.</p> <p>Values: 1-65535</p> <p>Default value: 7046</p>
ManagementServicesPort	<p>The listening TCP port for the Business Central management endpoint.</p> <p>Valid range: 1-65535</p> <p>Default value: 7045</p>

SETTING/KEYNAME	DESCRIPTION
ServicePrincipalNameRequired	<p>If this parameter is set to <code>true</code>, then the Business Central Web Server can only connect to a Business Central Server instance that has been associated with a service principal name (SPN).</p> <p>If this parameter is set to <code>false</code>, then the Business Central Web Server attempts to connect to the configured Business Central Server service, regardless of whether that service is associated with an SPN.</p> <p>For more information about SPNs, see Configure Delegation.</p> <p>Default: <code>false</code></p>
SessionTimeout	<p>Specifies the maximum time that a connection between the Business Central Web Server and the Business Central Server can remain idle before the session is stopped.</p> <p>In the Business Central Web Server, this setting determines how long an open Business Central page or report can remain inactive before it closes. For example, when the SessionTimeout is set to 20 minutes, if a user does not take any action on a page within 20 minutes, then the page closes and it is replaced with the following message: The page has expired and the content cannot be displayed.</p> <p>The time span has the format [dd.]hh:mm:ss[.ff]:</p> <ul style="list-style-type: none"> - dd is the number of days - hh is the number of hours - mm is the number of minutes - ss is the number of seconds - ff is fractions of a second <p>Default value: 00:20:00</p>
RequireSsl	<p>Specifies whether SSL (https) is required. If the value is set to <code>true</code> all cookies will be marked with a <code>\u0027secure\u0027</code> attribute. If SSL is enable on the Web server, you should set this to <code>true</code>.</p> <p>Values: <code>true</code>, <code>false</code></p> <p>Default value: <code>false</code></p>
ShowPageSearch	<p>Specifies whether to show the  Tell me what you want to do icon in the Business Central header. This feature lets users find Business Central objects, such as pages, reports, and actions.</p> <p>If you do not want to show the Tell me what you want to do icon, then set the parameter to <code>false</code>.</p> <p>Default value: <code>true</code></p>

SETTING/KEYNAME	DESCRIPTION
UnknownSpnHint	<p>Specifies whether to use a server principal name when establishing the connection between the Business Central Web Server and Business Central Server. This setting is used to authenticate the Business Central Server, and it prevents the Business Central Web Server from restarting when it connects to Business Central Server for the first time. You set values that are based on the value of the ServicePrincipalNameRequired key.</p> <p>Value: The value has the following format.</p> <p>(net.tcp://BCServer:Port/ServerInstance/Service)=NoSpn SPN</p> <ul style="list-style-type: none"> - BCServer is the name of the computer that is running the Business Central Server. - Port is the port number on which the Business Central Server is running. - ServerInstance is the name of the Business Central Server instance. - NoSpn SPN specifies whether to use an SPN. If the ServicePrincipalNameRequired key is set to <code>false</code>, then set this value to NoSpn. If the ServicePrincipalNameRequired key is set to <code>true</code>, then set this value to Spn. <p>Default value: (net.tcp://localhost:7046/BC140/Service)=NoSpn</p> <p>If you set this key to the incorrect value, then during startup, the Business Central Web Server will automatically determine a correct value. This will cause the Business Central Web Server to restart. Note: For most installations, you do not have to change this value. Unlike the Dynamics NAV Client connected to Business Central, this setting is not updated automatically. If you want to change the default value, then you must change it manually.</p>
DnsIdentity	<p>Specifies the subject name or common name of the service certificate for Business Central Server.</p> <p>This parameter is only relevant when the ClientServicesCredentialType is set to UserName, NavUserPassword, or AccessControlService, which requires that security certificates are used on the Business Central Web Server and Business Central Server to protect communication. Note: You can find the subject name by opening the certificate in the Certificates snap-in for Microsoft Management Console (MMC) on the computer that is running Business Central Web Server or Business Central Server.</p> <p>For more information, see Authentication and User Credential Type.</p> <p>Value: The subject name of the certificate.</p> <p>Default value: none</p>

SETTING/KEYNAME	DESCRIPTION
AuthenticateServer	<p>Specifies whether to authenticate the server by enabling service identity checks on protocol between the Web server and the Business Central Server instance.</p> <p>Values: <input type="checkbox"/> true , <input type="checkbox"/> false Default value: <input type="checkbox"/> true</p>
HelpServer	<p>Specifies the name of the Business Central Help Server if the deployment uses Help Server. If the deployment uses an online library, remove this setting.</p> <p>Default value: none</p>
HelpServerPort	<p>Specifies the TCP port on the specified Business Central Help Server if the deployment uses Help Server. If the deployment uses an online library, remove this setting.</p> <p>Default value: none</p>
OfficeSuiteShellServiceClientTimeout	<p>Defines the time Business Central will wait for the Office Suite Shell Service to respond.</p> <p>Important: This setting has been deprecated in Business Central, and it has no effect on the Web Server instance.</p> <p>Default value: 10</p>
UseAdditionalSearchTerms	<p>Specifies whether Tell me uses the additional search terms that are defined on pages and reports.</p> <p>The additional search terms are specified by the AdditionalSearchTerms and AdditionalSearchTermsML properties.</p> <p>If you set this to <input type="checkbox"/> false the additional search terms are ignored.</p> <p>Default value: true</p>
DefaultRelativeHelpPath	<p>Specifies the default Help article to open if no other context-sensitive link is specified.</p> <p>Default value: none</p>
PersonalizationEnabled	<p>Specifies whether personalization is enabled in the Business Central Web client. Set to <input type="checkbox"/> true to enable personalization.</p> <p>For more information, see Managing Personalization.</p>

ApplicationIdSettings **element settings**

SETTING/KEYNAME	DESCRIPTION
-----------------	-------------

SETTING/KEYNAME	DESCRIPTION
BaseHelpUrl	<p>Specifies the link to the online Help library that the deployment uses, such as https://mysite.com/{0}/mylibrary/.</p> <p>Default value: none</p> <p>For more information, see Configuring the Help Experience.</p>
BlogLink	<p>Specifies the target of the blog link on the Help & Support page. Use this link to give users access to your end-user blog.</p> <p>Value: a valid URL Default value: https://go.microsoft.com/fwlink/?linkid=2076643</p>
ComingSoonLink	<p>Specifies the target of coming soon link on the Help & Support page. Use this link to give users access to information about your roadmap or any upcoming features and fixes.</p> <p>Value: A valid URL. Default value: https://go.microsoft.com/fwlink/?linkid=2047422</p>
CommunityLink	<p>Specifies the URL to a community or resource for sharing information.</p> <p>Value: a valid URL Default value: http://go.microsoft.com/fwlink/?LinkId=287089</p>
ContactSalesLink	<p>Specifies the target of the contact sales link on the Help & Support page. Use this link to give users access to your sales-focused web page where users can engage with your sales process. Note This setting and link are not used for Business Central on-premises.</p>
SigninHelpLink	<p>Specifies the URL to open if the user selects help on the sign in page box.</p> <p>Value: a valid URL Default value: none</p>

See Also

[Setting up Multiple Business Central Web Server Instances](#)

[Install Business Central Components](#)

[Business Central Web Server Overview](#)

[Configuring Business Central Server](#)

[Configuring the Help Experience](#)

Configuring Internet Information Services for Business Central

3/31/2019 • 2 minutes to read

This topic describes the configuration of Internet Information Service (IIS) that is required for running Business Central. When you install the Business Central Web Server components, you can use the Business Central Setup to install and configure the IIS features, so you do not have to do this manually.

Required IIS Features

IIS must have the following features enabled:

- HTTP Activation
- NET Extensibility 4.5, .NET Extensibility 4.6, or .NET Extensibility 4.6 (depending on Windows version)
- ASP.NET 4.5, ASP.NET 4.6, or ASP.NET 4.7 (depending on Windows version)
- ISAPI Extensions
- ISAPI Filters
- Request Filtering
- Windows Authentication
- Default Document
- Directory Browsing
- HTTP Errors
- Static Content

Configure Headers in Application Request Routing (ARR) Rules

If you are hosting the Business Central Web Server components on an IIS server farm that is using Application Request Routing (ARR), you must configure headers. Business Central Web Server runs on ASP .NET Core, which requires both an `X-Forwarded-For` header and `X-Forwarded-Proto` header in ARR routing rules. However, by default, ARR only adds the `X-Forwarded-For` header; not the `X-Forwarded-Proto` header. So will have to configure the `X-Forwarded-Proto` header manually.

On the server farm in IIS, add or edit a routing rule to include a server variable for `X-Forwarded-Proto`. For example, using IIS Manager, select **Routing Rules > URL Rewrite > Edit > Server Variables**, and then add a server variable that has the following settings:

NAME	VALUE	REPLACE
<code>HTTP_X_FORWARDED_PROTO</code>	<code>http</code> or <code>https</code>	<code>true</code>

See Also

[Business Central Web Server Overview](#)

Configuring SSL to Secure the Business Central Web Client Connection

3/31/2019 • 3 minutes to read

We recommend that you secure Business Central data that is transmitted over the Internet by enabling Secure Sockets Layer (SSL) on the connection to Business Central Web client.

SSL is a web protocol that encrypts data that is transmitted over a network to make the data and the network more secure and reliable. A website that is enabled with SSL uses Hypertext Transfer Protocol Secure (HTTPS) instead of Hypertext Transfer Protocol (HTTP) as a communication protocol. Enabling SSL on a website requires that an SSL certificate is installed on the web server. An SSL certificate is a small file that the web server uses to prove its identity and establish a trusted connection with the browser that is trying to access Business Central Web client. When a browser connects to the Business Central Web client, the web server replies by sending its certificate to the browser. This certificate contains the web server's public encryption key and the name of the authority that granted the certificate. The browser verifies the certificate using the authority's public key.

To configure SSL, you must follow the steps in this article.

NOTE

You can configure SSL when you install the Business Central Web Server components using Business Central Setup .

Obtaining and Installing an SSL Certificate

In a production environment, you should obtain an SSL certificate from a certification authority. Some large organizations may have their own certification authorities, and other organizations can request a certificate from a third-party organization. In a test environment or development environment, you can create your own self-signed certificate.

Adding an HTTPS Binding That Uses the Certificate on the Dynamics 365 Business Central Website

After you get the certificate, you add a binding to the https protocol on the website. When you add the binding, you associate it with the certificate.

Add an https binding with the certificate to the website

1. Open Internet Information Services (IIS) Manager.
2. In the **Connections** pane, expand the **Sites** node, and then choose the Business Central Web client site to which you want to add the binding.

By default, the site has the name **Dynamics 365 Business Central Web Client**.

3. In the **Actions** pane, choose **Bindings**.
4. In the **Site Bindings** dialog box, choose **Add**.
5. In the **Add Site Binding** dialog box, set the **Type** field to **https**.

You can use the default port 443 or change it to another port. If you change it to another port, you will have to provide the port number in the URL when you try to open the client.

6. Set the **SSL certificate** field to the certificate that you obtained or created for the site.

7. Choose the **OK** button, and then choose the **Close** button.

Redirecting HTTP to HTTPS (Optional)

To ensure that users always access the site that is secured with SSL, you can automatically redirect HTTP requests to HTTPS. This means that users do not have to explicitly include https in the URL in the browser. For example, the nonsecure URL of the Business Central Web client could be `http://MyWebclient:8080/BC130` and the secure URL could be `https://MyWebclient:443/BC130`. If a user types `http://MyWebclient:8080/BC130`, the browser automatically redirects to `https://MyWebclient:443/BC130`.

There are different ways to redirect HTTP requests to HTTPS. The following procedure describes how to redirect HTTP requests to HTTPS by installing the Microsoft Application Request Routing for IIS 8 and modifying the [configuration file](#) for the Business Central Web Server instance.

Redirect HTTP to HTTPS

1. Download and install Microsoft Application Request Routing for IIS. For example, you can download from [Microsoft Application Request Routing](#).
2. On the computer that is running Business Central Web Server components, open the web.config file for the Business Central Web Server instance. Use a text editor, such as Notepad.

The web.config file is located in the physical path of the web application on IIS. By default, the path is %systemroot%\inetpub\wwwroot\[VirtualDirectoryName]. For example, the folder for the default application is %systemroot%\inetpub\wwwroot\BC140.

3. In the `<system.webServer>` element, add the following elements.

```
<rewrite>
  <rules>
    <rule name="Redirect to HTTPS">
      <match url="(.*)" />
      <conditions>
        <add input="{HTTPS}" pattern="off" ignoreCase="true" />
      </conditions>
      <action type="Redirect" url="https://{SERVER_NAME}/{R:1}" redirectType="SeeOther" />
    </rule>
  </rules>
</rewrite>
```

4. Save the navsettings.json file.

See Also

[Business Central Web Server Overview](#)

Setting Up Multiple Business Central Web Server Instances Using PowerShell

3/31/2019 • 5 minutes to read

Although you can use the Business Central Setup to install the Business Central Web Server components and create a single web server instance in IIS for client connection, there may be scenarios when you want to set up multiple instances. For example, you could set up a separate Business Central Web Server instance for the different companies of a business. For this scenario, you can use the Business Central Web Server PowerShell cmdlets, which are outlined in the following table.

CMDLET	DESCRIPTION
New-NAVWebServerInstance	Creates a new Business Central Web Server instance and binds this instance to a Business Central Server instance.
Set-NAVWebServerInstanceConfiguration	Specifies configuration values for a named Business Central Web Server instance.
Get-NAVWebServerInstance	Gets the information about the Business Central Web Server instances that are registered on a computer.
Remove-NAVWebServerInstance	Removes an existing Business Central Web Server instance.

Get started with the Business Central Web Server cmdlets

The Business Central Web Server cmdlets are contained in the PowerShell script module **NAVWebClientManagement.psm1**, which is available on the Business Central installation media (DVD).

The module is installed with the Business Central Server or the Business Central Web Server components.

There are different ways to launch this module and start using the cmdlets:

- If you are working on the computer where the Business Central Server was installed, run the Business Central Administration Shell as an administrator.

For more information, see [Business Central PowerShell Cmdlets](#).
- If you installed the Business Central Web Server components, just start Windows PowerShell as an administrator.
- Otherwise, start Windows PowerShell as an administrator, and use the [Import-Module](#) cmdlet to import the **NAVWebClientManagement.psm1** file:

```
Import-Module -Name [filepath]
```

For example:

```
Import-Module -Name "C:\Program Files\Microsoft Dynamics 365 Business  
Central\130\Service\NAVWebClientManagement.psm1"
```

For more information about starting Windows PowerShell, see [Starting Windows PowerShell](#).

Creating Business Central Web Server instances

Get Access to the WebPublish folder

To create a new web server instance, you need access to the **WebPublish** folder that contains the content files for the Business Central Web Server components.

- This folder is available on the Business Central installation media (DVD) and has the path "DVD\WebClient\Microsoft Dynamics NAV\13x\Web Client\WebPublish".
- If you installed the Business Central Web Server components, this folder has the path "C:\Program Files\Microsoft Dynamics 365 Business Central\140\Web Client\WebPublish".

You can use either of these locations or you can copy the folder to more convenient location on your computer or network.

Decide on the site deployment type for the instance

When you create a new Business Central Web Server instance, you can choose to create either a RootSite or SubSite type. Each instance type has a different hierarchical structure in IIS, which influences its configuration and the URLs for the accessing from the Business Central Web client.

RootSite

A *RootSite* instance is a root-level web site that is complete with content files for serving the Business Central Web client. It is configured with its own set of bindings for accessing the site, such as protocol (http or https) and communication port. The structure in IIS looks like this:

```
- Sites
- BusinessCentralWebSite (web site)
  + nn-NN (language versions)
  + www (content)
  navsettings.json
  ...
```

The Business Central Web client URL for the RootSite instance has the format:

```
http://[WebserverComputerName]:[port]
```

For example: `http://localhost:8080`.

SubSite

A *SubSite* instance is a web application that is under a container web site. The container web site is configured with a set of bindings, but the site itself has no content files. The content files are contained in the application (SubSite). The SubSite inherits the bindings from the container web site. This is the deployment type that is created when you install Business Central Web Server components in the Setup wizard. Using the `New-NAVWebServerInstance` cmdlet, you can add multiple SubSite instances in the container web site. The structure in IIS for two instances looks like this in IIS:

```

- Sites
- BusinessCentralWebSite (web site)
  - BusinessCentralWebInstance1 (application)
    + nn-NN (language versions)
    + www
    navsettings.json
    ...
  - BusinessCentralWebInstance2 (application)
    + nn-NN (language versions)
    + www
    navsettings.json
    ...

```

The Business Central Web client URL of a SubSite instance is generally longer than a RootSite because it also contains the application's alias (or virtual path) for the instance, which you define. The URL for a SubSite instance has the format:

```
http://[WebserverComputerName]:[port]/[WebServerInstance]
```

For example: `http://localhost:8080/BusinessCentralWebInstance1` and

```
http://localhost:8080/BusinessCentralWebInstance2
```

Run the New-NAVWebServerInstance cmdlet

At the command prompt, run the New-NAVWebServerInstance cmdlet. The following are simple examples for creating a RootSite and SubSite deployment type.

RootSite example:

```

...
New-NAVWebServerInstance -WebServerInstance MyBCWebsite -Server MyBCServer -ServerInstance MyBCServerInstance -
SiteDeploymentType RootSite -WebSitePort 8081 -PublishFolder "C:\Web Client\WebPublish"
...

```

SubSite example:

```

...
New-NAVWebServerInstance -WebServerInstance MyWebApp -Server MyBCServer -ServerInstance MyBCServerInstance -
SiteDeploymentType Subsite -ContainerSiteName MySiteContainer -WebSitePort 8081 -PublishFolder
"C:\WebClient\WebPublish"
...

```

- Substitute *MyBCWebsite* with the name that you want to give the web application in IIS for the web server instance. If you are creating a SubSite deployment type, this name will become part of the URL for opening the Business Central Web client application, for example, `http://MyWebServer:8081/MyWebApp`.
- Substitute *MyBCServer* to the name of the computer that is running the Business Central Server to which you want to connect.
- Substitute *MyBCServerInstance* with the name of the Business Central Server instance to use.
- Substitute *MySiteContainer* with name of the container web site under which you want to add the instance. If you specify a name that does not exist, then a new container web site will be created, which contains the new web server instance.
- Substitute *8081* with the port number that you want to bind the instance to. If you do not specify a port number, then port 80 is used.
- Substitute *C:\WebClient\WebPublish* with the path to your WebPublish folder. By default, the cmdlet looks

in the 'C:\Program Files\Microsoft Dynamics 365 Business Central\140\Web Client' folder. So if you are working on a computer where the Business Central Web Server components are installed, you do not have to set this parameter.

NOTE

This command only sets the required parameters of the NAVWebServerInstance cmdlet. The cmdlet has several other parameters that can use to configure the web server instance. For more information about the syntax and parameters, see [New-NAVWebServerInstance](#).

Modifying a Business Central Web Server instance

After you create the web server instance, if you want to change its configuration, you can use the Set-NAVWebServerInstanceConfiguration cmdlet. Or, you can modify the configuration file (navsettings.json) of the instance directly. For more information, see [Configuring Web Server Instances](#).

See Also

[Business Central Web Server Overview](#)
[Configuring Web Server Instances](#)

Migrating to Multitenancy

3/31/2019 • 2 minutes to read

You can choose to migrate your Business Central solution to a multitenant deployment architecture where you maintain a single application that is used by two or more companies that store their data in separate databases.

This can make maintenance of your solution easier if you support multiple customers with the same application functionality.

Tenants and Companies

When you upgrade your application and the data to Business Central, you have a database that has the same number of companies as you had before the upgrade. This database is considered a *tenant*. This does not mean that you have to turn your solution into a multitenant deployment. But it means that you can if you want to.

For example, your Business Central deployment in the earlier version consisted of a database that has 20 companies. In other words, you support 20 companies that all share the same application functionality. In this example, the companies are separate companies that have nothing to do with each other except that they are supported by you in one database. In Business Central, you can choose to extract the application-wide tables into a separate database and keep the data for all 20 companies in the original database. This becomes a single-tenant business data database. Then, you can choose to split the business data database into one for each company so that you run a truly multitenant environment. The application is stored separately in the application database, and you maintain application functionality centrally. When you modify the application, you make the changes available to one tenant at a time. As a result, if something goes wrong, all other tenants are not affected.

Compare this to earlier versions of Business Central where a database could contain several companies. These companies could be related or not, but they would all use the same application and write to the same database. Also, when you modified the application, it would affect all companies immediately. So if something went wrong, all companies would be affected.

NOTE

The email logging functionality in Business Central requires the Business Central Server service account to have access to the Exchange server. But in a multitenant deployment, this is not always possible.

In multitenant deployments of Business Central, permission sets are stored centrally in the application database, so only the administrator of the central application can add, remove, or modify permission sets. Instead, the tenants can use user groups to manage permissions.

Migration Process

If you decide to move to a multitenant architecture, you must complete the following steps:

1. Move the tables that describe the application to a separate database. For more information, see [Separating Application Data from Business Data](#).

After this step, you have two databases: an application database and a business data database.

2. Split the business data database into one for each company. For more information, see [Creating Tenants from Companies](#).

After this step, you have an application database and a business data database for each company in the

original database. The company-specific business data databases are tenants, and your solution is multitenant.

If you want to move back to storing application tables and business data in a single database, you can use the Business Central Windows PowerShell cmdlets to merge the databases. For more information, see [Merging an Application Database with a Tenant Database](#).

See Also

[Separating Application Data from Business Data](#)

[Creating Tenants from Companies](#)

[Upgrading the Application Code](#)

[Upgrading the Data](#)

[Upgrading to Business Central](#)

Installation Considerations for Microsoft SQL Server and Business Central

3/31/2019 • 7 minutes to read

This article describes the requirements for installing and configuring Microsoft SQL Server to work with Business Central.

Dynamics NAV can run on Microsoft SQL Server and Microsoft Azure SQL Database. For a list of supported editions of SQL Server, see [SQL Server Requirements](#).

Using Microsoft SQL Server

Storage

Use different disks or disk partitions for the following:

- Windows operating system.
- Data files for the system databases.
- Log files for system and user databases.
- Data and log files for the TempDB database.

For optimal read/write performance, make sure that disks that are used for SQL Server data files are formatted using 64 KB block size.

Virus scanning

To help you decide which kind of antivirus software to use on the computers that are running Microsoft SQL Server in your environment, see [How to choose antivirus software to run on computers that are running SQL Server](#).

Memory

For optimal read performance, maximize the available memory on the server according to the version and edition of SQL Server used. Refer to the SQL Server documentation for maximum values.

SQL Server components

If you are installing Microsoft SQL Server for use with Business Central, then install the following components:

- Database Engine Services
- Client Tools Connectivity
- Management Tools - Complete

Setup options for Microsoft SQL Server

When you are running Microsoft SQL Server Setup, you must provide additional information. Your responses can affect how you use SQL Server with Dynamics NAV.

TempDB database configuration

For servers with less than 8 cores, create as many data files for the TempDB database as the number of cores. For servers with more than 8 cores, start with 8 data files, and increment with 4 files at a time, if needed.

Make sure that all data files for the TempDB database are of the same size.

Consider putting data and log files for TempDB on a local SSD drive if you are using SAN storage.

Data file and log file configuration

Auto-growth of the database and/or transaction log files in production can degrade performance as all transaction must queue up and wait for SQL Server to grow the file before it can begin to process transactions again. This can create bottlenecks. We strongly recommend growing data and log files during off-peak periods and by 10% to 25% of the current size. We do not recommend disabling "Auto-Grow", as in an emergency it is still better to have SQL Server to auto-grow files than to run out of disk space and bring the database down.

Max degree of parallelism (MAXDOP)

The SQL queries generated by Dynamics NAV is of OLTP type (many, small transactions). It is therefore recommended to run Dynamics NAV with `MAXDOP` set to the value 1.

On SQL Server 2014, `MAXDOP` can only be set on the instance level, changing an advanced server configuration option. On SQL Server 2016, `MAXDOP` can be set on the database level, changing a database scoped configuration.

Both advanced server configuration options and database scoped configurations can be set by using SQL Server Management Studio, see the SQL Server documentation for details.

NOTE If you are running SQL Server Enterprise Edition, index maintenance can be done in parallel. If you run maintenance jobs to do this work in off-peak hours, you might want to set `MAXDOP` back to 0 while running these jobs. On SQL Server 2016, it is possible to set MAXDOP directly in the Rebuild Index Task wizard.

Instance configuration

If you plan on installing the Business Central Demo database, and you want Business Central Setup to use an already installed version of SQL Server (and not to install SQL Server Express), you must create a SQL Server instance named **NAVDEMO** in SQL Server before you run Setup. Otherwise, Setup will install SQL Server Express automatically, even if there is a valid version of SQL Server already on the computer. If you do not plan to install the Demo database, or if you have no objection to using SQL Server Express, you are free to use the **default instance** and **Instance ID** on the **Instance Configuration** page, or to specify any instance name.

Database engine service

Each SQL Server instance is run by its own windows service. The following two things are important to configure for these services

Startup options

Enable trace flags 1117 and 1118 as startup options for SQL Server 2014. For SQL Server 2016, these trace flags are enabled by default.

Startup options can be set by using SQL Server Configuration Manager, see the SQL Server documentation for details.

Service account

We recommend that you use dedicated domain user accounts for the Windows services running your Business Central Server instances and your SQL Server instances, instead of a Local System account or the Network Service account.

The Business Central Server account must have privileges on the SQL Server instances and on the Dynamics NAV database(s). See [Provisioning the Server Service Account](#) for details.

For installations on SQL Server 2014, consider adding the service account for the SQL Server engine to the **Perform Volume Maintenance Tasks** security policy. For SQL Server 2016, it is possible to do this from the installer.

Database configurations

After Dynamics NAV has been installed, it is important to check a few settings on the Dynamics NAV database(s). This is especially important for databases, which have been upgraded from previous versions of SQL Server.

Statistics

The databases used by Business Central should have set the options `AUTO_CREATE_STATISTICS` and

AUTO_UPDATE_STATISTICS to the value ON (this is the default behavior and should not be changed)

SQL Server (2014 and earlier) uses a threshold based on the percent of rows changed before triggering an update of the statistics for a table regardless of the number of rows in the table. It is possible to change this behaviour by setting trace flag 2371 as a startup option for the instance. See Knowledge Base article ID 2754171, <https://support.microsoft.com/en-gb/help/2754171/controlling-autostat-auto-update-statistics-behavior-in-sql-server> for more information about when to set this trace flag.

SQL Server (starting with 2016 and under the compatibility level 130) uses a threshold that adjusts according to the number of rows in the table. With this change, statistics on large tables will be updated more often.

Even with "Auto Update Statistics" enabled, we still strongly recommend running a periodic SQL Agent job to update statistics. This is because "Auto Update Statistics" will only be triggered according to the rules described above. On large tables with tens of millions of records (such as Value Entry, Item Ledger Entry and G/L Entry), a small percentage of data in a given statistic such as [Entry No.] can change and have a material effect on the overall data distribution in that statistic. This can cause inefficient query plans, resulting in degraded query performance until any threshold is reached. We recommend using the T-SQL procedure "sp_updatestats" to update statistics, as it will only update statistics where data has been changed. We recommend creating a SQL Agent Job that runs daily or weekly (depending on transaction volume) during off-peak hours to update all statistics where data has changed.

Other database options

We recommend to set the database option PAGE_VERIFY to the value CHECKSUM for all databases (including TEMPDB) as this is the most robust method of detecting physical database corruption. This is the default setting for new installations.

Backup

Remember to setup backup of both system and user databases. Remember also to test restore procedures regularly.

Using Microsoft Azure SQL Database

You can deploy a Business Central database to Azure SQL Database. Azure SQL Database is a cloud service that provides data storage as a part of the Azure Services Platform.

To optimize performance, we recommend that the Business Central Server instance that connects to the database is also deployed on a virtual machine in Azure. Additionally, the virtual machine and SQL Database must be in the same Azure region.

For development and maintenance work on Business Central applications, if the Dynamics NAV Development Environment is installed on the same virtual machine in Azure as the Business Central Server, then you can connect to the Azure SQL database from the development environment.

For more information, see [Deploying a Business Central Database to Azure SQL Database](#).

Data Encryption between Business Central Server and SQL Server

When SQL Server and Business Central Server are running on different computers, you can make this data channel more secure by encrypting the connection with IPsec. (Other encryption options are not supported.) For information on how to do this, see [Encrypting Connections to SQL Server](#), which is part of SQL Server 2008 Books Online in MSDN library.

See Also

[Data Access](#)

[Troubleshooting: SQL Server Connection Problems](#)

Configuring the Business Central Database

3/31/2019 • 7 minutes to read

For a Business Central Server instance to connect to and access a database in SQL Server, the server instance must be authenticated by the database. As in SQL Server, Business Central supports two authentication modes for database communication: Windows Authentication and SQL Server Authentication. When you set up Business Central, you must ensure that database authentication is configured correctly on both the server instance and database, otherwise the server instance will not be able to connect to the database. By default, Windows Authentication is configured on the server instance and database. With Windows Authentication the Business Central Setup does the work for you.

This article describes how to configure SQL Server Authentication. You perform the configuration in two places: on the databases in SQL Server and on the Business Central Server instance. The procedure is different when the Business Central Server instance is configured as a multitenant server instance than when it is not a multitenant server instance.

Set Up an Encryption Key

When using SQL Server authentication, Business Central requires an encryption key to encrypt the credentials (user name and password) that the Business Central Server instance uses to connect to the Business Central database in SQL Server. The encryption key must be installed on the computer where the Business Central Server is installed and also in the database in SQL Server. In a multitenant deployment, the encryption key must be installed in the application database.

To set up an encryption key, you can use one of the following methods:

- You can create and import your own encryption key by using Business Central Administration Shell cmdlets, as described in the following procedure.
- If you are configuring SQL Server authentication on a Business Central Server instance for the first time, you can use the Business Central Server Administration tool which can automatically create and install a system encryption key. If you decide to use this method, no action is required.

Create and import encryption key

1. In the Business Central Administration Shell, run the [New-NAVEncryptionkey cmdlet](#).

This creates a file that contains an encryption key. If you already have an encryption key file, you can skip this step.

2. Run the [Import-NAVEncryptionkey cmdlet](#) to install the encryption key on the Business Central Server instance and database.

On the computer running the Business Central Server instance, the encryption key file has the name BC140.key and is stored in the `%systemroot%\ProgramData\Microsoft\Microsoft Dynamics NAV\[version]\Server\Keys`. In the database, the encryption key is registered in the `dbo.ndopublicencryptionkey` table. In a multitenant deployment, the encryption key is registered in the application database.

Configure SQL Authentication on the Database

This section describes how to configure a Business Central database to use SQL Server Authentication with a Business Central Server instance. You can complete the steps in this procedure by using SQL Server Management Studio or Transact-SQL.

IMPORTANT

In a deployment where the Business Central Server instance is configured as a multitenant server instance, you must complete the following procedure on the application database and tenant database.

Configure SQL Server Authentication on the database in SQL Server

1. Configure the SQL Server instance (Database Engine) that hosts the Business Central database to use SQL Server Authentication.

To use SQL Server authentication, you configure the database instance to mixed authentication mode (SQL Server and Windows Authentication). For more information, see [Change Server Authentication Mode](#).

2. In the SQL Server instance, create a login that uses SQL Server authentication.

For more information, see [Create a Login](#).

3. Map the login to a user in the Business Central database, and add the user to the **db_owner** role of the Business Central database.

For more information, see [Create a Database User](#).

Configure SQL Server Authentication on the Business Central Instance (Non-Multitenant)

You configure the Business Central Server instance with the login credentials (user name and password) of the user account in the Business Central database in SQL Server that you want to use for authentication. You can do this using the Business Central Server Administration tool or Business Central Administration Shell.

Configure SQL Authentication on a server instance using Business Central Server Administration tool

1. Open the Business Central Server Administration tool.
2. In the console tree, which is the left pane, expand the node for the computer that contains the Business Central Server instance, and then select the Business Central Server instance.
3. In the **Actions** pane, choose **Database Credentials**.
4. On the **Database Credentials** page, choose the **Edit** button.
5. Set the **Database Authentication Type** to **SQL Authentication**.
6. In the **Database User Name** field, type the login name for the database user that you want to use to access the Business Central database in SQL Server.
7. In the **Password** field, type the login password for the database user that you want to use to access the Business Central database in SQL Server.
8. Choose the **Save** button, and then on the **Enable Encryption on SQL Server Connections** dialog box, choose the **OK** button.

Encryption keys are used to help secure the login credentials over the connection between the Business Central Server instance and the Business Central database in SQL Server.

9. On the **Information** dialog box about encryption, choose the **OK** button.

This dialog box is to inform you to enable encryption on SQL Server connections, which is disabled by default.

10. If you want to enable encryption on SQL Server connections, in the **Action** pane, choose **Configuration**,

and then choose the **Edit** button. In the **Database** tab, select **Enable Encryption on SQL Connections**, choose the **Save** button, and then the **OK** button.

11. Restart the server instance.

Configure SQL Authentication on a server instance using Business Central Administration Shell

- If you are modifying an existing Business Central Server instance, run the [Set-NAVServerConfiguration cmdlet](#).

Use the *DatabaseCredentials* parameter to provide the login credentials of the database user that you want to use to access the application database.

- If you are creating a new Business Central Server instance, run the [New-NAVServerInstance cmdlet](#).

Use the *DatabaseCredentials* parameter to provide the login credentials of the database user that you want to use to access the application database.

Configure SQL Server Authentication on Business Central Instance in a Multitenant Deployment

This section describes how to configure a Business Central database to use SQL Server Authentication with a Business Central Server instance. You can complete the steps in this procedure by using SQL Server Management Studio or Transact-SQL.

To configure a SQL Server Authentication on a Business Central Server instance, you set up the server instance with the login credentials (user name and password) for the user accounts for the application and tenant databases in SQL Server. You can do this using the Business Central Server Administration tool or Business Central Administration Shell.

Configure SQL Authentication using Business Central Server Administration tool

1. Open the Business Central Server Administration tool.
2. In the console tree, which is the left pane, expand the node for the computer that contains the Business Central Server instance, and then select the Business Central Server instance.
3. Configure SQL Server Authentication with the application database as follows:
 - a. In the **Actions** pane, choose **Database Credentials**.
 - b. On the **Database Credentials** page, choose the **Edit** button.
 - c. Set the **Database Authentication Mode** to **SQL Server Authentication**.
 - d. In the **Database User Name** field, type the login name for the database user that you want to use to access the Business Central application database in SQL Server.
 - e. In the **Password** field, type the login password for the database user that you want to use to access the Business Central database in SQL Server.
 - f. Choose the **Save** button, and then on the **Enable Encryption on SQL Server Connections** dialog box, choose the **OK** button.

Encryption keys are used to help secure the login credentials over the connection between the Business Central Server instance and the Business Central database in SQL Server.

- g. On the **Information** dialog box about encryption, choose the **OK** button.

This dialog box is to inform you to enable encryption on SQL Server connections, which is disabled by default.

- h. If you want to enable encryption on SQL Server connections, in the **Action** pane, choose **Configuration**, and then choose the **Edit** button. In the **Database** tab, select **Enable Encryption on SQL Connections**, choose the **Save** button, and then the **OK** button.
4. To configure SQL Server Authentication with the tenant database, mount the tenant to the Business Central Server instance and specify the login credentials (user name and password) for the database user that you want to use to access the Business Central tenant database in SQL Server.

If the tenant is already mounted to the Business Central Server instance, you must dismount the tenant, and mount it again.

For more information see [Mount or Dismount a Tenant](#).

5. Restart the server instance.

Configure SQL Authentication using Business Central Administration Shell

1. Configure SQL Server Authentication with the application database as follows:

- If you are modifying an existing Business Central Server instance, run the [Set-NAVServerConfiguration cmdlet](#).

Use the *DatabaseCredentials* parameter to provide the login credentials of the database user that you want to use to access the application database.

- If you are creating a new Business Central Server instance, run the [New-NAVServerInstance cmdlet](#).

Use the *DatabaseCredentials* parameter to provide the login credentials of the database user that you want to use to access the application database.

2. To configure SQL Authentication with the tenant database, run the [Mount-NAVTenant cmdlet](#).

Use the *DatabaseCredentials* parameter to provide the login credentials of the database user that you want to use to access the tenant database.

See Also

[Installation Considerations for Microsoft SQL Server](#)

[Deployment](#)

[Installing Business Central Using Setup](#)

Creating and Altering Business Central Databases

3/31/2019 • 11 minutes to read

You can create new Business Central databases in the Dynamics NAV Development Environment and by using the [New-NAVDatabase](#) cmdlet in the Dynamics NAV Development Shell.

When you create a database you must specify the SQL Server instance for the database and the authentication type.

Create a database

1. In the Dynamics NAV Development Environment, on the **File** menu, choose **Database**, and then choose **New**.
2. In the **Server Name** field, enter the name of the SQL Server instance. You can choose the up arrow to select the server from a list of available servers or you can enter the server name manually.
3. In the **Authentication** field, select the type of authentication that you require. Choose the drop-down arrow to select **Database Server Authentication** or **Windows Authentication**.
 - If you select **Database Server Authentication**, then authentication is performed by the SQL Server instance that you have selected.
 - If you select **Windows Authentication**, then authentication is performed by the Windows domain controller.
4. In the **User ID** field, enter your User ID if you have selected **Database Server Authentication**.
5. In the **Password** field, enter your password if you have selected **Database Server Authentication**.
6. To set the network type to be used when connecting to the server, choose the **Advanced** tab and select the net type from the drop down list box in the **Net Type** field. However, it is not usually necessary to change the network type from the default setting. The **Default** net type setting allows Business Central to connect to a server using the default client network type assigned by SQL Server. You can change the net type with the Client Network Utility, which is part of the SQL Server Client Utilities, if they have been installed on the client computer.
7. Choose **OK** to connect to the server and open the **New Database** window.

In the **New Database** window, enter the information about the database that you want to create. The window contains the same tabs as the **Alter Database** window. For more information, see sections in this article for the different tabs.
8. Now that you have created a new database, you must configure your Business Central Server instance to access the database and then restart the service. For more information, see [Configuring a Business Central Server Instance](#).
9. You must synchronize the schema for all tables of the new database.

From the development environment, on the **Tools** menu, choose **Sync. Schema For All Tables**, and then **With Validation**.

You can also use the Sync-NAVTenant cmdlet of the Business Central Administration Shell.

WARNING

You can always enlarge a database later on, but you cannot make it smaller.

After you have created the database, you can enter program objects and company data. Before you can create company data, you must import some basic data from another Business Central database. The imported data must at least include **Data Common to All Companies** and **Application Objects**.

Alter a Database

The changes will not take effect until you restart the Dynamics NAV Server instance.

NOTE

You cannot alter a database by using the development environment if the database is deployed on Azure SQL Database.

Database Files Tab

Increases the size of the database by either increasing the size of one or more of the database files or adding new data files to the database.

If you use secondary data files, then you must increase the size of the primary data file only when the catalog that it contains has become too large. When the catalog has become too large, new SQL Server objects, such as tables, cannot be created until you increase the size of the primary data file.

When you use secondary data files, you cannot create more space for storing Dynamics NAV data by just increasing the size of the primary data file. You can create more space for storing data by increasing the size of the secondary data files that contain Dynamics NAV information. You can also add new secondary data files in order to store more data.

To open this window, on the **File** menu, choose **Database**, choose **Alter**, and then choose the **Database Files** tab.

NOTE

The first data file that is listed on the **Database Files** tab is the primary file.

Transaction Log Files Tab

Increases the size of the existing transaction log files or adds new files to enable more transactions to be performed in the database. The transaction log grows as new transactions are performed in the database. SQL Server truncates the log after it performs a successful database or transaction log backup.

To open this window, on the **File** menu, choose **Database**, choose **Alter**, and then choose the **Transaction Log Files** tab.

You can also delete existing transaction log files that are empty. The first transaction log file that is listed is the primary file. You cannot delete the primary transaction log file.

Collation Tab

Changes the collation that is used by the database.

Before you change the collation, you have to select the **Single user** option on the **Options** tab.

Changing the Collation of a Dynamics NAV Database

IMPORTANT

You cannot change the collation directly in an existing database. To change the collation, you must create a new database that uses the correct collation, and then export the data from the old database and import it to the new database. For more information, see [Changing Collation of Existing Database](#).

If you change the database collation, then the collation of objects in the database is changed except for tables that have the **LinkedObject** property set to **Yes**. You must manually re-create these objects. For example, you can script them in SQL Server Management Studio.

If you change the collation from a case-sensitive to a case-insensitive collation or from an accent-sensitive to an accent-insensitive collation, then duplicates can occur in the primary keys of the tables. Duplicates can be caused by the values of the character data stored in the primary keys. If duplicates occur, then you receive an error message and the database collation change is stopped. We recommend that you do not change these attributes of a collation.

NOTE

Changing the collation can be a lengthy process that depends on the size of the database and the number of companies in the database. The system tables and all user table indexes that contain character data must be rebuilt.

The **Language** drop-down list displays the friendly name of the language, not the full Windows collation name. For some languages, there are multiple collations that sort characters differently. For example, the Windows collation languages include multiple Scandinavian languages, some of which sort Aa after Z, Æ, Ø, and some of which sort Aa after A and before B. If you upgrade from Microsoft Dynamics NAV 2009 to Business Central, you upgrade the database to the Windows collations. If you used SQL collation in earlier versions of Dynamics NAV, then after you upgrade, verify that the Windows collation sorts characters in the way that you expect.

If you set the **Validate Collation** check box, then collation languages that run with a different non-Unicode code page from your system non-Unicode code page are filtered out of the **Language** drop-down list. An example scenario of when you might want to choose a collation language that has a different code page from your system code page is if you want to prepare a Japanese database on a Danish computer.

Options Tab

Specifies database options that you set when you created the database. For example, you must select the **Single User** option before you perform any database tests. You must clear this option when the tests are completed.

To open this window, on the **File** menu, choose **Database**, choose **Alter**, and then choose the **Options** tab.

Access Section

FIELD	DESCRIPTION
Single user	<p>Specifies that only one user can access the database at a time. You can use this setting when you are performing administrative functions such as testing or restoring the database. By limiting access to the database to one user, you make sure that the database is not changed when you are testing it.</p> <p>Important: Clear this check box when you are finished to give other users to access the database.</p>

Settings Section

FIELD	DESCRIPTION
Recovery Model	<p>Determines the kind of information that is written to the transaction log and therefore the kind of recovery model that you want to use in this database.</p> <p>Note: The Full and Bulk-logged recovery models are similar, and many users of the Full recovery model will use the Bulk-logged recovery model occasionally.</p> <p>Option: Bulk-logged</p> <p>Description:</p> <p>The transaction log will contain only limited information about certain large-scale or bulk copy operations. The Bulk-logged recovery model provides protection against media failure combined with the best performance and the minimal use of log space for certain large-scale or bulk copy operations.</p> <p>The backup strategy for bulk-logged recovery consists of:</p> <ul style="list-style-type: none">* Database backups.* Differential backups (optional). <p>Option: Full</p> <p>Description:</p> <p>The details of every transaction are stored in the transaction log. This information can be used when you apply transaction log backups. The Full recovery model uses database backups and transaction log backups to provide complete protection against media failure. If one or more data files are damaged, media recovery can restore all the committed transactions. Incomplete transactions are rolled back.</p> <p>Full recovery lets you recover the database to the point of failure or to a specific point in time. All operations are fully logged to guarantee that the database is recoverable. This includes bulk operations such as SELECT INTO, CREATE INDEX, and bulk loading data.</p> <p>The backup strategy for full recovery consists of:</p> <ul style="list-style-type: none">* Database backups.* Differential backups (optional).* Transaction log backups. <p>Option: Simple</p> <p>Description:</p> <p>The database can be recovered to the point at which the last backup was made. However, you cannot restore the database to the point of failure or to a specific point in time. To do that, select either the Full or Bulk-logged recovery model.</p>

FIELD	DESCRIPTION
	<p>The backup strategy for simple recovery consists of:</p> <ul style="list-style-type: none"> * Database backups. * Differential backups (optional).
ANSI NULL default	Specifies whether the database default NULL settings for column definitions and user-defined data types are to be applied. When you select this option, all user-defined data types or columns that have not been explicitly defined as NOT NULL are set to allow NULL entries. Columns that have been defined by using constraints follow the constraint rules, regardless of this setting.
Recursive triggers	Specifies recursive trigger settings. Triggers can have direct recursion or indirect recursion. Direct recursion occurs when a trigger occurs and performs an action that causes the same trigger to be fired again. Indirect recursion occurs when a trigger occurs and performs an action that causes a trigger on another table to occur. This second trigger updates the original table which causes the first trigger to occur again.
Torn page detection	Enables SQL Server to detect incomplete input/output operations that have been caused by power failures or other system outages.
Auto shrink	Specifies whether SQL Server can periodically shrink data files and transaction log files.

Integration Tab

Specifies database settings that determine how Dynamics NAV integrates with SQL Server and external tools.

To open this window, on the **File** menu, choose **Database**, choose **Alter**, and then choose the **Integration** tab.

Objects Options

FIELD	DESCRIPTION
Convert Identifiers	<p>Defines characters that you want to map to the underscore character in the names of all SQL Server objects, such as tables, columns, and constraints. If these characters occur in tables or fields in Dynamics NAV, then they are converted to underscores in the SQL Server names.</p> <p>When the conversion is complete, you must close and reopen the database before you can use the new identifiers.</p>

License Options

FIELD	DESCRIPTION
Save license in database	Specifies that the license file is uploaded and stored in the database instead of on the server. This is useful if you are hosting several databases with separate license files on the same server.

Advanced Tab

Specifies how locking is handled in the database and specifies the start ID for elements on new objects.

Locking Options

FIELD	DESCRIPTION
Lock timeout	<p>Specifies whether a session waits to place a lock on a resource that has already been locked by another session.</p> <p>If you clear this field, then the session waits indefinitely.</p>
Timeout duration (sec)	<p>Specifies the maximum length of time that a session waits to place a lock on a resource that has already been locked by another session. The default value is 10 seconds.</p>

Designer Options

FIELD	DESCRIPTION
Start ID (UidOffset)	<p>Specifies the start ID for elements on new objects. When you create a new table, page, report, codeunit, query, or XMLport, the elements have IDs that are offset by the Start ID (UidOffset) value that you specify. Object elements include containers, groups, fields, parts, Dataltems, columns, filters, variables, functions, or text constants.</p> <p>You must specify an Integer that is greater than or equal to 0.</p> <p>The default value is 1.</p>

Deploy a Business Central Database to Azure SQL Database

4/10/2019 • 4 minutes to read

This topic describes how you can deploy a Business Central database to Microsoft Azure SQL Database.

To deploy a Business Central database to Azure SQL Database, the database must be exported as a data-tier application (DAC) file, which is known as a .bacpac file. This can be performed by using SQL Server Manager, as described in this topic.

IMPORTANT

To optimize, we recommend that the Business Central Server instance that connects to the database is deployed on a virtual machine in Azure. Additionally, the virtual machine and SQL Database must be in the same Azure region.

Prerequisites

Make sure that you have the following prerequisites for completing this procedure:

- A Microsoft Azure subscription and access to the Azure Portal.
- A Business Central database installed on a SQL Server Database Engine instance.
- SQL Server Manager is also installed on the same computer.
- Access to the Business Central installation media (DVD).

Create and configure an Azure SQL Database Server

In the Azure Portal, create an SQL Database Server for hosting the Business Central database. For more information about how to create and configure an SQL Database server, see [Create your first Azure SQL Database](#).

Here are some important notes when creating the Azure SQL Database:

1. You must specify a login name and password for the server. You will use this information in the next steps when you deploy the Business Central database to Azure SQL and set up the Business Central Server to authenticate with the database.
2. Configure the server to allow for access by Windows Azure Services.
3. Make a note of the SQL Database server name because you will need it later.

The name has a format similar to this: `mysqldatabaseserver.database.windows.net`.

4. Configure the SQL database server firewall to allow for access by the IP address of the computer that you are using to deploy the Business Central database.

For information, see [How to: Configure Firewall Settings \(Azure SQL Database\)](#).

Prepare the Business Central Database

Make sure the database meets these requirements:

1. Delete all users of the database that use Windows authentication.

This includes `NT AUTHORITY\NETWORK SERVICE` and `NT AUTHORITY\SYSTEM`. Only users with SQL authentication are allowed in Azure SQL Database..

2. Upload a valid Business Central license file to the database.

For more information, see [Uploading a License File for a Specific Database](#).

3. Delete the deadlock monitors for the Business Central database.

You can do this in SQL Server Management Studio by running a query similar to the following:

```
use [Demo Database BC (14-0)]

DROP VIEW [dbo].[deadlock_report_ring_buffer_view]
```

For more information about the deadlock monitor, see [Monitoring SQL Database Deadlocks](#).

Export Business Central Database to a BACPAC File (.zip file)

When you deploy your application online, you must provide a compressed .zip file that contains the database as data-tier application file, known as BACPAC (.bacpac) file. This article describes how you to create the BACPAC files and zip. You can do this using SQL Server Management Studio.

1. In SQL Server Management Studio, connect to the server instance that hosts the database.
2. In **Object Explorer**, right-click either the database, choose **Task**, and then choose **Export Data-tier Application**.
3. Follow the steps in the **Export Data-tier Application** wizard to export the database to a .bacpac file on your computer or network.

You can use any name for the .bacpac file. For more information about exporting databases to .bacpac format, see [Export a Data-tier Application](#).

Import the BACPAC to Azure SQL

1. In SQL Server Management Studio, connect to Azure SQL Database server that you created.
 - a. Select **File** > **Connect Object Explorer**.
 - b. In the **Server Name**, enter the server name assigned to your Azure SQL Database server.

For example, `*mysqldatabaseserver.database.windows.net`. You can get this from the Overview page in the Azure portal.
 - c. Enter the login name and password that you set up in the first task when creating the Azure SQL Database server.
2. Import the BACPAC file that you created for the Business Central Database.
 - a. In **Object Explorer**, right-click the **Database** folder, and select **Import Data-tier Application**.
 - b. Follow the wizard. On the **Import Settings** page, browse for the BACPAC that you create, and choose **Next**.
 - c. Review the **Database Settings** page. Make changes if needed, and then choose the **Next** > **Finish**.

Configure the SQL Server Authentication on the Business Central

Server instance

The last step is configure SQL Server Authentication on the Business Central Server instance. For the database credentials, use the login name and password that set up when you created the Azure SQL Database server.

For more information, see [Configuring SQL Server Authentication](#).

The Business Central database is now deployed and configured on Azure. For developing, you can connect to the database from the Dynamics NAV Development Environment.

Changing database login account database

If you want to use a different login account for the database, to the following:

1. Create a new login that uses SQL Server authentication.

For more information, see [Create a Login](#).

2. Map the login to a user in the Business Central database, and add the user to the **db_owner** role of the Business Central database.

For more information, see [Create a Database User](#).

See Also

[Installation Considerations for Microsoft SQL Server](#)

[Optimizing SQL Server Performance](#)

()

Administration of Business Central Online

6/25/2019 • 2 minutes to read

As a Business Central reselling partner, you are the administrator of the Business Central tenants of your customers. You have access to the administration tools of their Office 365 account and their Business Central administration center where you can specify update windows, for example.

Get started with the Cloud Solution provider program

You must enroll in the Cloud Solution Provider program in order to service Business Central online. In the Microsoft Partner Center documentation, you can learn how to [add a customer](#), [assign licenses to users](#), and [create new subscriptions](#). Business Central is one of the subscriptions that you can create, and there are Business Central-specific license types that you can assign to users.

Extending trials

A prospect can sign up for a free trial of Business Central. When they first sign up for Business Central, they get access to an evaluation version that does not include all capabilities in Business Central. If they then enable the 30 day trial experience, this enables all capabilities. However, sometimes 30 days is not quite enough to decide if they want to buy Business Central. In that case, they can extend their trial with an additional 30 days. For more information, see [Need More Time to Decide Whether to Subscribe?](#).

If the prospect wants to extend the trial further than those 30 days, you can extend it another 30 days if you, as the delegated administrator, log into their Business Central and extend the trial using the same **Extend Trial Period** guide. However, after those additional 30 days, the prospect must either purchase Business Central, or you can ask Microsoft for an additional extension of the trial by contacting Support.

TIP

You can suggest your prospects sign up for a trial, but you can also set up a customized demonstration environment based on your sandbox environment. This way, you can easily add or remove functionality based on your prospects' expectations. For more information, see [Choosing Your Dynamics 365 Business Central Development Sandbox Environment](#).

See Also

[The Business Central Administration Center](#)

[The Business Central Administration Center API](#)

[Resources for Help and Support for Dynamics 365 Business Central](#)

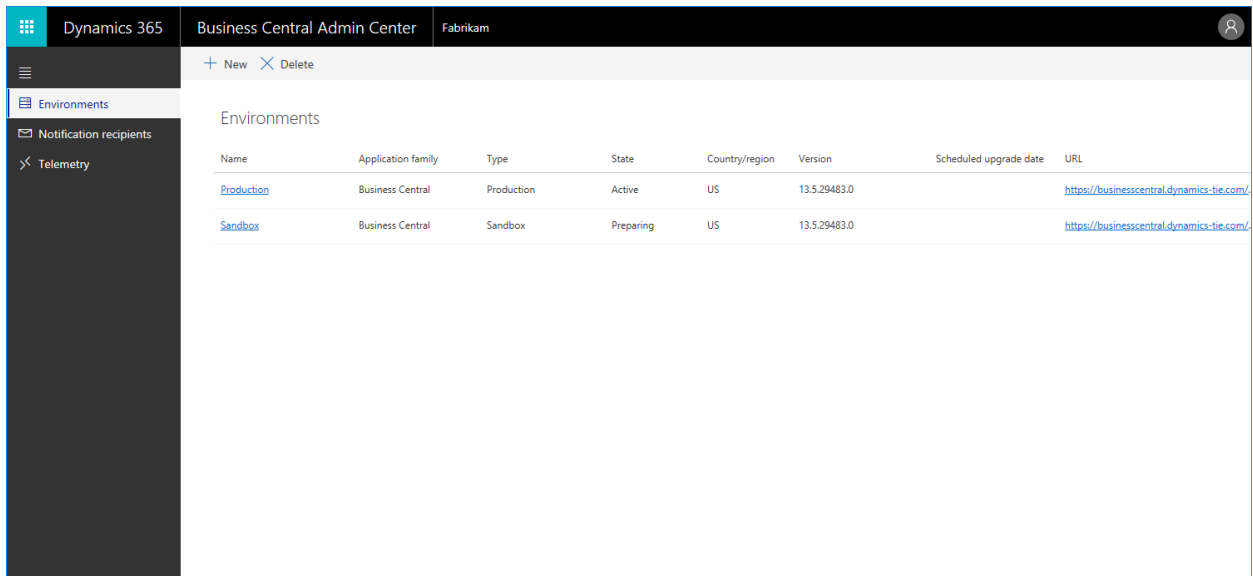
[Resell Different Solutions](#)

[Deliver consulting services as a VAR: aka.ms/BusinessCentralConsultingServices](#)

The Business Central Administration Center

6/25/2019 • 2 minutes to read

The Business Central administration center provides a portal for administrators to perform administrative tasks for a Business Central tenant. Here, administrators can view and work with production and sandbox environments for the tenant, set up update notifications, and view telemetry for events on the tenant.



Accessing the administration center

The following users are authorized to access the Business Central administration center:

- Internal tenant administrators
- Admin agent
- Helpdesk agent

The admin agent and helpdesk agent roles are assigned through the [Microsoft Partner Center](#) for the partner associated with the tenant. These roles are delegated administrators on the Business Central tenant.

As a partner, you can access the center from the Partner Dashboard in the Microsoft Partner Center:

1. Log into the [Partner Dashboard](#).
2. Select the **Customers** link in the navigation pane.
3. Select the customer tenant that you want to perform administrative tasks for.
4. Select **Service Management**.
5. Under the **Administer Services** heading, select Dynamics 365 Business Central.

You can also get to the administration center by navigating directly to the URL of a tenant's instance. This is done with the following URL, if you replace `[TENANT_ID]` with the tenant ID of the tenant.

```
https://businesscentral.dynamics.com/[TENANT_ID]/admin
```

See also

[Working with Administration Tools](#)

[Managing Environments](#)

[Tenant Notifications](#)

[Environment Telemetry](#)

[Administration Center API](#)

[Business Central Data Security](#)

[Introduction to automation APIs](#)

[Microsoft Partner Dashboard](#)

[Add a new customer in the Partner Center](#)

[Assign licenses to users in the Partner Center](#)

[Create new subscriptions in the Partner Center](#)

[Cloud Solution Provider program - selling in-demand cloud solutions](#)

Managing Environments

6/25/2019 • 5 minutes to read

The **Environments** tab of the Business Central administration center allows you to view information about the Business Central production and sandbox environments for the tenant, as well as manage updates for each environment.

The Environments list page provides an overview of the production and sandbox environments in the tenant. You can see the details and additional actions for an environment by clicking the link in the **Name** column of the list for the environment.

Create a new production environment

The Business Central administration center provides an easy method for creating the production environments for the tenant.

NOTE

Each Business Central tenant is limited to one production environment.

To create a production environment:

1. On the **Environments** tab of the Business Central administration center, select the **New** action on the action ribbon.
2. In the **Create Environment** pane, select **Production** in the **Environment Type** list.
3. Select **Create**.

When the new production environment is created, it will be on the latest production version of Business Central.

Create a new sandbox environment

A sandbox environment is a non-production instance of Business Central. Isolated from production, a sandbox environment is the place to safely explore, learn, demo, develop, and test the service without the risk of affecting the data and settings of your production environment.

NOTE

Each Business Central tenant is limited to three sandbox environments.

To create a sandbox environment:

1. On the **Environments** tab of the Business Central administration center, select the **New** action on the action ribbon.
2. In the **Create Environment** pane, provide a name for your environment.
3. Select **Sandbox** in the **Environment Type** list.
4. Specify if you want the sandbox environment to contain a copy of another environment. If this option is selected, select the environment to copy.

NOTE

When creating a sandbox environment as a copy of another environment, the new environment is created on the same application version as the environment being copied. The new environment will also contain all the per-tenant extensions and AppSource extensions installed and published in the environment being copied.

5. Select the application version for the new sandbox environment from the **Version** list.
6. Select **Create**.

NOTE

The sandbox environment will not be accessible until the **State** shows *Active*.

A single, default sandbox environment can also be created from within a page in the production environment of the Business Central application. For more information, see [How to: Create a Sandbox Environment](#).

To delete a sandbox environment, choose the environment on the **Environments** tab of the Business Central administration center, and then Select **Delete** on the action ribbon.

Precautions for sandbox environments with production data

If a sandbox is created with a copy of a production environment, a number of precautions are taken for that sandbox:

- The job queue is automatically stopped.
- Any base application integration settings are cleared.
- Outbound HTTP calls from extensions are blocked by default and must be approved per extension.
- Any General Data Protection Regulation (GDPR) action must be handled separately and repeated for the sandbox. There is no synchronization with the production environment after the sandbox has been created.

To enable outbound HTTP calls, go to the **Extension Management** page in Business Central, and choose **Configure**. Then, on the **Extension Settings** page, make sure that **Allow HttpClient Requests** is turned on. This setting must be enabled per extension.

Selecting a version for a new sandbox environment

If, when creating a new sandbox environment, the environment is not a copy of an existing environment, you have the option to select the application version for the new environment. The version list will show the latest Production version, which is the version used for new production environments.

The version list may also have one or more Preview versions. Preview versions are early release candidates of upcoming releases of Business Central that are made available specifically for sandbox environments. This allows for reviewing new functionality, validating extension compatibility, and other general testing of the upcoming release.

When a sandbox environment is created on a Preview version, the environment will automatically be updated to new Preview versions when they become available. However, the environment will not be updated to the Production version. Once a sandbox environment is on a Preview version, it must stay on a Preview version until it is deleted. The environment may also be deleted if an update between Preview versions fails. It is recommended that Preview versions be used only for temporary testing of an upcoming release.

Updating environments

Business Central environments are updated on a monthly cadence. Major updates occur semiannually, and minor updates occur each month that there is not a major update. The Business Central administration center gives you a level of control over the timing of updates for each environment.

Set the update window

The update window for an environment defines a window of time during the day in which the environment can be updated. When an update is rolling out to Business Central online, regardless of whether it's the monthly service update or a major update, the update will be applied to an environment within the time frame that the update window defines. This helps to ensure that updates are applied outside of the customer's normal business hours, for example.

NOTE

Desktop users who are signed in during the update will receive an alert in Business Central.

To set the update window for an environment:

1. On the **Environments** tab of the Business Central administration center, Select the **Name** of the relevant environment to open the environment details.
2. Select the **Set update window** action on the **Update** list on the action ribbon.
3. In the **Set update window** pane, specify the start time and the end time for the update window for the environment.

NOTE

The update window must be a minimum of six hours.

4. Select **Save**.

Schedule an update date

For major updates only, you have the option to select a specific date on which the environment is updated. When a major update version is available, a notification is sent to the notification recipients listed on the **Notification recipients** tab of the Business Central administration center (See [Managing Tenant Notifications](#) for more information). The **Update version** field in the **Version Management** section of the environment details also displays the version number of the available update version.

To schedule an update date:

1. On the **Environments** tab of the Business Central administration center, Select the **Name** of the relevant environment to open the environment details.
2. Select the **Schedule Update** action on the **Update** list on the action ribbon.
3. In the **Schedule Environment Update** pane, select the desired update date.

NOTE

The selected date must be within a given date range displayed in the pane.

4. Select **Schedule Update**.

See also

[Working with Administration Tools](#)

[The Business Central Administration Center](#)

[Managing Environments](#)

[Managing Tenant Notifications](#)

[Introduction to automation APIs](#)

Tenant Notifications

6/25/2019 • 2 minutes to read

Notifications are sent by email for administrative events that occur on the tenant. For example, notifications are sent when a major update is available for tenant environments, when an environment update has succeeded or failed, or when extensions require changes to be compatible with an upcoming release. When these and other similar events occur on the tenant, an email is sent to the notification recipients for the tenant.

Notification recipients

Notifications are sent to all email addresses that are listed in the **Notification recipients** list of the Business Central administration center. The list is managed manually by adding and removing recipients to ensure the right individuals are notified of the event.

NOTE

It is important that *at least* one administrator's email address has been entered as a notification recipient to ensure proper awareness of events requiring administrative attention.

See also

[Working with Administration Tools](#)

[The Business Central Administration Center](#)

[Managing Environments](#)

[Managing Tenant Notifications](#)

[Introduction to automation APIs](#)

Environment Telemetry

3/27/2019 • 2 minutes to read

The Business Central administration center provides telemetry for the tenant environments to enable troubleshooting and support for the tenant. The Telemetry tab provides telemetry of top-level AL events, and any errors resulting from calls through the telemetry stack.

To filter the telemetry for an environment:

1. Select a base point-in-time for the timestamp of the telemetry messages.
2. Enter a number of minutes before or after the base point-in-time to set a range of time for the timestamp. A negative number indicates a number of minutes before the base point-in-time, and a positive number indicates a number of minutes following the base point-in-time. For example, a value of -15 will filter the telemetry messages to a timestamp range of up to 15 minutes before the base point-in-time.
3. Choose the message type.
4. Choose the environment.
5. Select **Filter**.

See also

[Working with Administration Tools](#)

[The Business Central Administration Center](#)

[Managing Environments](#)

[Managing Tenant Notifications](#)

[Introduction to automation APIs](#)

The Business Central Administration Center API

6/25/2019 • 9 minutes to read

The Business Central administration center API enables administrators to programmatically perform administrative tasks for a Business Central tenant. Using the API, administrators can query and work with production and sandbox environments for the tenant, set up administrative notifications, and view telemetry for events on the tenant.

See [The Business Central Administration Center](#) for more details on administrative capabilities. This article describes the API contracts for these administrative capabilities.

Location

The Business Central administration center API is located at the following URL:

<https://api.businesscentral.dynamics.com>.

Setting up Azure Active Directory (AAD) based authentication

Sign in to the [Azure Portal](#) to register your client application as an app and enable it to call the Business Central administration center API.

1. Follow the instructions in the [Integrating applications with Azure Active Directory](#) article. The next steps elaborate on some of the specific settings you must enable.
2. Give the application a **Name**, such as **Business Central Web Service Client**.
3. For **Application type**, choose either **Native** or **Web app/API** depending on your scenario. The code examples below assume **Native**.
4. Choose a **Redirect URI**. In the case of a **Native** app, you can choose for example: **BusinessCentralWebServiceClient://auth**. In the case of a **Web app/API** app, set the value to the actual URL of the web application.
5. During the registration of the app, make sure to go to **Settings**, and then under **API ACCESS**, choose **Required permissions**. Choose **Add**, and then under **Add API Access**, choose **Select an API** and search for the **Dynamics 365** option. Choose **Dynamics 365**, select **Delegated permissions**, and then choose the **Done** button.

NOTE

If **Dynamics 365** does not show up in search, it's because the tenant does not have any knowledge of Dynamics 365. To make it visible, an easy way is to register for a [free trial](#) for Dynamics 365 Business Central with a user from the directory.

6. Make a note of both the **Application ID** and the **Redirect URI**. They will be needed later.

Getting an access token

HTTP requests sent to the Business Central administration center API must include the Authorization HTTP header, and the value must be an access token.

The following examples show how to obtain such a token using PowerShell. Using C# is straightforward.

Powershell example without prompt:

```
$cred = [Microsoft.IdentityModel.Clients.ActiveDirectory.UserPasswordCredential]::new($UserName, $Password)
$ctx =
[Microsoft.IdentityModel.Clients.ActiveDirectory.AuthenticationContext]::new("https://login.windows.net/$TenantName")
$token =
[Microsoft.IdentityModel.Clients.ActiveDirectory.AuthenticationContextIntegratedAuthExtensions]::AcquireTokenAsync($ctx, "https://api.businesscentral.dynamics.com", <Application ID>,
$cred).GetAwaiter().GetResult().AccessToken
```

Powershell example with prompt:

```
$ctx =
[Microsoft.IdentityModel.Clients.ActiveDirectory.AuthenticationContext]::new("https://login.windows.net/$tenantName")
$redirectUri = New-Object -TypeName System.Uri -ArgumentList <Redirect URL>
$platformParameters = New-Object -TypeName
Microsoft.IdentityModel.Clients.ActiveDirectory.PlatformParameters -ArgumentList
([Microsoft.IdentityModel.Clients.ActiveDirectory.PromptBehavior]::Always)
$token = $ctx.AcquireTokenAsync("https://api.businesscentral.dynamics.com", <Application ID>, $redirectUri,
$platformParameters).GetAwaiter().GetResult().AccessToken
```

Environments

Environments are the instances of the application that have been set up for the tenant. An instance can be of either a production type or a sandbox type. Currently, there can be one production environment and up to three sandbox environments per tenant. The environment APIs can be used to get information about the environments currently set up for the tenant, create a new sandbox environment using sample data or as a copy of the production environment, and delete the sandbox environment.

Get environments and Get environments by application family

Returns a list of all the environments for the tenant.

```
[200] GET /v1.2/admin/environments
```

Returns a list of the environments for the specified application family.

```
[200] GET /v1.2/admin/applications/{applicationFamily}/environments
```

Response:

Returns a wrapped array of environments.

```
{
  "value":
  [
    {
      "friendlyName": string, // Display name of the environment
      "type": string, // Environment type (e.g. "Sandbox", "Production")
      "name": string, // Environment name, unique within an application type
      "countryCode": string, // Country/Region the environment is deployed in
      "applicationFamily": string,
      "aadTenantId": Guid,
      "applicationVersion": string,
      "status": string (enum | "NotReady", "Removing", "Preparing", "Active"),
      "webClientLoginUrl": string // Url to use to log into the environment
    }
  ]
}
```

Get environment by application family and name

Returns the properties for the provided environment name if it exists.

```
[200] GET /v1.2/admin/applications/{applicationFamily}/environments/{environmentName}
```

Response:

Returns a single environment if exists.

```
{
  "friendlyName": string, // Display name of the environment
  "type": string, // Environment type (e.g. "Sandbox", "Production")
  "name": string, // Environment name, unique within an application type
  "countryCode": string, // Country/Region the environment is deployed in
  "applicationFamily": string,
  "aadTenantId": Guid,
  "applicationVersion": string,
  "status": string (enum | "NotReady", "Removing", "Preparing", "Active"),
  "webClientLoginUrl": string // Url to use to log into the environment
}
```

Create environment

Creates a new sandbox environment with sample data.

```
[201] PUT /v1.2/admin/applications/{applicationFamily}/environments/{environmentName | "Sandbox" or "Production"}
```

or

```
[201] PUT
/v1.2/admin/applications/{applicationFamily}/environments/{environmentName}/{applicationVersion}/{ringName | "PROD" or "PREVIEW"}
```

- See the section below about Available Versions for information about versions currently supported.

Body

```
{
  "type": "Sandbox"
}
```

Response:

Returns newly created environment.

```
{
  "type": string, // Environment type
  "name": string, // Environment name, unique within an application type
  "applicationFamily": string,
  "aadTenantId": Guid,
  "status": string (enum | "NotReady", "Removing", "Preparing", "Active")
}
```

Path parameters

- **applicationFamily**: Family of the application as is.

- **environmentName**: Free field; but, currently only "sandbox" is used. Stored case sensitive, indexed case insensitive.

Errors

```
{
  {
    "code": string, // e.g. "DestinationApplicationServiceNotFound"
    "message": string, // e.g. "A suitable destination service cannot be found"
    "target": string, // Such as a field name, where the error is occurred
    "clientError": [
      {
        "code": string,
        "message": string,
        "target": string
        "clientError": [...]
      }
    ]

    "innerError": {
      "code": string,
      "innerError": [...]
    }
  }
}
```

Copy environment

Creates a new sandbox environment with a copy of the production environment's data.

```
[201] PUT /v1.2/admin/applications/{applicationFamily}/environments/{environmentName}
```

Body

```
{
  "copyFromEnvironmentName": "Production",
  "type": "Sandbox"
}
```

Delete environment

Deletes the specified environment. Warning: A production environment should not be deleted.

```
[202] DELETE /v1.2/admin/applications/{applicationFamily}/environments/{environmentName}
```

Available Versions

Get information about the currently supported versions.

Versions and Corresponding Rings

```
[200] GET /v1.2/admin/applications/{applicationFamily}/rings
```

Response:

```
{
  "value": [
    {
      "applicationVersion": {
        "major": int,
        "minor": int,
        "build": int,
        "revision": int
      },
      "ringName": string,
      "ringFriendlyName": string
    }
  ]
}
```

Updates

The update settings allow you to specify an update window for the time of day an update can be performed on the tenant environment. The update window must be a minimum of six hours. (e.g. 1:00 - 7:00)

Get Update Settings

Returns the update settings for the environment. The update settings currently available are the start and end times for the update window.

```
[200] GET v1.2/admin/applications/{applicationFamily}/environments/{environmentName}/settings/update
```

Response:

Returns the environment's update settings, or "null" if not exists

```
{
  "preferredStartTimeUtc": datetime, // Start of environment update window
  "preferredEndTimeUtc": datetime, // End of environment update window
}
```

NOTE

The `date` components of the values are ignored, only the time components are used.

Put Update Settings

Set the update window start and end times.

```
[200] PUT v1.2/admin/applications/{applicationFamily}/environments/{environmentName}/settings/update
```

Body

```
{
  "preferredStartTimeUtc": datetime, // Start of environment update window
  "preferredEndTimeUtc": datetime, // End of environment update window
}
```

Response:

Returns the updated settings

```
{
  "preferredStartTimeUtc": datetime, // Start of environment update window
  "preferredEndTimeUtc": datetime, // End of environment update window
}
```

NOTE

The `date` components of the values are ignored, only the time components are used.

Telemetry

Telemetry includes the top-level AL events and any returned errors logged from the service. These events can provide necessary information and errors that can be used to troubleshoot issues happening in the tenant's environment.

Get Environment Telemetry

Returns the telemetry information for the provided environment and filters. It is recommended that you provide start and end time parameters in order to return a manageable data set.

```
[200] GET v1.2/admin/applications/{applicationFamily}/environments/{environmentName}/telemetry?startDateUtc={start}&endDateUtc={end}&logCategory={cat}
```

Query parameters:

start: datetime // The start of the telemetry entry time window to query

end: datetime // The end of the telemetry entry time window to query

cat: (All or 0) // Category of telemetry to query

Response:

Returns the telemetry logs and with data column headers.

```
{
  "queryColumns": [
    {
      "name": string, // Column display name
      "ordinal": int, // Index of the column in the data set
      "dataType": ( string or 0 | numeric or 1 | datetime or 2 )
      "expectations": (none or 0 | wide or 1) // Flags enum value
    }
  ],
  "queryResults": object[][] // Raw query data results
}
```

Notifications

Notifications are sent to the recipient email addresses set up for the tenant. For example, notifications are sent for update availability, successful updates, update failures, and extension validations.

Get Notification Recipients

Returns a list of notification recipients.

```
[200] GET /v1.2/admin/settings/notification/recipients
```

Response:

Returns a wrapped array of recipients.

```
{
  "value":
  [
    {
      "id": GUID, // Unique identifier of the recipient
      "email": string, // Email address of the recipient
      "name": string // Full name of the recipient
    }
  ]
}
```

Create Notification Recipient

Create a new notification recipient.

```
[200] PUT /v1.2/admin/settings/notification/recipients
```

Body

```
{
  "email": string, // Email address of the recipient
  "name": string // Full name of the recipient
}
```

Response:

Returns the newly created recipient.

```
{
  "id": GUID, // Unique identifier of the recipient
  "email": string, // Email address of the recipient
  "name": string // Full name of the recipient
}
```

Update Notification Recipient

Modify an existing notification recipient.

```
[200] PUT /v1.2/admin/settings/notification/recipients
```

Body

```
{
  "id": GUID, // Unique identifier of the recipient
  "email": string, // Email address of the recipient
  "name": string // Full name of the recipient
}
```

Response:

Returns the updated recipient.

```
{
  "id": GUID, // Unique identifier of the recipient
  "email": string, // Email address of the recipient
  "name": string // Full name of the recipient
}
```

Delete Notification Recipient

Deletes an existing notification recipient.

```
[200] DELETE /v1.2/admin/settings/notification/recipients/{id}
```

Get Notification Settings

Returns the properties of the specified notification recipient.

```
[200] GET /v1.2/admin/settings/notification
```

Response:

Returns the notification settings.

```
{
  "aadTenantId": GUID, // AAD Tenant ID of the caller
  "recipients": [
    {
      "id": GUID, // Unique identifier of the recipient
      "email": string, // Email address of the recipient
      "name": string // Full name of the recipient
    }
  ]
}
```

Application Access Management

It is possible for a **Delegated Tenant Admin** to manage access to application families available in the service. The application family is Business Central or other independent software vendor (ISV) applications that may be provisioned through the service.

You can get the list of applications that are available to the tenant. From this list you can determine, by setting the access property, for which applications an environment may be provisioned on the tenant.

Get List Of Manageable Applications

Returns a list of manageable applications.

```
[200] GET /v1.2/admin/manageableapplications
```

Response:

Returns a wrapped array of applications.

```
{
  "value": [
    {
      "applicationFamily": string,
      "access": boolean
    }
  ]
}
```

Control the access to Applications

Pass the application family name in the URL and a boolean in the body.

- True - enables the access.
- False - disables the access.

```
[200] PUT /v1.2/admin/manageableapplications/{applicationFamily}
```

Body

```
{
  boolean // Desired access state
}
```

NOTE

It is only possible to disable the access to applications for the AAD tenant if it does not have **application tenant** yet.

Get List Of Accessible Applications

Tenant Admin can obtain a list of accessible applications.

```
[200] GET /v1.2/admin/accessibleapplications
```

Response:

Returns a wrapped array of applications.

```
{
  "value": [
    {
      "applicationFamily": string,
      "access": boolean
    }
  ]
}
```

Reschedule Updates

Some environment updates are allowed to be rescheduled.

Get Scheduled Update

Get information about updates that have already been scheduled for a specific environment.

```
[200] GET /v1.2/admin/applications/{applicationFamily}/environments/{environmentName}/update
```

Response:

Returns information about the update job for that environment.

```
{
  "jobId": int,
  "jobEntryId": int,
  "sourceVersion": string,
  "targetVersion": string,
  "canTenantSelectDate": boolean,
  "didTenantSelectDate": boolean,
  "earliestSelectableUpgradeDate": datetime,
  "latestSelectableUpgradeDate": datetime,
  "upgradeDate": datetime,
  "updateStatus": string (enum | "Scheduled" or "Running"),
  "ignoreUpgradeWindow": boolean,
  "isActive": boolean
}
```

Reschedule Update

Reschedule an update, if able.

```
[200] PUT /v1.2/admin/applications/{applicationFamily}/update
```

Body

```
{  
  "jobId": int,  
  "jobEntryId": int,  
  "runOn": datetime,  
  "ignoreUpgradeWindow": boolean  
}
```

NOTE

All datetime values are in UTC

See Also

[Microsoft Dynamics 365 Business Central Server Administration Tool](#)

Introduction to automation APIs

3/31/2019 • 3 minutes to read

Automation APIs provide capability for automating company setup through APIs. Once the tenants have been created, the automation APIs can be used, in order to hydrate the tenant - to bring the tenant up to a desired state. Usually this involves creating a new company on the tenant, running RapidStart packages, installing extensions, adding users to user groups and assigning permission sets to users.

Delegated admin credentials and Dynamics 365 Business Central users with permissions, can call the APIs.

Automation APIs are placed in the `microsoft/automation` API namespace. In all the examples below, parameters are marked in parenthesis `{}`. Make sure that only valid parameters are passed.

Create a company

To create a company, an `automationCompany` endpoint is available. To create a Company issue a [POST request](#) as shown in the following example.

```
POST
https://api.businesscentral.dynamics.com/v1.0/api/microsoft/automation/{apiVersion}/companies({companyId})/automationCompanies
Authorization: Bearer {token}
Content-type: application/json
{
  "name": "CRONUS",
  "displayName": "CRONUS",
  "businessProfileId": ""
}
```

The `{companyId}` must be the ID of an valid company on the tenant. Issue a [GET automationCompany](#) request to fetch existing companies.

NOTE

The company which is created will not be initialized.

To rename a company, issue a [PATCH automationCompanies](#).

Upload and apply a RapidStart package

RapidStart is uploaded, installed, and applied using the APIs described below. RapidStart operations can be time consuming. To get the current status of the RapidStart packages and running operations issue a [GET configurationPackages](#) as shown in the following example.

```
GET
https://api.businesscentral.dynamics.com/v1.0/api/microsoft/automation/{apiVersion}/companies({companyId})/configurationPackages

Authorization: Bearer {token}
```

In the response, status for the import and apply status will be shown, as well as information about the RapidStart package.

Insert RapidStart

First step is to create the configuration package, by issuing a [POST configurationPackages](#) in the Dynamics 365 Business Central tenant. Once the configuration package is created, the RapidStart package can be uploaded. See the example below.

```
POST
https://api.businesscentral.dynamics.com/v1.0/api/microsoft/automation/{apiVersion}/companies({companyId})/configurationPackages

Authorization: Bearer {token}
Content-type: application/json
{
  "code": "{SAMPLE}",
  "packageName": "{SAMPLE}"
}
```

Upload RapidStart package

Once the configuration package is created, a RapidStart package can be uploaded with a [PATCH configurationPackages](#). See the example below.

```
PATCH
https://api.businesscentral.dynamics.com/v1.0/api/microsoft/automation/{apiVersion}/companies({companyId})/configurationPackages('{SAMPLE}')/file('{SAMPLE}')/content

Authorization: Bearer {token}
Content-type: application/octet-stream
If-Match: *
Body: RapidStart file.
```

Import and apply RapidStart package

Once uploaded, the RapidStart package needs to be imported by issuing a [POST on the bound action Microsoft.NAV.import](#).

```
POST
https://api.businesscentral.dynamics.com/v1.0/api/microsoft/automation/{apiVersion}/companies({companyId})/configurationPackages('{SAMPLE}')/Microsoft.NAV.import

Authorization: Bearer {token}
```

When the RapidStart package is imported it can be applied with a [POST on bound action Microsoft.NAV.apply](#).

```
POST
https://api.businesscentral.dynamics.com/v1.0/api/microsoft/automation/{apiVersion}/companies({companyId})/configurationPackages('{SAMPLE}')/Microsoft.NAV.apply

Authorization: Bearer {token}
```

Managing users, user groups, and permission sets

The automation APIs enable users to be set up in Dynamics 365 Business Central.

Modifying user properties

Get the current user properties by issuing a [GET users](#). This will get the UserSecurityId needed on subsequent requests.

To modify the user, create a [PATCH user](#) request as shown in the example below.

```
PATCH
https://api.businesscentral.dynamics.com/v1.0/api/microsoft/automation/beta/companies({id})/users({userSecurityId})
Content-type: application/json
If-Match:*
{
  "state": "Enabled",
  "expiryDate": "2035-01-01T21:03:53.444Z"
}
```

Assign user permissions and user groups

To assign users to a user group, issue a [POST request](#) against the **userGroupMembers** entity. See the example below.

```
POST
https://api.businesscentral.dynamics.com/v1.0/api/microsoft/automation/{apiVersion}/companies({companyId})/users({userSecurityId})/userGroupMembers

Authorization: Bearer {token}
{
  "code": "D365 EXT. ACCOUNTANT",
  "companyName": "CRONUS USA, Inc."
}
```

To retrieve the list of user groups issue a [GET userGroups](#). This will return the information that you need for the payload above.

Assigning permission sets is identical to adding users to user groups. [GET permissionSet](#) returns information about the available permission sets. To assign a permissionSet issue a [POST userPermission](#) as shown in the following example.

```
POST
https://api.businesscentral.dynamics.com/v1.0/api/microsoft/automation/{apiVersion}/companies({companyId})/users({userSecurityId})/userPermissions

Authorization: Bearer {token}
{
  "id": "SECURITY"
}
```

Removing the permissionSet from the user is done by issuing a [DELETE userPermissions](#) on the users entity.

Handling tenant extensions

Add-on extensions which are already published to the tenant can be installed and uninstalled. Per-tenant extensions can be uploaded and installed. To get the list of all extensions on the tenant, issue a [GET extensions](#). This will return the packageId needed for installing and uninstalling extensions.

Installing and uninstalling published add-on extensions

There are two bound actions available on the **extensions** endpoint: `Microsoft.NAV.install` and `Microsoft.NAV.uninstall`.

Issue a [POST extension](#) using the bound actions. See the example below.

POST

```
https://api.businesscentral.dynamics.com/v1.0/api/microsoft/automation/{apiVersion}/companies({companyId})/extensions({extensionId})/Microsoft.NAV.install
```

Authorization: Bearer {token}

Upload and install a per-tenant extension

Issue a [PATCH](#) against the **extensionUpload** endpoint to upload and install the extension.

NOTE

Installing per-tenant extensions using Automation APIs is only possible in SaaS.

Uninstalling the extension can be done through the bound action [Microsoft.NAV.uninstall](#), as with the add-on extensions.

See Also

[Automation API overview](#)

Automation Overview

3/31/2019 • 2 minutes to read

The following table provides an overview of the available resource types on a Dynamics 365 Business Central tenant. For an overview of how to use the automation APIs, see [Introduction to Automation APIs](#).

RESOURCE TYPE	DESCRIPTION
automationCompany resource type	Represents a user resource type in Dynamics 365 Business Central
company resource type	Represents a user resource type in Dynamics 365 Business Central.
configurationPackages resource type	Represents a configurationPackage resource type in Dynamics 365 Business Central.
extension resource type	Represents an extension resource type in Dynamics 365 Business Central.
permissionSet resource type	Represents a permissionSet resource type in Dynamics 365 Business Central.
user resource type	Represents a user resource type in Dynamics 365 Business Central.
userGroup resource type	Represents a userGroup resource type in Dynamics 365 Business Central.
userGroupMember resource type	Represents a userGroup resource type in Dynamics 365 Business Central.
userPermission resource type	Represents a userPermissions resource type in Dynamics 365 Business Central.

See Also

[Introduction to Automation APIs](#)

Connect to the Intelligent Cloud from On-Premises with Dynamics 365 Business Central

5/21/2019 • 8 minutes to read

Customers running their workloads on-premises can get access to the same intelligent cloud scenarios that customers using Business Central online have. Each on-premises solution that connects to the intelligent cloud through Business Central will be able to replicate data from on-premises to the cloud tenant. In this way, users can access intelligent cloud scenarios of Machine Learning, Power BI, Flow, and others to drive suggested actions.

For the list of currently supported on-premises solutions, see [Which products and versions are supported for connecting to the intelligent cloud?](#) in the FAQ.

Setting up your connection to the intelligent cloud

This section provides the steps required to get intelligent insights through a connection to Business Central online. This can simply be done by following the instructions in the **Intelligent Cloud Setup** assisted setup wizard in your Business Central online tenant.

There are a few key points that need to be understood before proceeding with the setup:

- It is always a best practice to test this configuration in your Sandbox environment before making changes to a production tenant. For more information see [Choosing Your Dynamics 365 Business Central Development Sandbox Environment](#).
- Any existing data in your Business Central tenant will be overwritten with data from your on-premises solution, or source, once the data replication process is run. If you do not want data in your Business Central online tenant to be overwritten, do not configure the connection.
- All users that do not have *SUPER* permissions will be automatically reassigned to the intelligent cloud user group. This will limit them to read-only access within the Business Central tenant. See more below.
- If your data source is Business Central (on-premises), several stored procedures will be added to the SQL server you define. These stored procedures are required to replicate data from your SQL server to the Azure SQL server associated with your Business Central tenant.
- In the current version of Business Central, the amount of data that can be replicated for any tenant is limited to 150GB. If your database is larger than 150GB, try reducing the number of companies you are replicating data for. This can be done using the company selection within the assisted setup guide. Additional options for databases exceeding 150GB will be available in future updates.
- Before setting up the connection to the intelligent cloud, ensure that at least one user in the system has *SUPER* permissions. This is the only user that will be allowed to make changes in the Business Central tenant.
- Configuring the intelligent cloud environment will have no impact on any users or data in your on-premises solution.

To begin configuring the connection, navigate to the assisted setup page and launch the **Intelligent Cloud Setup** assisted setup guide. If you are using Business Central on-premises, the same setup guide is also available in your on-premises solution. You will automatically be redirected to your Business Central online tenant to continue the configuration process.

The assisted setup guide

The assisted setup guide consists of up to 6 pages that take you through the process of connecting your solution to the intelligent cloud.

1. Welcome and Consent page

This page provides an overview of what the wizard will do. You must agree to the displayed warning message before you can continue to the next step.

2. Product selection

On this page, specify the on-premises solution that you want to replicate data from. All supported sources will appear in the list. If you don't see your product, navigate to the **Manage Extensions** page, and then verify that the intelligent cloud extension for your on-premises solution is installed.

3. SQL Connection

If the product you selected requires a SQL connection, this page will be presented. Other source applications may require different information to connect to them. This page will display the connection information based on the product that you specified in the previous page. This is defined from the installed extensions for the product you have selected.

FIELD	DESCRIPTION
<i>SQL Connection</i>	SQL Server , which is your locally installed SQL Server instance, or Azure SQL .
<i>SQL Connection string</i>	<p>You must specify the connection string to your SQL Server. For more information, see the SQL Server blog. The following snippets illustrate a couple connection strings with different formats:</p> <pre>Server={SQL Server Name};Initial Catalog={Database Name};UserID={SQL Authenticated Username};Password={SQL Authenticated Password};</pre> <pre>Server={SQL Server Name};Database={Database Name};User Id={SQL Server Authenticated Username};Password={SQL Server Authenticated Password};</pre> <p>The SQL connection string is passed to Azure Data Factory (ADF), where it is encrypted and delivered to your Self-Hosted Integration Runtime and used to communicate with your SQL Server instance during the data replication process.</p>
<i>Integration runtime name</i>	<p>If your SQL connection is SQL Server, you must specify the runtime service that will be used to replicate the data from the defined source to your Business Central online tenant.</p> <p>If you are a hosting partner, you may have multiple tenants running on the same Integration runtime service. Each tenant will be isolated in their own data pipeline. To add tenants to an existing integration runtime service, enter the name of the existing integration runtime service into this field. The integration runtime name can be found in the Microsoft Integration Runtime Manager. To create a new runtime service, leave the field empty, and then choose the Next button. Once you choose Next, a new replication pipeline will be created in the Azure service. This should take less than a minute to complete.</p>

4. Self-Hosted Integration Runtime (SHIR)

This is the service will allow access to the Azure replication services to your on-premises SQL Database during the replication process. Follow the instructions on this page to install the Self-Hosted Integration

Service (SHIR) to a local machine.

5. Company Selection

You will be provided with a list of companies from your on-premises solution, or source. Select the companies you would like to replicate data for. If the company does not exist in your Business Central tenant, it will be automatically created for you. This process may take several minutes depending on the number of companies that need to be created.

6. Enable & Scheduling Replication

The final page in the wizard allows you to enable the replication process and create a schedule for when the data replication should occur. These settings are also available within your Business Central tenant on the **Intelligent Cloud Management** page. You have the option to schedule replication daily or weekly. We recommend that you schedule your data replication for off-peak business hours.

NOTE

Depending on the amount of data, your SQL configuration and your connection speed, a full replication could take several hours to complete. Subsequent replications will complete more quickly as only changed data is replicating.

Adding a tenant to an existing runtime service, or updating companies

There are some scenarios where it will be necessary for you to run the intelligent cloud assisted setup wizard more than once.

One example is if you want to change the companies you replicate data for. If the companies in your on-premises solution have changed, either added or deleted, or you want to change the companies to replicate, simply run the assisted setup wizard again.

Another example of why you would want to run the wizard again is you may be a hosting partner and want to add tenants to your existing runtime service.

In both examples, you will be making updates to an existing runtime service. When you get to the point of the wizard where you can specify an existing runtime services name, open the Microsoft Integration Runtime Service Manager and enter the runtime name in the field in the wizard; you will not be allowed to copy/paste. The runtime service will identify that you are making updates to an existing service and will not create a new one.

Complete the steps in the wizard to update the runtime service. If the change was related to adding tenants to an existing service, a new data pipeline will be created for that tenant. Changing your replication schedule or regenerating an Azure Data Factory (ADF) key may be done using the **Intelligent Cloud Management** page in your Business Central cloud tenant. For more information, see [Managing your Intelligent Cloud environment](#).

User groups and permission sets

When running as connected with an on-premises solution, the Business Central online tenant will be, with very few exceptions, read-only. Because the on-premises solution is your primary application for running your business activities such as data entry, tax reporting, and sending invoices, these tasks will need to be completed in the on-premises solution. We limit the amount of data you can enter into your Business Central tenant to data that is not replicated, otherwise any data that was written to the tenant database would be continuously overwritten during the replication process.

To make setting up this read-only tenant more efficient, we created a new *Intelligent Cloud* user group and an *Intelligent Cloud* permission set. Once the intelligent cloud environment is configured, all users without SUPER permissions will be automatically assigned to the *Intelligent Cloud* user group. Only users with SUPER permissions will be allowed to make modifications to the system at this point.

NOTE

Before you configure the a connection from on-premises to bBusiness Central, make sure that at least one user in each company is assigned SUPER permissions.

Users that are reassigned to the Intelligent Cloud user group will have access to read ALL data by default. If you need to further restrict what data a user should be able to read, the SUPER user may create new user groups and permissions sets and assign users accordingly. It is highly recommended to create any new permissions sets from a copy of the Intelligent Cloud permission set and then take away permissions you do not want users to have.

WARNING

If you grant insert, modify or delete permissions to any resource in the application that was set to read-only, it could have a negative impact on the data in the Business Central cloud tenant. If this occurs, you may have to clear all your data and rerun a full replication to correct this.

Extensions

When an intelligent cloud environment is configured, it is highly recommended that you test the impact of any extension in a sandbox environment before having it installed in your production Business Central tenant to help avoid any data failures or untended consequences.

System requirements

To connect to the intelligent cloud through Business Central the on-premises solution must use SQL Server 2016 or a later version, and the database must have compatibility level 130 or higher. The on-premises solution must also be one of the supported versions. For more information, see [Which products and versions are supported for connecting to the intelligent cloud?](#) in the FAQ.

See Also

[Managing your intelligent cloud environment](#)

[Replicating on-premises data](#)

[Frequently Asked Questions about connecting to the intelligent cloud](#)

[Your Access to the Intelligent Cloud](#)

Replicating On-Premises Data to Business Central

5/28/2019 • 7 minutes to read

Data replication is the process of securely migrating data from your on-premises SQL Server instance to your Business Central online tenant. The process uses the Azure Data Factory (ADF) to migrate the data between databases directly, meaning it does not look at any permissions within the applications you are transferring data between, only SQL permissions.

In order for the data migration to take place, you must successfully complete the **Intelligent Cloud Setup** assisted setup wizard in your Business Central online tenant. Once the wizard is complete and data replication is activated, an initial data replication will happen at the scheduled time. Alternatively, you can trigger the data replication process manually.

Data is replicated between the two systems on a per-table basis, and success and failures are tracked for each table. If a table fails to replicate, the error will be captured, and the replications moves on to the next table until completed. Tables will fail to replicate if they cannot be found, or if the schema does not match between the cloud and the on-premises tables.

If a table fails to replicate, a blocker is placed on the table in Business Central online tenant. It is meant as a way for the service to inform you that the data you are viewing has not replicated to prevent you from viewing data that may be out of date. At no point will there be an impact on your on-premises SQL Server data.

The initial data replication time can vary depending factors such as the amount of data to replicate, your SQL Server configuration, and your connection speeds. The initial replication will take the longest amount of time to complete because all data is replicating. After the initial replication, only changes in data will be replicated so they should run more quickly.

Data replication from Business Central on-premises

Your Business Central on-premises solution can have an identical twin in a Business Central online tenant. The data replication can be started quite easily from the assisted setup wizard in your on-premises solution. For more information, see [Connect to the Intelligent Cloud from On-Premises](#).

Replicating data from extensions

When your on-premises solution is connected to the intelligent cloud, it is highly recommended that you test the impact of any extension in a sandbox environment before you install the extensions in your Business Central production tenant to help avoid any data failures or unintended consequences.

In order to support data replication, tables and table extensions must specify if data from that table must be replicated or not. By default, the **ReplicateData** property is set to *Yes* so that, by default, any extension that is installed in the Business Central cloud tenant will have all its tables replicated.

In certain circumstances, you may want to not replicate all data. Here are a few examples:

- The extension is installed in the Business Central online tenant but not in the Business Central on-premises solution

In this case, Business Central will attempt to replicate the data but fail. Since the extension is not installed on-premises, any table related to that extension table will fail to replicate, and blocker notifications will appear on pages that are associated with those tables.

If you own the extension, we recommend that you set the **ReplicateData** property to *No* on the extension tables. If you do not, and if you want data to replicate, install the extension in both your Business Central

cloud tenant and your on-premises solution. If you do not want data to replicate, uninstall the extension from your Business Central cloud tenant.

- The extension references a base table

This can cause your base table to appear empty when you view data in your Business Central cloud tenant. If that happens, uninstall the extension from your Business Central cloud tenant, and then run the data replication process again.

Data that is not replicated

During the data replication process, Business Central does not replicate most system tables, users, and permissions.

Data replication from Dynamics GP

When using the intelligent cloud replication for Dynamics GP 2018 R2, the following information is replicated from Dynamics GP to Business Central online:

- Chart of Accounts master records as of the time of the replication

The chart of accounts will be set up as the main account segment from Dynamics GP, and the additional segments will be set up as dimensions in Business Central

- Account Balance as of the time of the replication

The account balances are brought over as a sum amount of the balances grouped by the main account number.

Let's take a look at an example using Fabrikam data in Dynamics GP:

ACCOUNT NUMBER	ACCOUNT NAME	AMOUNT
000-1100-00	Cash	100.00
100-1100-00	Cash Admin	200.00
200-1100-00	Cash Accounting	200.00
000-1100-01	Cash West	200.00
000-1100-02	Cash Midwest	100.00

Because the account number's main segment in Dynamics GP is defined as the second segment, the data replication creates new accounts in Business Central based on the number *1100* in this example. The data replication process then sets up an account in Business Central as shown in the following table:

ACCOUNT NUMBER	ACCOUNT NAME	AMOUNT
1100	Cash	800.00

The data replication generates dimensions on that account based on the different segments. User will see a *Department* dimension with the values 000, 100, and 200 respectively. Another dimension, *Division*, will show the values 00, 01, and 02 respectively.

- Customer master records and outstanding transactions from the Receivables module

These transactions will be brought in as the amount remaining in Dynamics GP.

- Vendor master records and outstanding transactions from the Payables module

These transactions will be brought in as the amount remaining in Dynamics GP.

- Inventory items

Inventory is imported with the cost valuation method that was selected when the company setup wizard was run. Currently, the data replication brings in the quantity on hand for the items at the time of migration. This quantity is brought into the blank location.

- Historical data from Sales Order Processing, Purchase Order Processing, and Inventory

This data can be used in Power BI reports and Power Apps. In Business Central online, the data is included in the SmartList views in the **Customers**, **Vendors**, and **Items** lists. Technically, the data is stored in table extensions.

Data replication from Dynamics NAV

The data replication process for Microsoft Dynamics NAV 2018 is very similar to the process for Business Central on-premises with one major exception: A transformation process that is run on tables that have upgrade logic on them because of differences in data structure between Microsoft Dynamics NAV 2018 and Business Central.

In order to set up the data replication process, you must upgrade to Microsoft Dynamics NAV 2018 CU 15 or later. Cumulative update 15 added an extension that is needed to set up the replication process.

From the standpoint of walking through the wizard, the process is the same. For more information, see [Connect to the Intelligent Cloud from On-Premises](#).

Let's look at an example of the transformation process. In Dynamics NAV, the **Sales & Receivables Setup** window includes a field, **Archive Quotes and Orders**, that specifies whether to automatically archive sales quotes and sales orders when a sales quote or order is deleted. In Business Central, the **Sales & Receivables Setup** window includes an **Archiving** FastTab where you can specify how and when to archive quotes and orders separately.

<
SALES & RECEIVABLES SETUP | WOR
+

✓ SAVED
↗

Sales & Receivables Setup

Customer Groups
Payments
More options

Blanket Order Nos.	S-BLK	Reminder Nos.	S-REM
Order Nos.	S-ORD-1	Issued Reminder Nos.	S-REM+
Return Order Nos.	S-RETORD	Fin. Chrg. Memo Nos.	S-FIN
Invoice Nos.	S-INV	Issued Fin. Chrg. M....	S-FIN+
Posted Invoice Nos.	S-INV+	Posted Prepmt. Inv....	S-INV+
Credit Memo Nos.	S-CR	Posted Prepmt. Cr. M...	S-CR+
Posted Credit Memo...	S-CR+	Direct Debit Mandate...	DDM

Background Posting

Post with Job Queue ...
Post & Print with Job...
Job Queue Category...
Notify On Success

Archive Quotes
Batch Archiving Quot...

Never
Question
Always

Archive Orders
Archive Return Orders

Archive Blanket Orders

When you connect your Microsoft Dynamics NAV 2018 to Business Central, the data replication process must make the relevant data transformation to put the correct values into the Business Central table. Technically, it is the same process that is used for upgrading from Dynamics NAV to Business Central.

Upgrading to a new version of Business Central

If you upgrade to a new version of Business Central, including a cumulative update, then you must update the extensions as well. Depending on your on-premises solution, your Business Central tenant contains different extensions for the intelligent insights. For more information, see [Business Central Intelligent Cloud Extensions](#).

IMPORTANT

You must always install, publish, or upgrade the **Intelligent Cloud Base Extension** extension first, and then the product-specific extension or extensions. Also, if your on-premises solution is Business Central on-premises, then you must update the extensions both on-premises and online.

See also

[Connect to the Intelligent Cloud from On-Premises](#)
[Managing your Intelligent Cloud Environment](#)
[ReplicateData Property](#)
[Intelligent Insights with Business Central](#)

Managing your intelligent cloud environment

3/31/2019 • 2 minutes to read

You can connect your on-premises solution to the Intelligent Cloud through a Business Central cloud tenant. Once you have set up this configuration, you have access to the **Intelligent Cloud Management** page in the Business Central cloud tenant, from where you can manage your Intelligent Cloud environment and data replication.

Intelligent cloud management

The **Intelligent Cloud Management** page provides information about your data replication runs as well as the ability to manage your replication services, for example.

The page provides a view of the status of all replications. You can view the time the replication ran, the status of each replication, and when your next replication is scheduled to run. The **Replication Statistics** tiles show the number of tables replicated and the number of tables that did not replicate due to any errors that occurred during the replication process. Choose a tile to drill into additional details regarding the replication status of each table as well as any messaging to assist you in correcting any errors.

The following table describes the actions that you can run from the page:

ACTION	DESCRIPTION
Manage Schedule	Opens a page where you can set the replication schedule without having to run the assisted setup wizard again.
Run Replication Now	You can disable automatic data migration and trigger data replication manually. Ideally, this would be used only when you received errors in the scheduled data replication, you corrected any errors, and want to push updated data to the cloud outside of a normally scheduled run.
Reset Cloud Data	You may run into instances where you need to reset your cloud data. This option will clear all data in your cloud tenant and enable you to start over with data replication. If you need to clear data in your cloud tenant and are having connectivity issues that persist for more than 7 days, you will need to contact customer support. They will create a ticket to have your tenant data cleared.
Reset Runtime Key	If at any time you feel that your Self Hosted Integration Runtime key is no longer secure, you may select this option to regenerate a new key. A new key will be generated for you and automatically be updated in the Self Host Integration Runtime service.
Get Runtime Key	Returns the existing runtime key.
Disable Intelligent Cloud	Opens a guide that helps you through a checklist of instructions to disable the intelligent cloud configuration. Once the steps in this process are complete, data replication will be discontinued, and you can choose to use your Business Central cloud tenant as your primary solution.

See also

[Frequently Asked Questions about Connecting to the Intelligent Cloud](#)

[Replicating on-premises data](#)

[Connect to the Intelligent Cloud with Dynamics 365 Business Central](#)

[Your Access to the Intelligent Cloud](#)

Frequently Asked Questions about Connecting to the Intelligent Cloud from On-Premises Solutions

5/21/2019 • 6 minutes to read

This section contains answers to frequently asked questions about connecting on-premises solutions to the intelligent cloud through Business Central online.

Which products and versions are supported for this connection?

The current version of Business Central can connect the following products in order to provide intelligent insights:

- Dynamics GP 2018 R2
- Business Central on-premises
- Dynamics NAV 2018 CU 16

Support added with the April 2019 Business Central update

How is my on-premises data replicated to my Business Central online tenant?

Data is replicated using an Azure service called Azure Data Factory (ADF). The Azure Data Factory is a service that is always running within the Business Central online service manager. When the intelligent cloud is configured for your on-premises solution, a data pipeline is created within the ADF service that enables data to flow from your on-premises solution to your Business Central cloud tenant. If your data source is a local SQL Server instance, you will also be asked to configure a self-hosted integration runtime (SHIR). The runtime is installed locally and enables the communication between the cloud services and your on-premise data to communicate without opening any ports or firewalls.

Are there any limits on the amount or type of data will replicate?

Data replication for the initial release will have a limit of 150GB. There are no restrictions on the type of data that can be replicated.

Is my SQL connection string required to set up the connection?

Yes. The SQL connection string is passed to Azure Data Factory, where it is encrypted and delivered to your Self-Hosted Integration Runtime, and used to communication with your SQL Server instance during the data replication process. For more information, see [How do I find my SQL connection string?](#).

I am a hosting partner - do I need to configure the Self-Hosted Runtime Service for each tenant?

No, there is no limit on the number of tenants that can be added to your Self-Hosted Integration Runtime. Each added tenant will have a dedicated pipeline created.

Will data from tables with code customizations replicate?

No, only tables that are available in both your on-premises solution and your Business Central online tenant will

replicate. Any customization would need to be made into an extension and installed on both your on-premises solution and your Business Central online tenant to replicate.

Why are my permissions restricted in the Business Central online tenant?

When you connect your on-premises solution to Business Central online for intelligent insights, all existing users are automatically added to the *Intelligent Cloud* user group, unless they have the SUPER permission set. In this configuration, your on-premises solution is the master where all business transactions take place. The Business Central online environment is read-only, and the data is used to generate intelligent business insights based on your on-premises data for you. We restrict permissions to prevent users from accidentally entering transactions or updating master records only to have that information overwritten and lost when data replication takes place.

Can I 'turn off' my intelligent cloud?

You can switch off your connection to the Business Central online environment at any point. Once you disable your intelligent cloud configuration, your on-premises solution and the Business Central online tenant will become completely independent of one another. If you switch off the connection, and you want to use your Business Central online environment as your primary solution to run and manage your business, you must reassign permissions to provide read/write access to the relevant users.

For more information, see [Managing Users and Permissions](#).

Will my on-premises users and permissions replicate?

No. Since you are not required to configure your on-premises solution with Azure Active Directory (Azure AD), we cannot guarantee a mapping between on-premises users and users in your Business Central online tenant. Business Central online requires Azure AD accounts, and users must be manually added. All permissions must be granted in the Business Central tenant, independent from your on-premises permissions.

For more information, see [Managing Users and Permissions](#).

Can I view insights from cloud services in my on-premises solution?

Yes, the **Intelligent Cloud Insights** page can be hosted within your on-premises solution if that is one of [the currently supported solutions](#). Each user will need to have a Business Central license to view the data.

Can you export to Excel, modify the contents, and import the data back in?

You can export the list to Excel from the Business Central online tenant, but since the data is read-only you cannot make changes and import it again.

Is the data replication only one-way?

Yes, data is only replicated from the on-premises solution to your Business Central online tenant.

Is there a cost to connect to the intelligent cloud?

Currently, the only costs associated with the intelligent cloud are your named user license costs. For more information, see the [Business Central Licensing Guide \(download\)](#).

Why did my Role Center change after configuring the intelligent cloud?

To keep the Role Center experience as clean as possible and avoid permission errors, we automatically hide actions that would generate a permission error for the user.

Should I uninstall all my Business Central extensions?

Not necessarily. Most extensions will run without issues in the online environment. You may want to consider uninstalling extensions that send data to an external service to avoid potential duplicated calls to that service. It is a best practice to test any extension in a sandbox tenant configured for the Business Central online environment that you are connecting to.

How do I build an extension that enables data replication?

The extension must be created in the same manner as any other extension. For data to replicate, you must add a **ReplicateData** property to your table and set the value to *True*. If your extension connects with an external service and you want to restrict any service calls from your Business Central online tenant, a good practice would be to store the connection information in a separate table and set the **ReplicateData** property to *False*. This would enable you to keep the extension installed but prevent it from making any type of service calls from the read-only Business Central tenant. Once the extension is installed in Business Central online and on-premises, the data will begin to replicate.

How do I find my SQL connection string?

A connection string to your SQL database can be found in SQL Management Studio or using Visual Studio. The user name and password defined in the connection requires a SQL Authenticated user name/password. Your connection string should look something like this:

```
Server=tcp:{ServerName},1433;Initial Catalog={DatabaseName};Persist Security Info=False; User ID={UserName};Password={Password};MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=True;Connection Timeout=30;
```

How do I find the Integration Runtime name?

The Integration Runtime name can be found in the Microsoft Integration Runtime Manager. You can find this application in your Windows system tray or by searching for the program. You will not be able to copy and paste the name. You must manually type the name.

Can I connect my Microsoft Invoicing data to the intelligent cloud?

No. Microsoft Invoicing currently does not support connecting to the intelligent cloud through Business Central. If your organization has an existing Invoicing tenant and want to create a Business Central tenant, you must contact Support to have them delete your existing Invoicing tenant.

For more information, see [Using the same Office 365 Account in Dynamics 365 Business Central and Microsoft Invoicing](#).

See also

[Connect to the intelligent cloud with Business Central](#)
[Managing your intelligent cloud environment](#)
[Replicating on-premises data](#)
[ReplicateData Property](#)

-- title: "Microsoft Dynamics 365 Business Central Administration Center Tool" ms.custom: na ms.date: 04/01/2019 ms.reviewer: na ms.suite: na ms.tgt_pltfrm: na ms.topic: article ms.service: "dynamics365-business-central"

5/3/2019 • 4 minutes to read

Business Central Server Administration tool

The Business Central Server Administration tool is a Microsoft Management Console (MMC) snap-in for creating and managing Business Central Server instances.

TIP

You can also administrate your Business Central deployment using Windows PowerShell cmdlets. For more information, see [Microsoft Dynamics 365 Windows PowerShell Cmdlets](#).

Install the Business Central Server Administration tool

To install Business Central Server Administration tool, use the Business Central Setup and choose either **Server Option** or **Administration Tool** under the custom options page. For more information, see [Installing Business Central Using Setup](#).

Run the Business Central Server Administration tool

You typically run the Business Central Server Administration tool by choosing **Business Central Administration** from the Start menu. Or, you can open the MMC first and then add the Business Central snap-in. In this case, choose **Run** from the Start menu and then specify the Microsoft Management Console:

mmc

IMPORTANT

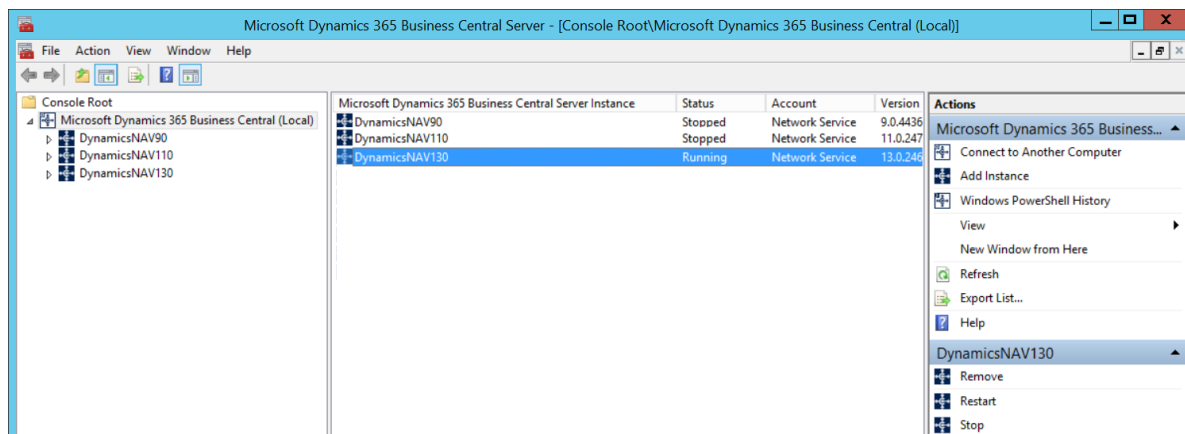
Only members of the Administrator group on the computer are able to use the Business Central Server Administration tool.

The Business Central Server Administration tool is not supported for multi-user environments.

Navigating the Business Central Server Administration tool

Business Central Server Administration tool is divided into panes:

- The left pane shows a tree view that lists all Business Central Server computers that you are administering from this computer and all Business Central Server instances on those computers.



- The **center pane** shows information about the item that you have selected in the left pane. When the selected item is a computer running Business Central Server, the center pane shows a list of Business Central Server instances on that computer and the status of each instance (running or stopped), and the name of the account the instance is running under.

When the item selected in the left pane is a Business Central Server instance, the center pane shows the settings for that instance. For information about a specific setting, see [Configuring Business Central Server Instances](#).

- If the Business Central Server is configured for multitenancy, then you can expand the Business Central Server instance items in the left pane to display a **Tenants** item. Select the **Tenants** item to display all the tenants that are mounted on a Business Central Server instance in the center pane. For more information, see [Multitenant Deployment Architecture](#)
- The **right pane** displays available actions for the object that is selected in the left pane. These options differ depending on whether a Business Central Server computer or a Business Central Server instance is selected.
- The **Windows PowerShell History** pane lists the Windows PowerShell commands that the equivalent of the tasks you perform in the Business Central Server Administration tool. You can access the Windows PowerShell History pane from the **Actions** menu and from the right pane. To run a command that is shown in the **Windows PowerShell History** pane, you can copy the command and paste it into the Dynamics NAV Administration Shell, for example.

Connect to remote computers and multiple server instances

You can use the Business Central Server Administration tool to connect to other computers on your network where Business Central Server instances are installed, and then manage those instances.

1. Configure the remote computers to receive Windows PowerShell remote commands by running the [Enable-PSRemoting](#) cmdlet on each computer.
2. Next, you can start the Business Central Server Administration tool.
3. If you want to connect to a single remote computer, you can start the Business Central Server Administration tool from the Start menu of your computer.
4. If you want to connect to multiple remote computers, you must start the Business Central Server Administration tool from the **Run** program that you can access in the Start menu,
 - a. Enter the command `mmc`.
 - b. In the Management Console, on the **File** menu, choose **Add/Remove Snap-in** to open the **Add or remove Snap-ins** dialog box.
 - c. In the **Available snap-ins** list, double-click **Microsoft Dynamics 365 Business Central**.

- d. In the **Connect to another computer** dialog box, type the name of a Business Central Server computer in the **Server Name** box, and then choose **OK**.
- e. Double-click *Microsoft Dynamics 365 Business Central** again, and then enter the name of a different Business Central Server computer in the **Server Name** box. Choose **OK**.

You can also accept the default value, which is **(Local)**, if this is one of the Business Central Server computers that you will be administering.

- f. Continue selecting Business Central Server computers as needed. When you are finished selecting computers, choose **OK** to close the **Add or remove Snap-ins** dialog box.

Now you see multiple Business Central Server computers listed in the tree view pane of the Business Central Server Administration tool.

TIP

When you close MMC, you are prompted to save your settings to a Microsoft Management Console (.msc) file. If you save your settings, then you can use this file to open MMC with your Business Central Server computers already listed

See Also

[Configuring Business Central Server Instances](#)
[Administration Center API](#)

Authentication and Credential Types for Dynamics 365 Business Central

3/31/2019 • 4 minutes to read

In Business Central online, users are added through the Office 365 Admin Center. Once users are created in Office 365, they can be imported into the **Users** window in Business Central. For more information, see [Managing Users and Permissions](#) in the business functionality content.

Configuring Authentication for On-Premises Deployments

An on-premises deployment of Business Central supports several credential authorization mechanisms for users. When you create a user, you provide different information depending on the credential type that you are using in the current Business Central Server instance.

IMPORTANT

All users of a Business Central Server instance must be using the same credential type. In on-premises deployments, you can specify which credential type is used for a particular Business Central Server instance in the Business Central Server Administration tool.

Credential Types

Business Central on-premises supports the following credential types.

CREDENTIAL TYPES	DESCRIPTION
Windows	With this credential type, users are authenticated using their Windows credentials. You can only specify Windows as the credential type if the corresponding user exists in Windows (Active Directory, local workgroup, or the local computer's users). Because they are authenticated through Windows, Windows users are not prompted for credentials when they access Business Central.
UserName	With this setting, the user is prompted for username/password credentials when they access Business Central. These credentials are then validated against Windows authentication by Business Central Server. There must already be a corresponding user in Windows. Security certificates are required to protect the passing of credentials across a wide-area network. Typically, this setting should be used when the Business Central Server computer is part of an authenticating Active Directory domain, but the computer where the Dynamics NAV Client connected to Business Central is installed is not part of the domain.

CREDENTIAL TYPES	DESCRIPTION
NavUserPassword	With this setting, authentication is managed by Business Central Server but is not based on Windows users or Active Directory. The user is prompted for username/password credentials when they start the client. The credentials are then validated by an external mechanism. Security certificates are required to protect the passing of credentials. This mode is intended for hosted environments, for example, where Business Central is implemented in Azure.
AccessControlService	<p>With this setting, Business Central relies on Azure Active Directory (Azure AD\ for user authentication services.</p> <p>Azure AD is a cloud service that provides identity and access capabilities, such as for applications on Azure, in Microsoft Office 365, and for applications that install on-premises. If the Business Central Server instance is configured to use AccessControlService authentication, you can specify an Azure AD account for each user in the Office 365 Authentication field so that they can access both the Business Central and their Office 365 site. Also, if you use Business Central in an app for SharePoint, users have single sign-on between the SharePoint site and Business Central. For more information, see Authenticating Users with Azure Active Directory or Authenticating Users with Active Directory Federation Services.</p> <p>Security certificates are required to protect the passing of credentials across a wide-area network.</p>
None	For internal use on system sessions and typically should not be used. If you choose None , then the Business Central Server instance cannot start.
ExchangeIdentity and TaskScheduler	For internal use only. Do not use.

IMPORTANT

If Business Central Server is configured to use NavUserPassword or AccessControlService authentication, then the username, password, and access key can be exposed if the SOAP or OData data traffic is intercepted and the connection string is decoded. To avoid this condition, configure SOAP and OData web services to use Secure Socket Layer (SSL). For more information, see [Walkthrough: Configuring Web Services to Use SSL \(SOAP and OData\)](#) in the ITPro content for Microsoft Dynamics NAV 2018.

Configuring the Credential Type for Client and Server

For on-premises deployment, you must make sure that clients and Business Central Server are configured to use the same credential type.

When you change the credential type for a Business Central Server instance and the relevant client configurations, the changes take effect when you restart the Business Central Server instance and users connect to the instance again.

Server Configuration

To edit the configuration for the Business Central Server instance, you can use either the Business Central Server Administration tool or the Business Central Administration Shell. In the Business Central Server Administration tool, you configure the credential type in the **Credential Type** field on the **General** tab. Alternatively, you can edit the CustomSettings.config file. For more information, see [Configuring Business](#)

IMPORTANT

When Business Central Server services are deployed on Azure but not as part of Business Central online, you must configure them on Azure. For more information, see [How to: Open Microsoft Dynamics NAV Clients that Connect to Microsoft Dynamics NAV on Microsoft Azure](#) in the ITPro content for Microsoft Dynamics NAV 2018.

Client Configuration

In the relevant configuration file, find the **ClientServicesCredentialType** parameter and change the value to one of the options listed earlier.

For the Business Central Web client users, you must modify the *navsettings.json* for the Business Central Web Server. The *navsettings.json* file is a Java Script Object Notification file type that is similar to files that use the XML file format. The file is stored in the physical path of the web server instance, which is by default is *c:\inetpub\wwwroot\BC140*. For more information, see [Settings in the navsettings.json](#).

For each Dynamics NAV Client connected to Business Central user, you must modify the ClientUserSettings.config file. The default location for this file is **C:\Users\<username>\AppData\Roaming\Microsoft\Microsoft Dynamics NAV\130**, where *<username>* is the name of the user. For more information, see [Configuring the Microsoft Dynamics NAV Windows Client](#) in the ITPro content for Microsoft Dynamics NAV 2018.

Security Certificates

With UserName, NavUserPassword, and AccessControlService credential types require that you install and configure security certificates on components. For more information, see [Using Security Certificates with Business Central On-Premises](#)

See Also

[Understanding Users, Profiles, and Role Centers](#)
[Configuring Business Central Server](#)

[]

Configuring Business Central Server

5/21/2019 • 49 minutes to read

When you run Business Central Setup and install Business Central Server, you provide information that is then used as the configuration for the default Business Central Server instance. This information is stored in a configuration file for the server instance called CustomSetting.config. The default location of the CustomSettings.config file is *C:\Program Files\Microsoft Dynamics 365 Business Central\140\Service*.

After you install Business Central Server, you can change any of the settings that you provided during Setup, plus several other settings that were not available to you in Setup.

NOTE

Each Business Central Server instance has its own CustomSettings.config file.

Configuring Business Central Server in Setup

You configure the default instance of Business Central Server by running Business Central Setup and selecting one of the following:

- Demo Option
- Server Option
- Developer Option
- Customize > Server

After you specify an installation option or customize your component list, the **Specify parameters** pane is displayed in Setup. The list of parameters that you see in the **Specify parameters** pane depends on which components you have selected for configuration. Setup provides a short description for each parameter.

Configuring Business Central Server After Installation

After you install Business Central Server, you can change the configuration settings in the CustomSettings.config file of a Business Central Server instance in the following ways:

- Using the Business Central Server Administration tool.

For more information, see [Settings in the Business Central Server Administration Tool](#) and [Business Central Server Administration Tool](#).

- Using the [Set-NAVServerConfiguration cmdlet](#) that is available in the Business Central Administration Shell.

For more information, see [Using Administration Shell Cmdlets to Modify Settings](#).

- By directly editing CustomSettings.config using a text editor.

We recommend that you do not directly edit the configuration file, because if you make any errors in typing, then you may not be able to start the instance.

Restarting Business Central Server after modifications

If you use the Business Central Server Administration tool or modify the CustomSettings.config file directly, you must restart the Business Central Server instance before any changes can take effect.

If you use the [Set-NAVServerConfiguration cmdlet](#), whether you need to restart the server instance will depend on the configuration setting that you change. There are several settings that are *dynamically updatable*, which means that a server instance restart is not necessarily required after modification. For more information, see [Modifying dynamically updatable settings](#). In the tables that follow, these settings are indicated by the text **Dynamically Updatable: Yes**.

Business Central Server Instance Settings

This section describes all the configuration settings for a Business Central Server instance. The settings are grouped according to the tabs under which they appear in the Business Central Server Administration tool.

- The **Setting** column displays the name of the setting as it appears in the Business Central Server Administration tool.
- The **Key Name** column displays the name of the setting as it appears in the CustomSettings.config file, and is also the name to use for the setting when using the Set-NAVServerConfiguration cmdlet.

General Settings

The following table describes fields on the **General** tab in the Business Central Server Administration tool.

SETTING	KEY NAME	DESCRIPTION				
Build Restriction	ClientBuildRestriction	<p>Specifies what happens when a Business Central client tries to connect to the Business Central Server instance when the client is running a different build version of Business Central than the server instance.</p> <p>Values:</p> <p>AlwaysConnect</p> <p>WarnClient Before connecting the client to the server instance, a message appears that informs the user that the build versions for the client and server instance are different. The user can choose to continue or cancel the connection.</p> <p>DoNotAllow A message appears that informs the user that the client and server instance build versions are different, and the client does not connect to the server instance.</p> <p>Note: With the Business Central Web client and Business Central Tablet client, this setting compares the build version of the Business Central Web Server on IIS with the Business Central Server instance. It controls the connection between Business Central Web Server and the server instance.</p> <p>Default: WarnClient Dynamically Updatable: No</p>				

SETTING	KEY NAME	DESCRIPTION				
Certificate Thumbprint	ServicesCertificateThumbprint	<p>If you use security certificates to protect communications between Business Central Server and client services or web services over an open or wide-area network, you must provide the certificate thumbprint to Business Central Server by updating this setting. For more information, see Using Security Certificates.</p> <p>Default: Dynamically Updatable: No</p>				
Compile and Load Business Application	CompileBusinessApplicationAtStartup	<p>Specifies whether the Business Central Server instance compiles all the business application assemblies and loads them to cache memory when the server instance is started. The assemblies are then retrieved from memory when requested by a Business Central client.</p> <p>Enabling this setting will reduce the time it takes for the server instance to load application objects the first time they are requested by a Business Central client after the server instance started. However, it will also slightly increase the memory usage by the server instance.</p> <p>If you enable this setting, when the server instance starts for the first time, the business application assemblies will be compiled and loaded to the cache memory of the computer that is running the server instance. The assemblies, along with metadata such as object timestamp information, are</p>				

SETTING	KEY NAME	DESCRIPTION				
		<p>also stored to a temporary folder on the computer's file system.</p> <p>Whenever the server instance is restarted, it will compare the assemblies that are stored in memory with corresponding objects in the connected database to determine whether the assemblies in memory can be reused. An assembly will be reused if the following conditions are met:</p> <ul style="list-style-type: none">- The connected database is the same as before, based on the <i>databasemagic</i> field in the dbproperty table.- The object time stamp that is recorded on the compiled assembly matches the object timestamp in metadata of the connected database. <p>If the conditions are not met for an assembly or an assembly for an object in the database is not found in the memory, then a new assembly is built and stored for reuse to cache memory and the file system of the server instance compute for reuse.</p> <p>If you disable this setting, individual assemblies will be compiled on-demand as application objects are requested by the Business Central client. The compiled assemblies will not be reused on subsequent server instance restarts.</p> <p>Notes:</p> <ul style="list-style-type: none">• This setting does not apply to query objects.• Assembly				

SETTING	KEY NAME	DESCRIPTION				
		<p>compilation happens asynchronously.</p> <ul style="list-style-type: none">• On average, all application objects will be loaded within the first few minutes that the server instance operates. <p>Default: Enabled Dynamically Updatable: No</p>				

SETTING	KEY NAME	DESCRIPTION				
Credential Type	ClientServicesCredentialType	<p>Specifies the authentication mechanism for Business Central users of this Business Central Server instance.</p> <p>The options are Windows, Username, NavUserPassword, AccessControlService, and None. For more information, see Authentication and User Credential Types.</p> <p>If you choose AccessControlService, you must specify a federation metadata location for use with Azure AD. If you choose NavUserPassword, and you specify a token signing key, you can use both NavUserPassword and AccessControlService for this server instance.</p> <p>Notes:</p> <ul style="list-style-type: none"> • None is for internal use on system sessions and typically should not be used. If you choose None, then the Business Central Server instance cannot start. • ExchangeIdentity and TaskScheduler are for internal use only, and should not be used. <p>Default: Windows Dynamically Updatable: No</p>				

SETTING	KEY NAME	DESCRIPTION				
Data Cache Size	DataCacheSize	<p>The contextual size of the data cache. The value must be in the range 1-20.</p> <p>Default: 9 Dynamically Updatable: Yes</p>				
Default Client	DefaultClient	<p>Specifies the client type that is used to generate URLs when the client type is set to Default.</p> <p>The options are Current, Windows, Web, SOAP, and OData.</p> <p>Default: Current Dynamically Updatable: No</p>				

SETTING	KEY NAME	DESCRIPTION				
Default Language	DefaultLanguage	<p>Specifies which of the installed Business Central languages on the server instance will be used as the default language in the clients. Set the value to a valid language culture name, such en-US or da-DK.</p> <p>In the Business Central Web and Tablet clients, the Default Language setting determines the language that is used if the web browser's language setting does not match any installed language or a language in the Supported Languages setting, if used. In the Business Central Windows client, this is the language that is used if the language setting of the computer does not have a match.</p> <p>If there are application-specific configuration settings, this setting will be overridden by the default language setting that is specified in application-specific configuration file. For more information, see Set-NAVServerAppConfiguration cmdlet.</p> <p>Default: en-US Dynamically Updatable: No</p>				

SETTING	KEY NAME	DESCRIPTION				
Diagnostic Trace Level	TraceLevel	<p>Specifies the lowest severity level of custom telemetry events to be emitted and recorded in the event log for the Business Central Server instance. This includes system telemetry trace events and custom telemetry events. Telemetry events have IDs from 700-706.</p> <p>The setting has the following values, which correspond to the event severity levels (listed from highest to lowest level): Critical, Error, Warning, Normal (this corresponds to the Information level), Verbose, and Off.</p> <p>You use this setting to filter out lower-level events from the log. For example, if you set this setting to Error, only Error and Critical events will be logged.</p> <p>Set to Off if you do not want to record telemetry events. When set to Off, events are not emitted.</p> <p>Note: Telemetry trace events are recorded in the Business Central Server channel logs, which you can see in Event Viewer, under Applications and Services Logs > Microsoft > Dynamics365BusinessCentral > Common > Admin.</p> <p>Default: Normal Dynamically Updatable: Yes</p>				

SETTING	KEY NAME	DESCRIPTION				
Diagnostic Trace Level for External Proxies	ExternalTraceLevel	<p>Specifies the lowest severity level of telemetry events from external proxies that you want the Business Central Server instance to emit if an error related to the external system occurs on the server instance. This setting pertains to systems and components that Business Central integrates with, like Dynamics 365 for Sales (CRM/Xrm).</p> <p>The server instance listens for event traces from the external proxy. If an error occurs on the server instance, it will emit the last 10 telemetry trace events from the external proxy. The trace events can then be recorded in the Windows event log or picked up by other event trace collection tools.</p> <p>The setting has the following values, which correspond to the event severity levels (listed from highest to lowest level): Critical, Error, Warning, Information, Verbose, and Off.</p> <p>Events that have a lower severity level than the set value will not be emitted. For example, if you set this setting to Error, only Error and Critical events will be emitted. Set to Off if you do not want to emit any of these events.</p> <p>Default: Error Dynamically Updatable: Yes</p>				

SETTING	KEY NAME	DESCRIPTION				
Disable Token-Signing Certificate Validation	DisableTokenSigningCertificateValidation	<p>Specifies whether to enable or disable the validation of the token-signing certificate used by Active Directory Federation Services (AD FS). If the check box is cleared (or the value set to <code>false</code>), the validation is enabled. If the check box is selected (or the value is set to <code>true</code>), then validation is disabled.</p> <p>You should disable token signing certificate validation when configuring Azure Active Directory authentication with single sign-on.</p> <p>Default: Checkbox cleared; set to <code>false</code>. Dynamically Updatable: No</p>				
Enable Certificate Validation	ServicesCertificateValidationEnabled	<p>Specifies whether validation should be performed on the security certificate.</p> <p>Default: Enabled Dynamically Updatable: No</p>				
Enable Debugging	EnableDebugging	<p>Specifies whether the Business Central Server instance starts with debugging enabled.</p> <p>If this option is enabled, the following occurs:</p> <p>When the client first connects, all C# files for the application are generated. C# files persist between Business Central Server restarts. Application objects are compiled with debug information.</p> <p>Default: Not enabled Dynamically Updatable: No</p>				

SETTING	KEY NAME	DESCRIPTION				
Enable Event Logging to Windows Application Log	EnableApplicationChannelLog	<p>Specifies whether to record admin and operational type events (errors, warnings, and information messages) that occur on Business Central Server instances in the Windows Application log on the computer that is running Business Central Server.</p> <p>Because Business Central Server instance events are always logged to the Application and Services Logs, you can disable logging Business Central Server instance events in the Windows Application log and not lose any data. For more information, see Monitoring Business Central Server Events Using Event Viewer and Disable Logging Events to the Windows Application Log.</p> <p>Important: If you are using System Center Operations Manager to monitor Business Central Server instances, do not disable logging to the Windows Application log. If you do, monitoring will not work.</p> <p>Default: Enabled Dynamically Updatable: No</p>				
Enable File Access by AL Functions	EnableALServerFileAccess	<p>Specifies whether AL functions of the file data type can access files on the Business Central Server computer.</p> <p>Default: Enabled Dynamically Updatable: No</p>				

SETTING	KEY NAME	DESCRIPTION				
Enable Full AL Function Tracing	EnableFullALFunctionTracing	<p>Specifies whether full AL function tracing is enabled on Event Tracing for Windows (ETW) sessions.</p> <p>When this setting is enabled, all AL function calls and statements are traced.</p> <p>When this setting is disabled, only root AL function calls are traced. Statements and functions that are called from a function are not traced.</p> <p>For more information, see Monitoring Business Central Server Events .</p> <p>Default: Not enabled Dynamically Updatable: Yes</p>				
Enable Incremental Company Deletion	UseIncrementalCompanyDelete	<p>Specifies whether to delete companies incrementally. If you enable this setting, when you delete a company, the company record is deleted from the database immediately but the company data that is stored in the SQL tables will be deleted later by a system task in task scheduler.</p> <p>You can override this setting when using the Remove-NAVCompany cmdlet by setting the - ForceImmediateDataDeletion parameter.</p> <p>Default: Not enabled Dynamically Updatable: Yes</p>				

SETTING	KEY NAME	DESCRIPTION				
Enable Session While Sync Pending	AllowSessionWhileSyncPending	<p>Specifies whether new client sessions can be created while the tenant's state is OperationalWithSyncPending.</p> <p>The OperationalWithSyncPending state occurs when changes have been made to one or more application tables, but the schema changes have not been synchronized with the tenant. In this state, unless you enable client sessions, clients trying to connect with will get an error message similar to: The tenant 'tenantID' is not accessible.</p> <p>Default: Not enabled Dynamically Updatable: Yes</p>				
Encryption Key Provider	EncryptionProvider	<p>Specifies where the encryption key is that is used to encrypt data in the database, either LocalKeyFile or AzureKeyVault values. If you use AzureKeyVault, see the Azure Key Vault Encryption Provider tab settings.</p> <p>Default: LocalKeyFile Dynamically Updatable: No</p>				
Lockout Sign-In Attempts Count	LockoutPolicyFailedAuthenticationCount	<p>Specifies the number of failed sign-in attempts on a user account (within the time window set by the Lockout Failed Sign-In Attempts Window setting) at which the user account is disabled.</p> <p>Default: 0 Dynamically Updatable: No</p>				

SETTING	KEY NAME	DESCRIPTION				
Lockout Failed Sign-In Attempts Window	LockoutPolicyFailedAuthenticationWindow	<p>Specifies time window, in seconds, during which consecutive failed authentication attempts are counted. This setting works in conjunction with the Account Lockout Max. Sign-In Attempts setting. When the number of failed sign-in attempts by a user hits the value of the Account Lockout Max. Sign-In Attempts setting within this time window, the user account is disabled.</p> <p>Default: 0 Dynamically Updatable: No</p>				
Max Concurrent Calls	MaxConcurrentCalls	<p>The maximum number of concurrent client calls that can be active on this Business Central Server instance.</p> <p>Range: 1 - 2,147,483,647</p> <p>You can also use MaxValue as a value to indicate no limit.</p> <p>Default: 40 Dynamically Updatable: No</p>				

SETTING	KEY NAME	DESCRIPTION				
Max Data Rows Allowed to Send to Excel	MaxRowsToExportToExcel	<p>Specifies the maximum number of rows that can be included in an Excel document that is generated from data in a list type page in the client.</p> <p>If you do not want to have a limit on rows, set the value to MaxValue.</p> <p>Note: This setting only pertains to list type pages in the client. For other pages types, like cards, the limit on rows is configured in the client.</p> <p>Default: MaxValue Dynamically Updatable: Yes</p>				
Maximum Stream Read Size	MaxStreamReadSize	<p>Specifies the maximum number of bytes that can be read from a stream (InputStream object) in a single AL read operation, such a READ or InputStream.READTEXT function call. This setting pertains to UTF-8 and UTF-16 text encoding; not MS-DOS encoding.</p> <p>Default: 1000000 Dynamically Updatable: Yes</p>				
Multitenant	Multitenant	<p>Specifies if the Business Central Server instance can be used in a multitenant environment.</p> <p>Tenant databases can only be mounted on the Business Central Server instance if this setting is selected. For more information, see Multitenant Deployment Architecture.</p> <p>Default: Not enabled Dynamically Updatable: No</p>				

SETTING	KEY NAME	DESCRIPTION				
Network Protocol	NetworkProtocol	<p>Specifies the network protocol for accessing the database.</p> <p>Valid values: Default, Named, Sockets, MultiProtocol</p> <p>Default: Default Dynamically Updatable: No</p>				
Services Default Company	ServicesDefaultCompany	<p>Specifies the Business Central company that the client services, OData web services, and NAS services use as the default company.</p> <p>If your Business Central database contains only one company, leave the setting blank.</p> <p>Default: Dynamically Updatable: No</p>				

SETTING	KEY NAME	DESCRIPTION				
Services Default Time Zone	ServicesDefaultTimeZone	<p>Specifies the time zone in which web service and NAS services calls are run.</p> <p>Values:</p> <p>UTC All business logic for web services and NAS services on the server instance runs in Coordinated Universal Time (UTC).</p> <p>Server Time Zone Services use the time zone of the computer that is running Business Central Server.</p> <p>ID of any time zone recognized by the current version of Windows Specifies any Windows time zone as defined in the system registry under HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Time Zones. For example, Romance Standard Time.</p> <p>Default: UTC Dynamically Updatable: No</p>				
Services Language	ServicesLanguage	<p>Specifies the global language version to use for text strings with SOAP and OData web services.</p> <p>The value must be valid culture name for a language that is available for the Microsoft Dynamics NAV solution, such as en-US and da-DK.</p> <p>Default: en-US Dynamically Updatable: No</p>				

SETTING	KEY NAME	DESCRIPTION				
Services Option Text Source	ServicesOptionFormat	<p>Specifies the source of the text strings to use for the option values of an option data type field.</p> <p>Values:</p> <p>OptionString Uses the text strings that are specified by the OptionString Property of a field.</p> <p>OptionCaption Uses the text strings that are specified by the OptionCaption Property of a field.</p> <p>Default: OptionCaption Dynamically Updatable: No</p>				
Session Event Table Retain Period	SessionEventTableRetainPeriod	<p>Specifies the number of months that sessions in the Session Event table remain before they are deleted.</p> <p>Default: 3 Dynamically Updatable: No</p>				
Non-Interactive Sessions Log Retain Period	NonInteractiveSessionsLogRetainPeriod	<p>Specifies the number of days that background and web service sessions remain in the Session Event table before they are deleted.</p> <p>Default: 5 Dynamically Updatable: No</p>				

SETTING	KEY NAME	DESCRIPTION				
Supported Languages	SupportedLanguages	<p>Specifies which of the installed Business Central languages on the server instance will be available for use in the clients. If you do not specify a language, then all installed languages will be available. In the client, users can switch among the supported languages.</p> <p>The setting's value is a semicolon-separated list that contains the language culture names for each language. For example, if you want client users to be able to choose among da-DK, en-US, and en-CA, set the value to da-DK;en-US;en-CA.</p> <p>If you specify any languages in this setting, then you must include the language that you specified in the Default Language setting.</p> <p>If there are application-specific configuration settings, this setting will be overridden by the supported language setting that is specified in application-specific configuration file. For more information, see Set-NAVServerAppConfiguration cmdlet.</p> <p>Default: Dynamically Updatable: No</p>				

SETTING	KEY NAME	DESCRIPTION				
Token Signing Validation Mode	TokenSigningCertificateValidationMode	<p>Specifies which certificate validation mode to use for token signing validation. This setting is applicable only if the Enable Certificate Validation setting is selected. IssuerNameValidation validates tokens by verifying the issuer name (tenant) only. PeerOrChainValidation validates tokens by verifying that the certificate is either in the Trusted People store or is part of a chain trust to a certification authority in the Trusted Root store.</p> <p>Default: IssuerNameValidation Dynamically Updatable: No</p>				
UI Elements Removal	UIElementRemovalOption	<p>Specifies whether UI elements are hidden when the related object is not accessible according to the license or according to user permissions or both. For more information, see Hiding UI Elements.</p> <p>Default: LicenseFileandUserPermissions Dynamically Updatable: No</p>		Use NTLM Authentication	ServicesUseNTLM Authentication	<p>Specifies whether NTLM authentication is enabled for web services. To require Kerberos authentication, disable this option.</p> <p>Default: Not enabled Dynamically Updatable: No</p>
<i>not available</i>	XmlMetadataCacheSize	<p>For internal use only.</p> <p>Default: 500</p>				

Database Settings

The following table describes fields on the **Database** tab in the Business Central Server Administration tool.

NOTE

If the Business Central Server instance is configured as a multitenant server instance, then except for the **Database Name**, **Database Instance**, and **Database Server** settings, the settings apply to both the application database and the tenant database.

SETTING	KEY NAME	DESCRIPTION
---------	----------	-------------

SETTING	KEY NAME	DESCRIPTION
Database Instance	DatabaseInstance	<p>The name of the SQL Server database instance to connect to. If the value is a null string (""), Business Central Server instance connects to the default database instance of SQL Server.</p> <p>If the Business Central Server instance is configured as a multitenant server instance, then this setting specifies the SQL Server database instance that hosts the application database.</p> <p>Default: NAVDEMO Dynamically Updatable: No</p>
Database Name	DatabaseName	<p>The name of the Business Central database in SQL Server.</p> <p>If the Business Central Server instance is configured as multi-tenant server instance, then this setting specifies the application database.</p> <p>Default: "Demo Database BC (14-0)" Dynamically Updatable: No</p>
Database Server	DatabaseServer	<p>A valid network name for the computer that is running SQL Server.</p> <p>If the Business Central Server instance is configured as multi-tenant server instance, then this setting specifies the computer that hosts the application database.</p> <p>Default: The computer that you selected in Business Central Setup. Dynamically Updatable: No</p>
Disable SmartSQL	DisableSmartSql	<p>Specifies whether the SmartSQL performance optimization feature is disabled.</p> <ul style="list-style-type: none"> - If the check box is selected, SmartSQL is disabled. - If the check box is cleared, SmartSQL is enabled. <p>When SmartSQL is enabled, Business Central Server converts FIND calls and FlowField calculations into a single SQL statement. This can improve performance when running pages that contain FlowFields. However, it can be helpful to disable SmartSQL when troubleshooting database queries because statements are separated into more discrete statements. For more information, see Troubleshooting: Long Running SQL Queries Involving FlowFields by Disabling SmartSQL.</p> <p>Default: SmartSQL performance optimization is enabled (check box is cleared) Dynamically Updatable: Yes</p>
Disable SQL Query Hint FORCE ORDER	DisableQueryHintForceOrder	<p>Specifies whether the FORCE ORDER Query Hint is used in queries. FORCE ORDER instructs the query optimizer to preserve the join order that is indicated by the query syntax.</p> <p>If you clear the check box (<input type="checkbox"/> <code>false</code>), the OPTIMIZE FOR UNKNOWN hint is used in queries.</p> <p>For more information, see Configuring Query Hints for Optimizing SQL Server Performance.</p> <p>Default: OPTIMIZE FOR UNKNOWN hint is disabled (check box is selected) Dynamically Updatable: Yes</p>

SETTING	KEY NAME	DESCRIPTION
Disable SQL Query Hint LOOP JOIN	DisablQueryHintLoopJoin	<p>Specifies whether the LOOP JOIN Query Hint is used in queries. LOOP JOIN instructs the query optimizer to use LOOP JOIN for all join operations in the whole query.</p> <p>If you clear the check box (<input type="checkbox"/>) , the OPTIMIZE FOR UNKNOWN hint is used in queries.</p> <p>For more information, see Configuring Query Hints for Optimizing SQL Server Performance.</p> <p>Default: OPTIMIZE FOR UNKNOWN hint is disabled (check box is selected) Dynamically Updatable: Yes</p>
Disable SQL Query OPTIMIZE FOR UNKNOWN	DisableQueryHintOptimizeForUnknown	<p>Specifies whether the OPTIMIZE FOR UNKNOWN Query Hint is used in queries. OPTIMIZE FOR UNKNOWN instructs the query optimizer to use statistical data instead of the initial values for all local variables when the query is compiled and optimized, including parameters created with forced parameterization.</p> <p>If you clear the check box (<input type="checkbox"/>) , the OPTIMIZE FOR UNKNOWN hint is used in queries.</p> <p>For more information, see Configuring Query Hints for Optimizing SQL Server Performance.</p> <p>Default: OPTIMIZE FOR UNKNOWN hint is enabled (check box is cleared) Dynamically Updatable: Yes</p>
Enable Buffered Insert	BufferedInsertEnabled	<p>Specifies whether to buffer rows that are being inserted into a SQL Server database table.</p> <p>When this parameter is enabled, up to 5 rows will be buffered in the table queue before they are inserted into the table.</p> <p>To optimize performance in a production environment, you should enable this parameter. In a test environment, you can disable this parameter to help debug failures that occur when you insert rows in an SQL database table. For more information, see Bulk Inserts.</p> <p>Default: Enabled Dynamically Updatable: No</p>
Enable Encryption on SQL Server Connections	EnableSqlConnectionEncryption	<p>Specifies whether the SQL connect string should request encryption when connecting to SQL Server services.</p> <p>Default: Not enabled Dynamically Updatable: No</p>
Enable Trust of SQL Server Certificate	TrustSQLServerCertificate	<p>Specifies whether Business Central Server should trust the SQL Server certificate.</p> <p>Default: Not enabled Dynamically Updatable: No</p>
Search Timeout	SearchTimeout	<p>Specifies the time (in seconds) that a search operation on lists in the client will continue until it is terminated. When the limit is reached, the following message displays in the client: Searching for rows is taking too long. Try to search or filter using different criteria.</p> <p>Time format: hh:mm:ss Default: 00:00:10 Dynamically Updatable: Yes</p>

SETTING	KEY NAME	DESCRIPTION
SQL Bulk Import Batch Size	SqlBulkImportBatchSize	<p>Specifies how many SQL memory chunks that a data import must be distributed across. Lowering the value increases the number of network transfers and decreases performance, but also lowers the amount of memory that the server instance consumes. If the database is on SQL Server 2016 or later, a low value can lead to large data files. If you do not want to use batching, specify 0.</p> <p>Default: 448 Dynamically Updatable: No</p>
SQL Command Timeout	SqlCommandTimeout	<p>The contextual time-out for a SQL command.</p> <p>Default: 0:30:00 Dynamically Updatable: No</p>
SQL Connection Idle Timeout	SqlConnectionIdleTimeout	<p>Specifies the time that a SQL connection can remain idle before being closed. The value has the format HH:MM:SS.</p> <p>Default: 00:05:00 Dynamically Updatable: Yes</p>
SQL Connection Timeout	SqlConnectionTimeout	<p>Specifies the time to wait while trying to connect to the database before terminating the attempt and generating an error. This setting also applies to begin, rollback and commit of transactions.</p> <p>The value has the format HH:MM:SS.</p> <p>Default: 00:01:30 Dynamically Updatable: Yes</p>
Enable SQL Parameters by Ordinal	SqlParametersByOrdinal	<p>Specifies whether parameters in SQL statements are referenced by their ordinal number.</p> <p>Enabling this setting improves performance when using buffered inserts.</p> <p>Default: Enabled Dynamically Updatable: No</p>
SQL Query Logging Threshold	SqlLongRunningThreshold	<p>Specifies the amount of time (in milliseconds) that an SQL query can run before a warning event is recorded in the application log for the server instance. If this threshold is exceeded, the following event is logged: Action completed successfully, but it took longer than the given threshold.</p> <p>Default: 1000 Dynamically Updatable: Yes</p>

Client Services Settings

The following table describes fields on the **Client Services** tab in the Business Central Server Administration tool.

SETTING	KEY NAME	DESCRIPTION
---------	----------	-------------

SETTING	KEY NAME	DESCRIPTION
Allowed File Types	ClientServicesAllowedFileTypes	<p>Specifies the file types that can be stored by the server when requested by the client. The value is a semicolon-separated list of the file name extensions. The server will not store other file types.</p> <p>Example values:</p> <ul style="list-style-type: none"> Blank or empty string (" "): The setting is disabled. File types will be limited based on Prohibited File Types setting instead. Asterisk (*): Specifies that all file types are allowed. List of file type extensions separated by semi-colons, for example <code>txt;xml;pdf</code> : Specifies that only .txt, .xml, and .pdf file types can be stored. <p>Trailing semicolons are ignored.</p> <p>Default: blank Dynamically Updatable: Yes</p>
Chunk Size	ClientServicesChunkSize	<p>The default size for a chunk of data that is transferred between Business Central Server and the Dynamics NAV Client connected to Business Central or Business Central Web Server, in kilobytes.</p> <p>The range of values is from 4 to 80.</p> <p>Default: 28 Dynamically Updatable: No</p>
Compression Threshold	ClientServicesCompressionThreshold	<p>The threshold in memory consumption at which Business Central Server starts compressing datasets, in kilobytes.</p> <p>Default: 64 Dynamically Updatable: No</p>
Enable Client Services	ClientServicesEnabled	<p>Specifies whether client services are enabled for this Business Central Server instance.</p> <p>Default: Enabled Dynamically Updatable: No</p>

SETTING	KEY NAME	DESCRIPTION
Exchange Auth. Metadata Location	ExchangeAuthenticationMetadataLocation	<p>Specifies the URLs for Microsoft Exchange authentication metadata document of the services or authorities that are trusted to sign Exchange identity tokens.</p> <p>This setting is used for setting up the Office Add-Ins for Outlook. For more information about the Office Add-ins, see Setting Up the Office Add-Ins for Outlook Integration</p> <p>The value is URL that is used to confirm the identity of the signing authority when using Exchange Authentication. The URL is compared to the Exchange authentication metadata document URL in the Exchange identity token. The scheme and host part of the two URLs must match to pass authentication. Paths in the URLs require only a partial match.</p> <p>With a multitenant server instance, the Exchange Auth. Metadata Location setting (if any) on the tenant will overrule the value of this setting.</p> <p>Value:</p> <ul style="list-style-type: none"> - One or more valid URLs. A URL must include the scheme, such as http:// or https://, and the host name. - Separate multiple URLs with a comma. - Wildcards (*) in URLs are supported. <p>Default: https://outlook.office365.com/ Dynamically Updatable: No</p>
Idle Client Timeout	ClientServicesIdleClientTimeout	<p>The interval of time that a Business Central Server client session can remain inactive before the session is dropped.</p> <p>Time interval format: [dd.]hh:mm:ss[.ff]</p> <p>Where: dd: days hh: hours mm: minutes ss: seconds ff: hundredths of a second</p> <p>Set Idle Client Timeout to equal or lower than the Keep Alive Interval, to enable Idle Client Timeout. You can also use MaxValue as a value to indicate no time-out.</p> <p>Default: MaxValue Dynamically Updatable: Yes</p>
Keep Alive Interval	ClientServicesKeepAliveInterval	<p>Specifies the time interval between keep-alive messages that are sent from the Dynamics NAV Client connected to Business Central to the server instance. This setting is used to keep inactive sessions alive until the time that is specified by the Idle Client Timeout setting expires. You should use a time interval that is less than the Idle Client Timeout setting, to hold the session constantly alive. For more information, see Understanding Session Timeouts.</p> <p>Time interval format: [dd.]hh:mm:ss[.ff]</p> <p>Default: 120 Dynamically Updatable: No</p>
Max Concurrent Connections	ClientServicesMaxConcurrentConnections	<p>Specifies the maximum number of concurrent client connections that the current Business Central Server instance accepts. You can use MaxValue as a value to indicate no limit.</p> <p>Default: 500 Dynamically Updatable: No</p>

SETTING	KEY NAME	DESCRIPTION
Max Items in Object Graph	ClientServicesMaxItemsInObjectGraph	<p>The maximum number of objects to serialize or deserialize.</p> <p>Default: 512 Dynamically Updatable: No</p>
Max Number of Orphaned Connections	ClientServicesMaxNumberOfOrphanedConnections	<p>Specifies the maximum number of orphaned connections to be kept alive at the same time for the time that is specified by ReconnectPeriod.</p> <p>A connection is orphaned when the client is involuntarily disconnected from Business Central Server.</p> <p>You can also use MaxValue as a value to indicate no limit.</p> <p>Default: 20 Dynamically Updatable: No</p>
Max Upload Size	ClientServicesMaxUploadSize	<p>The maximum size of files that can be uploaded to or downloaded from Business Central Server, in megabytes. Use this setting to avoid out-of-memory errors.</p> <p>Default: 150 Dynamically Updatable: No</p>
Operation Timeout	ClientServicesOperationTimeout	<p>The maximum time that Business Central Server can take to return a call from the client.</p> <p>Time span format: [dd.]hh:mm:ss[.ff]</p> <p>Where: dd: days hh: hours mm: minutes ss: seconds ff: hundredths of a second</p> <p>You can also use MaxValue as a value to indicate no time-out.</p> <p>Default: MaxValue Dynamically Updatable: No</p>
Port	ClientServicesPort	<p>The listening HTTP port for client services.</p> <p>Valid range: 1 - 65535 Default: 7046 Dynamically Updatable: No</p>

SETTING	KEY NAME	DESCRIPTION
Prohibited File Types	ClientServicesProhibitedFileTypes	<p>Specifies the file types that cannot be stored by the server when requested by the client. The value is a semicolon-separated list of the file name extensions. This setting is ignored if the Allowed File Types setting has a value.</p> <p>Example values:</p> <ul style="list-style-type: none"> Asterisk (*): All file types are prohibited. Blank or empty string (" "): The default value is used. Whitespace string (" "): All file types are allowed. List of file types separated by a semicolon, for example <code>txt;xml;pdf</code>: Prohibits the file types txt, xml and pdf. <p>Trailing semi-colons will be ignored.</p> <p>Default: ade;adp;app;asp;bas;bat;chm;cmd;com;cpl;csh;exe;fxp;gadget;hlp;hta;inf;ins;isp;its;js;jse;ksh;lnk;mad;maf;mag;mam;maq;mar;mas;mat;mau; ; maw;maw;mda;mdb;mde;mdt;mdw;mdz;msc;msi;msp;mst;ops;pcd;pif;prf;prg;pst;reg;scf;scr;sct;shb;shs;url;vb;vbe;vbs;vsmacros;vss;vst;vsw;ws;wsc;wsf;wsh Dynamically Updatable: Yes</p>
Protection Level	ClientServicesProtectionLevel	<p>Specifies the security services for protecting the data stream between clients and Business Central Server.</p> <p>All Dynamics NAV Client connected to Business Central clients connecting to the Business Central Server instance must have the same ProtectionLevel value in their ClientUserSettings.config files. For more information, see Configuring the Windows Client in the Dev and IT Pro Help for Microsoft Dynamics NAV 2018.</p> <p>For background information about transport security, see Understanding Protection Level (links to MSDN Library).</p> <p>Values: EncryptAndSign, Sign, None Default: EncryptAndSign Dynamically Updatable: No</p>
Reconnect Period	ClientServicesReconnectPeriod	<p>The time during which a client can reconnect to a running instance of Business Central Server.</p> <p>Time span format: [dd.]hh:mm:ss[.ff]</p> <p>Where: dd: days hh: hours mm: minutes ss: seconds ff: hundredths of a second</p> <p>You can also use MaxValue as a value to indicate no time limit.</p> <p>Default: 00:10:00 Dynamically Updatable: No</p>

SETTING	KEY NAME	DESCRIPTION
Token Signing Key	ClientServicesTokenSigningKey	<p>Specifies the signing information that you obtain from the Azure management portal. The parameter value is a 256-bit symmetric token signing key for use with Azure Access Control service (ACS). This parameter is relevant only when Credential Type, on the General tab, is set to AccessControlService.</p> <p>Default: EncryptAndSign Dynamically Updatable: No</p>
Use the Simplified Filters	UseSimplifiedFilters	<p>Specifies how the search on list pages behaves for plain text search filters (that is, filters that do not use search symbols like @ or *).</p> <p>If you enable this setting, the search uses a case sensitive and accent sensitive search to find fields that start with the provided filter text. For example, the search on man returns all records that include a field that starts with <i>man</i> (with a lowercase m), and the search on Man returns all records that include a field that starts with <i>Man</i> (with an uppercase M). Notice that you can get the same results by entering man* and Man* respectively.</p> <p>If the setting is disabled (which is default), the search on man and Man return the same results, which is all records that include fields that contain the text <i>man</i>, regardless of the case.</p> <p>For more information about the search, see Sorting, Searching, and Filtering Lists.</p> <p>Default: Not enabled Dynamically Updatable: No</p>
Web Client Base URL	PublicWebBaseUrl	<p>Specifies the root of the URLs that are used to open hyperlinks to pages and reports in the Business Central Web client. For example, you can change the value if you want to change the externally facing endpoint.</p> <p>The base URL must have the following syntax:</p> <pre>http[s]://[hostname]:[port]/[webserverinstance]</pre> <p>This field maps to the <code>PublicWebBaseUrl</code> setting in the CustomSettings.config file for the Business Central Server instance.</p> <p>Default: The URL of the Web client Dynamically Updatable: No</p>
Windows Client Base URL	PublicWinBaseUrl	<p>Specifies the root of the URLs that are used to open hyperlinks to pages and reports in the Dynamics NAV Client connected to Business Central. For example, you can change the value if you want to change the externally facing endpoint.</p> <p>The base URL must have the following syntax:</p> <pre>DynamicsNAV://[hostname]:[port]/[instance]/</pre> <p>This field maps to the <code>PublicWinBaseUrl</code> setting in the CustomSettings.config file for the Business Central Server instance.</p> <p>Default: The URL of the Windows client Dynamically Updatable: No</p>

SOAP Services Settings

The following table describes fields on the **SOAP Services** tab in the Business Central Server Administration tool.

SETTING	KEY NAME	DESCRIPTION
Enable SOAP Services	SOAPServicesEnabled	<p>Specifies whether SOAP web services are enabled for this Business Central Server instance.</p> <p>Default: Not enabled Dynamically Updatable: No</p>
Enable SSL	SOAPServicesSSLEnabled	<p>Specifies whether SSL (https) is enabled for the SOAP web service port. For more information, see Using Security Certificates with Business Central On-Premises.</p> <p>Default: Not enabled Dynamically Updatable: No</p>
Max Message Size	SOAPServicesMaxMsgSize	<p>The maximum permitted size of a SOAP web services request, in kilobytes.</p> <p>Important: This setting also pertains to OData web services.</p> <p>Default: 1024 Dynamically Updatable: No</p>
Port	SOAPServicesPort	<p>The listening HTTP port for SOAP web services.</p> <p>Valid range: 1 - 65535 Default: 7047 Dynamically Updatable: No</p>
SOAP Base URL	PublicSOAPBaseUrl	<p>Specifies the root of the URLs that are used to access SOAP web services. For example, you can change the value if you want to change the externally facing endpoint.</p> <p>The base URL must have the following syntax:</p> <p><code>http[s]://hostname:port/instance/WS/</code></p> <p>This field maps to the <code>PublicSOAPBaseUr1</code> setting in the CustomSettings.config file for the Business Central Server instance.</p> <p>Default: The SOAP URL for the server instance Dynamically Updatable: No</p>

OData Services Settings

The following table describes fields on the **OData Services** tab in the Business Central Server Administration tool.

SETTING	KEY NAME	DESCRIPTION
Enable Add-in Annotations	ODataEnableExcelAddInAnnotations	<p>Specifies whether Excel add-in annotations should be provided in OData metadata.</p> <p>Default: Enabled Dynamically Updatable: No</p>
Enable API Services	ApiServicesEnabled	<p>Specifies whether API web services are enabled for this server instance.</p> <p>Default: Not enabled Dynamically Updatable: No</p>
Enable OData Services	ODataServicesEnabled	<p>Specifies whether OData web services are enabled for this Business Central Server instance.</p> <p>Default: Enabled Dynamically Updatable: No</p>

SETTING	KEY NAME	DESCRIPTION
Enable SSL	ODataServicesSSLEnabled	<p>Specifies whether SSL (https) is enabled for the OData web service port. For more information, see Using Security Certificates with Business Central On-Premises.</p> <p>Default: Not enabled Dynamically Updatable: No</p>
Enable V3 Endpoint	ODataServicesV3EndpointEnabled	<p>Specifies whether the ODataV3 service endpoint will be enabled.</p> <p>Default: Enabled Dynamically Updatable: No</p>
Enable V4 Endpoint	ODataServicesV4EndpointEnabled	<p>Specifies whether the ODataV4 service endpoint will be enabled.</p> <p>Default: Enabled Dynamically Updatable: No</p>
Max Connections	ODataMaxConnections	<p>Specifies the maximum number of simultaneous OData requests on the server instance (for all tenants). When the limit is exceeded, a 429 (Too Many Requests) error occurs. If you do not want a limit, set the value 0.</p> <p>OData requests consume server instance resources and can affect the performance of the clients if the number of requests gets too large. This setting enables you to control the resources allocated for OData requests.</p> <p>Default: 0 Dynamically Updatable: Yes</p>
Max Connections Per Tenant	ODataMaxConnectionsPerTenant	<p>Specifies the maximum number of simultaneous OData requests per tenant. When the limit is exceeded, a 429 (Too Many Requests) error occurs. If you do not want a limit, set the value 0.</p> <p>If the server is not configured for multitenancy or only has a single tenant, then this setting does the same as the Max Connections (ODataMaxConnections) setting.</p> <p>OData requests consume server instance resources and can affect the performance of the clients if the number of requests gets too large. This setting enables you to control the resources allocated for OData requests.</p> <p>Default: 0 Dynamically Updatable: Yes</p>
Max Page Size	ODataServicesMaxPageSize	<p>Specifies the maximum number of entities returned per page of OData results. For more information, see Server-Driven Paging in OData Web Services.</p> <p>Default: 1000 Dynamically Updatable: No</p>

SETTING	KEY NAME	DESCRIPTION
OData Base URL	PublicODataBaseUrl	<p>Specifies the root of the URLs that are used to access OData web services. For example, you can change the value if you want to change the externally facing endpoint.</p> <p>The base URL must have the following syntax:</p> <p><code>http[s]://hostname:port/instance/OData/</code></p> <p>This field maps to the <code>PublicODataBaseUrl</code> setting in the CustomSettings.config file for the Business Central Server instance.</p> <p>Default: The OData URL for the server instance Dynamically Updatable: No</p>
Port	ODataServicesPort	<p>The listening HTTP port for Business Central OData web services.</p> <p>Valid range: 1 - 65535 Default: 7048 Dynamically Updatable: No</p>
Timeout	ODataServicesOperationTimeout	<p>Specifies the maximum amount of time that the server instance can allocate to a single OData request. When the limit is exceeded, a 408 (Request Timeout) error occurs.</p> <p>If you do not want a limit, set the value to <code>MaxValue</code>.</p> <p>Time format: hh:mm:ss Default: 00:05:00 Dynamically Updatable: Yes</p>

IMPORTANT

The maximum permitted size of an OData web services request is specified by the **Max Message Size** option on the **SOAP Services** tab.

NAS Services Settings

The following table describes fields on the **NAS Services** tab in the Business Central Server Administration tool.

NOTE

Instead of using NAS services, we recommend that you use the Task Scheduler (see [Task Scheduler](#)). If you decide to use NAS, and want to read more about its configuration, see [Configuring NAS Services](#) in the Dev and IT Pro Help for Microsoft Dynamics NAV 2018.

SETTING	KEY NAME	DESCRIPTION
Enable Debugging	NAServicesEnableDebugging	<p>Specifies if the Business Central Debugger must attach to the NAS Services session. When this is enabled, the NAS Services session waits 60 seconds before the first AL statement is run.</p> <p>Default: Not enabled Dynamically Updatable: No</p>

SETTING	KEY NAME	DESCRIPTION
Run NAS Services with Admin Rights	NASServicesRunWithAdminRights	<p>Specifies whether NAS services run operations with administrator rights instead of the rights granted to the Business Central Server service account.</p> <p>If you select this setting, NAS services will have full permissions in Business Central, similar to the permissions that are granted by the SUPER permission set. The Business Central Server service account is not required to be set up as a user in Business Central.</p> <p>If you clear this setting, the Business Central Server service account must be added as a user in Business Central and assigned the permissions that are required to perform the operations.</p> <p>Default: Not enabled Dynamically Updatable: No</p>
Startup Argument	NASServicesStartupArgument	<p>Specifies a string argument that will be used when NAS services start. The argument typically specifies an application type, sometimes with additional configuration information.</p> <p>Example value: <code>"OSYNCH"</code></p> <p>Default: Dynamically Updatable: No</p>
Startup Codeunit	NASServicesStartupCodeunit	<p>Specifies the codeunit that contains the method that will be called by the NASStartupMethod setting.</p> <p>Example values:</p> <p>0 When NASStartupCodeunit is set to 0, NAS Services do not start. This is the default value.</p> <p>55 When NAS services start, they run the trigger specified by the NAS Startup Method in codeunit 55.</p> <p>Note: When the codeunit specified by NASStartupCodeunit is a single instance codeunit, the NAS service session will remain alive even after you run all code in the specified NASStartupMethod.</p> <p>Default: Dynamically Updatable: No</p>
Startup Method	NASServicesStartupMethod	<p>Specifies the method that will be called in the NASStartupCodeunit.</p> <p>Example values:</p> <p><code>""</code></p> <p>If no start method is specified (null string), the OnRun trigger is called.</p> <p>StartNAS NAS services runs the StartNAS method in the NAS Startup Codeunit.</p> <p>Default: Dynamically Updatable: No</p>

Management Services Settings

The following table describes fields on the **Management Services** tab in the Business Central Server Administration tool.

SETTING	KEY NAME	DESCRIPTION
Enable Management Services	ManagementServicesEnabled	Specifies whether Business Central Server Administration tool is enabled for this Business Central Server instance. Default: Enabled Dynamically Updatable: No
Port	ManagementServicesPort	The listening TCP port for the Business Central Server Administration tool. Valid range: 1 - 65535 Default: 7045 Dynamically Updatable: No

Azure Key Vault Encryption Provider Tab Settings

The following table describes fields on the **Azure Key Vault Encryption Provider** tab in the Business Central Server Administration tool.

NOTE

These settings are used when you want to use Azure Key Vault to help encrypt data in the database. If you want to use Azure Key Vault to encrypt the connection between Business Central Server and an Azure SQL database, you must store that key in the database.

SETTING	KEY NAME	DESCRIPTION
Client Certificate Store Location	AzureKeyVaultClientCertificateStoreLocation	Specifies the location of the certificate store for the Key Vault client certificate if you set the Encryption Key Provider field to AzureKeyVault . LocalMachine specifies that the certificate is stored in a certificate store for the computer that the Business Central Server is running on. CurrentUser specifies that the certificate is stored in a certificate store for your account on the computer that the Business Central Server is running on. Default: LocalMachine Dynamically Updatable: No
Client Certificate Store Name	AzureKeyVaultClientCertificateStoreName	Specifies the certificate store where the Key Vault client certificate is stored. Default: My Dynamically Updatable: No
Client Certificate Thumbprint	AzureKeyVaultClientCertificateThumbprint	Specifies the thumbprint of the Key Vault client certificate Default: My Dynamically Updatable: No
Client ID	AzureKeyVaultClientId	Specifies the unique identifier (GUID) of the Key Vault client application in Microsoft Azure. Default: 00000000-0000-0000-0000-000000000000 Dynamically Updatable: No
Key URI	AzureKeyVaultKeyUri	Specifies the URI of the key in the Key Vault encryption provider setup. Default: Dynamically Updatable: No

Azure Active Directory (Azure AD) Settings

The following table describes fields on the **Azure Active Directory (Azure AD)** tab in the Business Central Server Administration tool.

The settings in this tab configure the Business Central Server instance to use Azure AD authentication. The settings are only relevant when the server instance is configured Access Control Service, that is, when the **Credential Type** is set to **AccessControlService**. For more information about authenticating using Azure AD, see [Authenticating Users with Azure Active Directory](#).

SETTING	KEY NAME	DESCRIPTION
Application Client Certificate Thumbprint	AzureActiveDirectoryClientCertificateThumbprint	<p>Specifies the thumbprint of the x509 certificate that is used with the Azure AD application client for authentication.</p> <p>A public certificate file (.cer) must be installed on the application client and associated with an Azure AD service principal.</p> <p>A private certificate file (.pfx) must be installed on the computer on which the Business Central Server instance is installed. The server instance service account must have access to the private key of that certificate.</p> <p>Default: Dynamically Updatable: No</p>
Application Client ID	AzureActiveDirectoryClientId	<p>Specifies the ID of the application tenant. The ID is used when accessing data in Azure AD.</p> <p>The authentication token for communicating with Azure AD should be retrieved by specifying the Application Client Certificate Thumbprint, with a fallback to use the Application Client Secret.</p> <p>Default: 00000000-0000-0000-0000-000000000000 Dynamically Updatable: No</p>
Application Client Secret	AzureActiveDirectoryClientSecret	<p>Specifies the secret to use with Application Client ID for Azure AD authentication.</p> <p>Default: Dynamically Updatable: No</p>
Azure AD App ID URI	AppIdUri	<p>Specifies the App ID URI that is registered for Business Central in the Microsoft Azure Active Directory (Azure AD). You use this setting to configure Business Central web services for OAuth authentication, specifically when the <i>Credential Type</i> setting is AccessControlService. It is used to validate the security tokens that the server instance receives in SOAP and OData calls.</p> <p>The App ID URI is a logical identifier and does not have to represent a valid location, although it is common practice to use the physical URL of the Business Central web service.</p> <p>The App ID URI is typically the same as the value of <i>wtrealm</i> parameter of the ACSUri setting that is included in the ClientUserSettings.config file for the Dynamics NAV Client connected to Business Central.</p> <p>An example of an App ID URI is <i>https://localhost:7047/</i>.</p> <p>For more information about how to use the Azure Active Directory App ID URI with OAuth authentication, see Using OAuth to Authenticate Web Services (Odata and SOAP).</p> <p>Default: Dynamically Updatable: No</p>

SETTING	KEY NAME	DESCRIPTION
Enable Membership Entitlement	EnableMembershipEntitlement	<p>Configures the server instance to use membership entitlement for controlling access the Business Central.</p> <p>This setting is typically only used for software as a service (SaaS) solutions.</p> <p>Default: Dynamically Updatable: No</p>
Excel add-in AAD client app ID	ExcelAddInAzureActiveDirectoryClientId	<p>This setting is used to set up the Excel Add-in that enables users to use Excel to modify and update Business Central data.</p> <p>The setting specifies the client ID of the Azure AD tenant that is used for the Excel add-in. The Excel add-in requires a separate Azure AD tenant. For more information, see Configuring the Excel Add-In.</p> <p>Default: Dynamically Updatable: No</p>
Extended Security Token Lifetime	ExtendedSecurityTokenLifetime	<p>Specifies the number of hours that are added to the lifetime of Azure AD security tokens, which are used to authenticate client users. When the lifetime expires, the client is disconnected from the server instance. An event with a message such as "The SAML2 token is invalid because its validity period ended." is recorded in the event log for the server instance. In general, the lifetime of security tokens is 1 hour.</p> <p>Valid range: 0 to 24 hours Default: 0 Dynamically Updatable: No</p>
Valid Audiences	ValidAudiences	<p>Specifies the allowed audiences for Azure AD authentication. This setting is used to authenticate other Azure AD applications that will communicate with the server instance.</p> <p>The value is a semicolon-separated list of audiences. You specify an audience by using the App URI ID or App ID that is assigned to the application in Azure AD.</p> <p>Default: Dynamically Updatable: No</p>
WS-Federation Login Endpoint	WSFederationLoginEndpoint	<p>Specifies the URL for the federation sign-on page that Business Central redirects to when configured for single sign-on.</p> <p>You must specify a URL in the following format:</p> <div> https://login.microsoftonline.com/[AADTENANTID]/wsfederation/...?wa=wsignin1.0%26wtrealm=...%26wreply=... </div> <p>The placeholder [AADTENANTID] represents the GUID of your Azure AD tenant. If the server instance has to support multiple Azure AD tenants, then the Azure AD Tenant ID parameter that is specified when mounting a tenant replaces the placeholder.</p> <p>Default: Dynamically Updatable: No</p>

SETTING	KEY NAME	DESCRIPTION
WS-Federation Metadata Location	ClientServicesFederationMetadataLocation	<p>Specifies the URL for the federation metadata document that describes the configuration information for your Azure AD tenant. The federation metadata document is used to validate the security tokens that the Business Central Web client and Business Central Tablet client receive, and to establish a trust relationship with between Business Central and an application that you have added to Azure AD.</p> <p>You must specify a URL in the following format:</p> <pre>https://login.microsoftonline.com/[AADTENANTID]/FederationMetadata.xml</pre> <p>The placeholder [AADTENANTID] represents the GUID of your Azure AD tenant. If the server instance has to support multiple Azure AD tenants, then the Azure AD Tenant ID parameter that is specified when mounting a tenant replaces the placeholder.</p> <p>This parameter is relevant only when Credential Type, on the General tab, is set to AccessControlService. For more information, see Authenticating Users with Azure Active Directory.</p> <p>Default: Dynamically Updatable: No</p>

Task Scheduler Settings

The following table describes fields on the **Task Scheduler** tab in the Business Central Server Administration tool.

The task scheduler processes jobs and other processes on a scheduled basis. For more information about task scheduler, see [Task Scheduler](#).

SETTING	KEY NAME	DESCRIPTION
Enable Task Scheduler	EnableTaskScheduler	<p>Specifies whether the server instance starts with the task scheduler enabled.</p> <p>If this option is enabled, the server instance will process scheduled tasks.</p> <p>Default: Enabled Dynamically Updatable: No</p>
Maximum Concurrent Running Tasks	TaskSchedulerMaximumConcurrentRunningTasks	<p>Specifies the maximum number of tasks that can run simultaneously on the server instance.</p> <p>The value that you specify will depend on the hardware (CPUs) of the deployment environment and what you want to prioritize: client performance or scheduled tasks (such as job queue entries). The setting is particularly relevant when the server instance is used for both scheduled tasks and client services. If there are many jobs running at the same time, you might experience that the response time for clients gets slower. In which case, you could decrease the value. However, if the value is too low, it might take longer than desired for scheduled tasks to process. When you have a dedicated server instance for scheduled tasks, this setting is less important with respect to client performance.</p> <p>Default: 10 Dynamically Updatable: Yes</p>

SETTING	KEY NAME	DESCRIPTION
System Task Start Time	TaskSchedulerSystemTaskStartTime	<p>Specifies the time of day after which system tasks can start. The time is based on the time zone of the computer that is running the server instance.</p> <p>The value has the format HH:MM:SS.</p> <p>Default: 00:00:00 Dynamically Updatable: Yes</p>
System Task End Time	TaskSchedulerSystemTaskEndTime	<p>Specifies the time of day after which system tasks cannot start. The time is based on the time zone of the computer that is running the server instance.</p> <p>The value has the format HH:MM:SS.</p> <p>Default: 23:59:59 Dynamically Updatable: Yes</p>

Reports Settings

The following table describes fields on the **Reports** tab in the Business Central Server Administration tool.

SETTING	KEY NAME	DESCRIPTION
Enable Application Domain Isolation	ReportAppDomainIsolation	<p>Specifies whether application domain isolation is used for rendering custom RDLC layouts. This setting pertains to on-premise installations only.</p> <p>Enabling application domain isolation provides a more secure and reliable environment for processing custom RDLC layouts; however, it can considerably increase the time it takes to render reports. Disabling application domain isolation can improve the rendering time but might have a negative impact on security and reliability.</p> <p>Default: Enabled Dynamically Updatable: No</p>
Enable Save as Excel on Request Pages of RDLC-layout Reports	EnableSaveToExcelForRdlcReports	<p>Specifies whether users can open or save a report as an Microsoft Excel document if the report uses an RDLC layout.</p> <p>If you clear this check box, the Excel option is removed from the Print menu on the request page.</p> <p>Default: Enabled Dynamically Updatable: No</p>
Enable Save as Word on Request Pages of RDLC-layout Reports	EnableSaveToWordForRdlcReports	<p>Specifies whether users can open or save a report as a Microsoft Word document if the report uses an RDLC layout.</p> <p>If you clear this check box, the Word option is removed from the Print menu on the request page.</p> <p>Default: Enabled Dynamically Updatable: No</p>
Enable Save from Report Preview	EnableSaveFromReportPreview	<p>Specifies whether users can save a report as a PDF, Microsoft Word, or Microsoft Excel document from the report preview window.</p> <p>If you clear this check box, the Save As icon is removed from the report preview window.</p> <p>Default: Enabled Dynamically Updatable: No</p>

SETTING	KEY NAME	DESCRIPTION
Report PDF Font Embedding	ReportPDFFontEmbedding	<p>Specifies whether fonts are embedded in PDF files that are generated for reports when the report uses an RDLC report layout at runtime. This setting applies when reports are run and saved as PDF files on the client (from the report request page or print preview window) or on the server instance (by the SAVEAS function or SAVEASPDF function in AL code).</p> <p>Note: This setting does not apply when a report uses a Word report layout at runtime.</p> <p>Embedding fonts in a PDF of a report makes sure that the PDF will use the same fonts as the original file, regardless of where the PDF is opened and which fonts are installed on the computer. However, embedding fonts can significantly increase the size of the PDF files. By disabling font embedding, you can decrease the size of the report PDF files.</p> <p>Note: This is a global setting for font embedding in report PDF files. You can override this setting on a report basis by the specifying the PDFFontEmbedding property.</p> <p>Default: Enabled Dynamically Updatable: No</p>
<i>not available</i>	CalculateBestPaperSizeForReportPrinting	<p>Determines the paper size to use when printing reports from the client.</p> <p>If set to <code>true</code>, the system calculates which of the available paper sizes on the printer is best suited for printing, and then uses that paper size.</p> <p>If set to <code>false</code>, the printer's default paper size is used.</p> <p>Default: true</p>

Development Settings

The following table describes fields on the **Development** tab in the Business Central Server Administration tool.

SETTING	KEY NAME	DESCRIPTION
Allowed Extension Target Level	ExtensionAllowedTargetLevel	<p>Specifies the allowed target level when publishing extensions. The options are Internal, Extension, Solution, and Personalization.</p> <p>- If you specify the Internal option, the allowed compilation target is set to everything on-premises. The Internal setting allows using all restricted APIs. The <code>target</code> setting in the <code>app.json</code> file must also be set to Internal. For more information, see JSON Files.</p> <p>- If you specify the Extension option, the allowed extension target level is set to SaaS.</p> <p>- By adding the setting <code>"target": "Extension"</code> in the manifest enables you to submit the extension to AppSource.</p> <p>The Personalization and Solution settings are currently for internal use only.</p> <p>Note: It is recommended to use either Internal or Extension options to set Allowed Extension Target Level.</p> <p>Default: Internal Dynamically Updatable: No</p>

SETTING	KEY NAME	DESCRIPTION
Debugger – Long Running SQL Statements Threshold	LongRunningSqlStatementsInDebuggerThreshold	<p>Specifies the amount of time (in milliseconds) that an SQL query can run before it is logged in debugger telemetry.</p> <p>Default: Enabled Dynamically Updatable: Yes</p>
Debugger - Number of SQL Statements to Show	AmountOfSqlStatementsInDebugger	<p>Specifies the amount of SQL statements used in the debugger; the higher number you choose, the more data will be sent to the debugger.</p> <p>Default: Enabled Dynamically Updatable: Yes</p>
Debugger - Show Long Running SQL Statements	EnableLongRunningSqlStatementsInDebugger	<p>Specifies whether long running SQL statements will be shown in the debugger.</p> <p>Default: Enabled Dynamically Updatable: Yes</p>
Debugger - Show SQL Statements	EnableSqlInformationDebugger	<p>Specifies whether the debugger should collect the last used SQL statements and show them in the debugger.</p> <p>Default: Enabled Dynamically Updatable: Yes</p>
Debugging Allowed	DebuggingAllowed	<p>Specifies whether AL debugging is allowed for this Business Central Server instance.</p> <p>Default: Enabled Dynamically Updatable: No</p>
Enable Debugging	EnableDebugging	<p>Specifies whether the Business Central Server instance starts with debugging enabled.</p> <p>If this option is enabled, the following occurs:</p> <p>When the client first connects, all C# files for the application are generated. C# files persist between Business Central Server restarts. Application objects are compiled with debug information.</p> <p>Default: Not enabled Dynamically Updatable: No</p>
Enable Developer Service Endpoint	DeveloperServicesEnabled	<p>Specifies whether the Developer service endpoint is enabled. This setting must be enabled to publish extensions and download symbols.</p> <ul style="list-style-type: none"> - If the check box is selected, the extension can be published to the allowed extension target level. - If the check box is not selected, the extension cannot be published. - This setting can also be controlled from the .xml file. <p>Default: Not enabled (check box is cleared) Dynamically Updatable: No</p>
Enable Loading Application Symbol References at Server Startup	EnableSymbolLoadingAtServerStartup	<p>Specifies whether application symbol references should be loaded at server startup. This setting must be enabled to allow any symbol generation. If the setting is not enabled, the generatesymbolreference setting does not have any effect. For more information, see Running C/SIDE and AL Side-by-Side.</p> <p>Default: Not enabled (check box is cleared) Dynamically Updatable: No</p>

SETTING	KEY NAME	DESCRIPTION
Enable SSL	DeveloperServicesSSLEnabled	<p>Specifies whether SSL (HTTPS) is enabled for the developer web service port.</p> <p>SSL (Secure Sockets Layer) secures the connection for the web services.</p> <ul style="list-style-type: none"> - If the check box is selected, then SSL is enabled. - If the check box is not selected, the developer web service port cannot establish a secure connection. <p>Default: Not enabled (check box is cleared) Dynamically Updatable: No</p>
HttpClient AL Function Maximum Timeout	NavHttpClientMaxTimeout	<p>Specifies the maximum allowed timeout value that can be set for the HttpClient Timeout AL function.</p> <p>The value has the format HH:MM:SS.</p> <p>Default: 00:05:00 Dynamically Updatable: Yes</p>
HttpClient AL Function Response Size	NavHttpClientMaxResponseContentSize	<p>Specifies the maximum size in megabytes of a response buffer used by the HttpClient AL function.</p> <p>The maximum allowed extension size can be adjusted based on the HttpClient AL Function Maximum Timeout setting.</p> <p>Default: 15 Dynamically Updatable: Yes</p>
Port	DeveloperServicesPort	<p>The listening HTTP port for Microsoft Dynamics NAV Developer web services.</p> <p>Valid range: 1 - 65535 Default: 7049 Dynamically Updatable: No</p>

Compatibility Settings

The following table describes settings that you can adjust for compatibility with other systems. In most cases, we do not recommend that you change these settings from their default values.

SETTING	KEY NAME	DESCRIPTION
Security Protocol	SecurityProtocol	<p>Specifies the default security protocol level for the server instance.</p> <p>Values: Ssl3, Tls, Tls11, Tls12, SystemDefault. For more information about these values, see SecurityProtocolType Enum.</p> <p>Default: Tls12 Dynamically Updatable: No</p>
Use Client Timestamp For Report Execution Timestamp	ReplaceReportExecutionTimeWithClientTime	<p>Specifies whether to replace the report execution timestamp with the client timestamp instead of the server instance timestamp.</p> <p>Default: Enabled (check box is selected) Dynamically Updatable: No</p>

SETTING	KEY NAME	DESCRIPTION
Use FIND('-') to Populate Pages Instead of FIND('=><')	UseFindMinusWhenPopulatingPage	Specifies whether pages are initially populated by using FIND('-') instead of FIND('=><'). This setting is relevant to pages that display lists in descending order. Enabling this setting ensures that the first record, instead of the last record, is in focus when the page opens. Pages that use the OnFindRecord trigger will ignore this setting and always use FIND('=><'). Default: Enabled (check box is selected) Dynamically Updatable: No

Using Business Central Administration Shell to Modify Server Instance Settings

The Business Central Administration Shell includes several `Set-` cmdlets that enable you to create and modify Business Central Server instances.

The main cmdlet for configuring a server instance is the `Set-NAVServerConfiguration` cmdlet. You can use this cmdlet to change any of the configuration settings that are listed in the previous sections. To change a configuration setting, you set the `-KeyName` parameter to the **Key Name** that corresponds to the setting, and set the `-KeyValue` parameter to the new value. For example, you can change the value for `DatabaseServer` to `DatabaseServer.Domain.Com` for the server instance named `MyInstance` by executing this cmdlet:

```
Set-NAVServerConfiguration -ServerInstance "MyInstance" -KeyName "DatabaseServer" -KeyValue "DatabaseServer.Domain.Com"
```

Modifying dynamically updatable settings

For dynamically updatable settings, use the `-ApplyTo` parameter to specify how to apply the change. The change can be written directly to the configuration file (`CustomSettings.config`) and/or applied to the current server instance state. The option you choose will determine whether a server instance restart is required for the change to take effect. The parameter has three options, as described in the following table:

OPTION	DESCRIPTION
ConfigFile	Saves the change to the configuration file of the server instance. The change will not take effect until the server instance is restarted.
Memory	Applies the change only to the server instance's current state. The changes take effect immediately, without having to restart the server instance. The change is stored in memory, so the next time the server instance is restarted, it reverts to the setting in the configuration file.
All	Applies the change to the server instance's current setting state (in memory) and to the configuration file. The changes take effect immediately, without having to restart the server instance.

For example, the following command sets the value for the `MaxStreamReadSize` key to `42424242`, without having to restart the server instance.

```
Set-NAVServerConfiguration -ServerInstanceMyInstance -KeyName MaxStreamReadSize -KeyValue 42424242 -ApplyTo Memory
```

For more information about running the Business Central Administration Shell, see [Microsoft Dynamics NAV Windows PowerShell Cmdlets](#).

See Also

[w](#)
[Business Central Server Administration Tool](#)
[Enhancing Business Central Server Security](#)
[Business Central Windows PowerShell Cmdlets](#)
[Configuring Help](#)
[Hiding UI Elements](#)
[Debugging](#)

Configuring Business Central Web Server Instances

6/17/2019 • 11 minutes to read

Accessing Business Central from the Business Central Web client or App requires a Business Central Web Server instance on IIS. You create a Business Central Web Server instance for the Business Central Web Server by using the Business Central Setup to install the Business Central Web Server or by running the [New-NAVWebServerInstance cmdlet](#). When you set up a web server instance, you are configuring the connection from the Business Central Web Server to the Business Central Server instance. The connection settings, along with several other configuration settings, are saved in a configuration file for the web server instance.

About the configuration file

The configuration file for the web server instances is a json file type called **navsettings.json**. The navsettings.json file is a Java Script Object Notification file type that is similar to files that use the XML file format.

After installation, you can change the configuration by modifying the navsettings.json. There are two ways to modify this file: directly or using PowerShell.

Where to find the navsettings.json file

The navsettings.json file is stored in the physical path of the web server instance, which is by default is `%systemroot%\inetpub\wwwroot\[WebServerInstanceName]`.

`[WebServerInstanceName]` corresponds to the name (alias) of the web server instance in IIS, for example, `c:\inetpub\wwwroot\BC140`.

Modify the navsettings.json file directly

1. Open the navsettings.json in any text or code editor, such as Notepad or Visual Studio Code.

Each setting is defined by a key-value pair.

- In the navsettings.json file, a setting has the format:

```
"keyname": "keyvalue",
```

The `keyname` is the name of the configuration setting and the `keyvalue` is the value.

For example, the configuration setting that specifies the Windows credential type for authenticating users is:

```
"ClientServicesCredentialType": "Windows",
```

Include values in double quotes.

2. Find the configuration settings that you want to change, and then change the values.

See the [Settings](#) section for a description of each setting.

3. When you are done making changes, save the file.
4. Restart the Business Central Web Server instance for the changes to take effect.

For example, in IIS Manager, in the **Connections** pane, select website node for Business Central Web Server, and then in the **Actions** pane, choose **Restart**. Or, from your desktop, run `iisreset`.

Modify the navsettings.json file by using the Set-NAVWebServerInstanceConfiguration cmdlet

The PowerShell script module **NAVWebClientManagement.psm1** includes the [Set-NAVWebServerInstanceConfiguration cmdlet](#) that enables you to configure a web server instance.

1. Depending on your installation, run the Dynamics NAV Development Shell or Windows PowerShell as an administrator.

For more information, see [Get started with the Business Central Web Server cmdlets](#).

2. For each setting that you want to change, at the command prompt, run the following command:

```
Set-NAVWebServerInstanceConfiguration -Server [MyComputer] -ServerInstance [ServerInstanceName] -WebServerInstance [MyBCWebServerInstance] -KeyName [Setting] -KeyValue [Value]
```

Replace:

- `[MyComputer]` with the name of the computer that is running the Business Central Server
- `[ServerInstanceName]` with the name of the server instance, such as **BC140**.
- `[MyBCWebServerInstance]` with the name of the web server instance for the Business Central Web Server.
- `[KeyName]` with the name of the setting. Refer to the next section in this article.
- `[KeyValue]` with the new value of the setting.

Settings in the navsettings.json

The navsettings.json has the following structure, where settings are included under one of two root elements:

`NAVWebSettings` and `ApplicationIdSettings`:

```
{
  "NAVWebSettings": {
    "//ServerInstance": "Name of the Business Central Server instance to connect to (for client) or listen on (for server).",
    "ServerInstance": "BC140",
    [...more keys]
  },
  "ApplicationIdSettings": {
    "BlogLink": "https://myBCBlog.com",
    [...more keys]
  }
}
```

`//` indicates a comment that provides help for the setting, and has no affect on the Web Server instance.

The following table describes the settings that are available in the navsettings.json for each root element. If you do not see a setting in the file, this is because some settings are not automatically included as a key in the file. For these settings, you can add the key manually. If you do not add the key, the default value of the setting is used.


IMPORTANT

If modifying the file directly, place values in double quotes `""`.

NAVWebSettings element settings

SETTING/KEYNAME	DESCRIPTION
AllowedFrameAncestors	<p>Specifies the host name of any web sites in which the Business Central Web client or parts of are embedded. By default, the Business Central Web Server will not allow a website to display it inside an iframe unless the website is hosted on the same web server. This value of this setting is a comma-separated list of host names (URIs). Wildcard names are accepted. For example:</p> <pre>https:mysite.sharepoint.com, https:*.myportal.com</pre>
GlobalEndPoints	<p>Specifies the comma-separated list of global endpoints that are allowed to call this website. The values must include http scheme and fully qualified domain name (FDQN), such as <code>https://financials.microsoft.com</code>.</p> <p>Default value: none</p>
LoadScriptsFromCdn	<p>Specifies whether to load scripts from Content Distribution Networks (CDNs). This only applies to scripts that are available from a CDN, like jQuery.</p> <p>If set to <code>false</code>, scripts will be loaded from the Web server, which is useful in, for example, an intranet scenario where there is no internet access.</p> <p>Default value: <code>true</code></p>
AllowNtlm	<p>Specifies whether NT LAN Manager (NTLM) fallback is permitted for authentication.</p> <p>To require Kerberos authentication, set this value to <code>false</code>.</p> <p>Values: <code>true</code>, <code>false</code></p> <p>Default value: <code>true</code></p>
ClientServicesChunkSize	<p>Sets the maximum size, in kilobytes, of a data chunk that is transmitted between Business Central Web Server and Business Central Server. Data that is transmitted between Business Central Web Server and Business Central Server is broken down into smaller units called chunks, and then reassembled when it reaches its destination.</p> <p>Values: From 4 to 80.</p> <p>Default value: 28</p>

SETTING/KEYNAME	DESCRIPTION
ClientServicesCompressionThreshold	<p>Sets the threshold in memory consumption at which Business Central Web Server starts compressing data sets. This limits amount of consumed memory. The value is in kilobytes.</p> <p>Default value: 64</p>
ClientServicesProtectionLevel	<p>Specifies the security services used to protect the data stream between the Business Central Web Server and Business Central Server. This value must match the value that is specified in the Business Central Server configuration file. For more information, see Configuring Business Central Server.</p> <p>Values: EncryptAndSign, Sign, None</p> <p>Default value: EncryptAndSign</p>
Server	<p>Specifies the name of the computer that is running the Business Central Server.</p> <p>Default value: localhost</p>
ServerInstance	<p>Specifies the name of the Business Central Server instance that the Business Central Web Server connects to.</p> <p>Default value: BC140</p>
ClientServicesCredentialType	<p>Specifies the authorization mechanism that is used to authenticate users who try to connect to the Business Central Web Server. For more information, see Authentication and User Credential Type.</p> <p>The credential type must match the credential type configured for the Business Central Server instance that the Business Central Web Server connects to. For information about how to set up the credential type on Business Central Server, see Configuring Business Central Server.</p> <p>Values: Windows, UserName, NavUserPassword, AccessControlService</p> <p>Default value: Windows</p>
ClientServicesPort	<p>Specifies the TCP port for the Business Central Server. This is part of the Business Central Server's URL.</p> <p>Values: 1-65535</p> <p>Default value: 7046</p>
ManagementServicesPort	<p>The listening TCP port for the Business Central management endpoint.</p> <p>Valid range: 1-65535</p> <p>Default value: 7045</p>

SETTING/KEYNAME	DESCRIPTION
ServicePrincipalNameRequired	<p>If this parameter is set to <code>true</code>, then the Business Central Web Server can only connect to a Business Central Server instance that has been associated with a service principal name (SPN).</p> <p>If this parameter is set to <code>false</code>, then the Business Central Web Server attempts to connect to the configured Business Central Server service, regardless of whether that service is associated with an SPN.</p> <p>For more information about SPNs, see Configure Delegation.</p> <p>Default: <code>false</code></p>
SessionTimeout	<p>Specifies the maximum time that a connection between the Business Central Web Server and the Business Central Server can remain idle before the session is stopped.</p> <p>In the Business Central Web Server, this setting determines how long an open Business Central page or report can remain inactive before it closes. For example, when the SessionTimeout is set to 20 minutes, if a user does not take any action on a page within 20 minutes, then the page closes and it is replaced with the following message: The page has expired and the content cannot be displayed.</p> <p>The time span has the format [dd.]hh:mm:ss[.ff]:</p> <ul style="list-style-type: none"> - dd is the number of days - hh is the number of hours - mm is the number of minutes - ss is the number of seconds - ff is fractions of a second <p>Default value: 00:20:00</p>
RequireSsl	<p>Specifies whether SSL (https) is required. If the value is set to <code>true</code> all cookies will be marked with a <code>\u0027secure\u0027</code> attribute. If SSL is enable on the Web server, you should set this to <code>true</code>.</p> <p>Values: <code>true</code>, <code>false</code></p> <p>Default value: <code>false</code></p>
ShowPageSearch	<p>Specifies whether to show the  Tell me what you want to do icon in the Business Central header. This feature lets users find Business Central objects, such as pages, reports, and actions.</p> <p>If you do not want to show the Tell me what you want to do icon, then set the parameter to <code>false</code>.</p> <p>Default value: <code>true</code></p>

SETTING/KEYNAME	DESCRIPTION
UnknownSpnHint	<p>Specifies whether to use a server principal name when establishing the connection between the Business Central Web Server and Business Central Server. This setting is used to authenticate the Business Central Server, and it prevents the Business Central Web Server from restarting when it connects to Business Central Server for the first time. You set values that are based on the value of the ServicePrincipalNameRequired key.</p> <p>Value: The value has the following format.</p> <p>(net.tcp://BCServer:Port/ServerInstance/Service)=NoSpn SPN</p> <ul style="list-style-type: none"> - BCServer is the name of the computer that is running the Business Central Server. - Port is the port number on which the Business Central Server is running. - ServerInstance is the name of the Business Central Server instance. - NoSpn SPN specifies whether to use an SPN. If the ServicePrincipalNameRequired key is set to <code>false</code>, then set this value to NoSpn. If the ServicePrincipalNameRequired key is set to <code>true</code>, then set this value to Spn. <p>Default value: (net.tcp://localhost:7046/BC140/Service)=NoSpn</p> <p>If you set this key to the incorrect value, then during startup, the Business Central Web Server will automatically determine a correct value. This will cause the Business Central Web Server to restart. Note: For most installations, you do not have to change this value. Unlike the Dynamics NAV Client connected to Business Central, this setting is not updated automatically. If you want to change the default value, then you must change it manually.</p>
DnsIdentity	<p>Specifies the subject name or common name of the service certificate for Business Central Server.</p> <p>This parameter is only relevant when the ClientServicesCredentialType is set to UserName, NavUserPassword, or AccessControlService, which requires that security certificates are used on the Business Central Web Server and Business Central Server to protect communication. Note: You can find the subject name by opening the certificate in the Certificates snap-in for Microsoft Management Console (MMC) on the computer that is running Business Central Web Server or Business Central Server.</p> <p>For more information, see Authentication and User Credential Type.</p> <p>Value: The subject name of the certificate.</p> <p>Default value: none</p>

SETTING/KEYNAME	DESCRIPTION
AuthenticateServer	<p>Specifies whether to authenticate the server by enabling service identity checks on protocol between the Web server and the Business Central Server instance.</p> <p>Values: <code>true</code>, <code>false</code> Default value: <code>true</code></p>
HelpServer	<p>Specifies the name of the Business Central Help Server if the deployment uses Help Server. If the deployment uses an online library, remove this setting.</p> <p>Default value: none</p>
HelpServerPort	<p>Specifies the TCP port on the specified Business Central Help Server if the deployment uses Help Server. If the deployment uses an online library, remove this setting.</p> <p>Default value: none</p>
OfficeSuiteShellServiceClientTimeout	<p>Defines the time Business Central will wait for the Office Suite Shell Service to respond.</p> <p>Important: This setting has been deprecated in Business Central, and it has no effect on the Web Server instance.</p> <p>Default value: 10</p>
UseAdditionalSearchTerms	<p>Specifies whether Tell me uses the additional search terms that are defined on pages and reports.</p> <p>The additional search terms are specified by the AdditionalSearchTerms and AdditionalSearchTermsML properties.</p> <p>If you set this to <code>false</code> the additional search terms are ignored.</p> <p>Default value: true</p>
DefaultRelativeHelpPath	<p>Specifies the default Help article to open if no other context-sensitive link is specified.</p> <p>Default value: none</p>
PersonalizationEnabled	<p>Specifies whether personalization is enabled in the Business Central Web client. Set to <code>true</code> to enable personalization.</p> <p>For more information, see Managing Personalization.</p>

ApplicationIdSettings **element settings**

SETTING/KEYNAME	DESCRIPTION
-----------------	-------------

SETTING/KEYNAME	DESCRIPTION
BaseHelpUrl	<p>Specifies the link to the online Help library that the deployment uses, such as https://mysite.com/{0}/mylibrary/.</p> <p>Default value: none</p> <p>For more information, see Configuring the Help Experience.</p>
BlogLink	<p>Specifies the target of the blog link on the Help & Support page. Use this link to give users access to your end-user blog.</p> <p>Value: a valid URL Default value: https://go.microsoft.com/fwlink/?linkid=2076643</p>
ComingSoonLink	<p>Specifies the target of coming soon link on the Help & Support page. Use this link to give users access to information about your roadmap or any upcoming features and fixes.</p> <p>Value: A valid URL. Default value: https://go.microsoft.com/fwlink/?linkid=2047422</p>
CommunityLink	<p>Specifies the URL to a community or resource for sharing information.</p> <p>Value: a valid URL Default value: http://go.microsoft.com/fwlink/?LinkId=287089</p>
ContactSalesLink	<p>Specifies the target of the contact sales link on the Help & Support page. Use this link to give users access to your sales-focused web page where users can engage with your sales process. Note This setting and link are not used for Business Central on-premises.</p>
SigninHelpLink	<p>Specifies the URL to open if the user selects help on the sign in page box.</p> <p>Value: a valid URL Default value: none</p>

See Also

[Setting up Multiple Business Central Web Server Instances](#)

[Install Business Central Components](#)

[Business Central Web Server Overview](#)

[Configuring Business Central Server](#)

[Configuring the Help Experience](#)

Setting Up Multiple Business Central Web Server Instances Using PowerShell

3/31/2019 • 5 minutes to read

Although you can use the Business Central Setup to install the Business Central Web Server components and create a single web server instance in IIS for client connection, there may be scenarios when you want to set up multiple instances. For example, you could set up a separate Business Central Web Server instance for the different companies of a business. For this scenario, you can use the Business Central Web Server PowerShell cmdlets, which are outlined in the following table.

CMDLET	DESCRIPTION
New-NAVWebServerInstance	Creates a new Business Central Web Server instance and binds this instance to a Business Central Server instance.
Set-NAVWebServerInstanceConfiguration	Specifies configuration values for a named Business Central Web Server instance.
Get-NAVWebServerInstance	Gets the information about the Business Central Web Server instances that are registered on a computer.
Remove-NAVWebServerInstance	Removes an existing Business Central Web Server instance.

Get started with the Business Central Web Server cmdlets

The Business Central Web Server cmdlets are contained in the PowerShell script module **NAVWebClientManagement.psm1**, which is available on the Business Central installation media (DVD).

The module is installed with the Business Central Server or the Business Central Web Server components.

There are different ways to launch this module and start using the cmdlets:

- If you are working on the computer where the Business Central Server was installed, run the Business Central Administration Shell as an administrator.

For more information, see [Business Central PowerShell Cmdlets](#).

- If you installed the Business Central Web Server components, just start Windows PowerShell as an administrator.
- Otherwise, start Windows PowerShell as an administrator, and use the [Import-Module](#) cmdlet to import the **NAVWebClientManagement.psm1** file:

```
Import-Module -Name [filepath]
```

For example:

```
Import-Module -Name "C:\Program Files\Microsoft Dynamics 365 Business  
Central\130\Service\NAVWebClientManagement.psm1"
```

For more information about starting Windows PowerShell, see [Starting Windows PowerShell](#).

Creating Business Central Web Server instances

Get Access to the WebPublish folder

To create a new web server instance, you need access to the **WebPublish** folder that contains the content files for the Business Central Web Server components.

- This folder is available on the Business Central installation media (DVD) and has the path "DVD\WebClient\Microsoft Dynamics NAV\13x\Web Client\WebPublish".
- If you installed the Business Central Web Server components, this folder has the path "C:\Program Files\Microsoft Dynamics 365 Business Central\140\Web Client\WebPublish".

You can use either of these locations or you can copy the folder to more convenient location on your computer or network.

Decide on the site deployment type for the instance

When you create a new Business Central Web Server instance, you can choose to create either a RootSite or SubSite type. Each instance type has a different hierarchical structure in IIS, which influences its configuration and the URLs for the accessing from the Business Central Web client.

RootSite

A *RootSite* instance is a root-level web site that is complete with content files for serving the Business Central Web client. It is configured with its own set of bindings for accessing the site, such as protocol (http or https) and communication port. The structure in IIS looks like this:

```
- Sites
- BusunessCentralWebSite (web site)
  + nn-NN (language versions)
  + www (content)
  navsettings.json
  ...
```

The Business Central Web client URL for the RootSite instance has the format:

```
http://[WebserverComputerName]:[port]
```

For example: `http://localhost:8080`.

SubSite

A *SubSite* instance is a web application that is under a container web site. The container web site is configured with a set of bindings, but the site itself has no content files. The content files are contained in the application (SubSite). The SubSite inherits the bindings from the container web site. This is the deployment type that is created when you install Business Central Web Server components in the Setup wizard. Using the New-NAVWebServerInstance cmdlet, you can add multiple SubSite instances in the container web site. The structure in IIS for two instances looks like this in IIS:

```
- Sites
- BusinessCentralWebSite (web site)
- BusinessCentralWebInstance1 (application)
  + nn-NN (language versions)
  + www
  navsettings.json
  ...
- BusinessCentralWebInstance2 (application)
  + nn-NN (language versions)
  + www
  navsettings.json
  ...
```

The Business Central Web client URL of a SubSite instance is generally longer than a RootSite because it also contains the application's alias (or virtual path) for the instance, which you define. The URL for a SubSite instance has the format:

```
http://[WebserverComputerName]:[port]/[WebServerInstance]
```

For example: `http://localhost:8080/BusinessCentralWebInstance1` and

```
http://localhost:8080/BusinessCentralWebInstance2 .
```

Run the New-NAVWebServerInstance cmdlet

At the command prompt, run the New-NAVWebServerInstance cmdlet. The following are simple examples for creating a RootSite and SubSite deployment type.

RootSite example:

```
...
New-NAVWebServerInstance -WebServerInstance MyBCWebsite -Server MyBCServer -ServerInstance MyBCServerInstance
-SiteDeploymentType RootSite -WebSitePort 8081 -PublishFolder "C:\Web Client\WebPublish"
...
```

SubSite example:

```
...
New-NAVWebServerInstance -WebServerInstance MyWebApp -Server MyBCServer -ServerInstance MyBCServerInstance -
SiteDeploymentType Subsite -ContainerSiteName MySiteContainer -WebSitePort 8081 -PublishFolder
"C:\WebClient\WebPublish"
...
```

- Substitute *MyBCWebsite* with the name that you want to give the web application in IIS for the web server instance. If you are creating a SubSite deployment type, this name will become part of the URL for opening the Business Central Web client application, for example, `http://MyWebServer:8081/MyWebApp`.
- Substitute *MyBCServer* to the name of the computer that is running the Business Central Server to which you want to connect.
- Substitute *MyBCServerInstance* with the name of the Business Central Server instance to use.
- Substitute *MySiteContainer* with name of the container web site under which you want to add the instance. If you specify a name that does not exist, then a new container web site will be created, which contains the new web server instance.
- Substitute *8081* with the port number that you want to bind the instance to. If you do not specify a port number, then port 80 is used.
- Substitute *C:\WebClient\WebPublish* with the path to your WebPublish folder. By default, the cmdlet looks

in the 'C:\Program Files\Microsoft Dynamics 365 Business Central\140\Web Client' folder. So if you are working on a computer where the Business Central Web Server components are installed, you do not have to set this parameter.

NOTE

This command only sets the required parameters of the NAVWebServerInstance cmdlet. The cmdlet has several other parameters that can use to configure the web server instance. For more information about the syntax and parameters, see [New-NAVWebServerInstance](#).

Modifying a Business Central Web Server instance

After you create the web server instance, if you want to change its configuration, you can use the Set-NAVWebServerInstanceConfiguration cmdlet. Or, you can modify the configuration file (navsettings.json) of the instance directly. For more information, see [Configuring Web Server Instances](#).

See Also

[Business Central Web Server Overview](#)
[Configuring Web Server Instances](#)

Configuring the Business Central Database

3/31/2019 • 7 minutes to read

For a Business Central Server instance to connect to and access a database in SQL Server, the server instance must be authenticated by the database. As in SQL Server, Business Central supports two authentication modes for database communication: Windows Authentication and SQL Server Authentication. When you set up Business Central, you must ensure that database authentication is configured correctly on both the server instance and database, otherwise the server instance will not be able to connect to the database. By default, Windows Authentication is configured on the server instance and database. With Windows Authentication the Business Central Setup does the work for you.

This article describes how to configure SQL Server Authentication. You perform the configuration in two places: on the databases in SQL Server and on the Business Central Server instance. The procedure is different when the Business Central Server instance is configured as a multitenant server instance than when it is not a multitenant server instance.

Set Up an Encryption Key

When using SQL Server authentication, Business Central requires an encryption key to encrypt the credentials (user name and password) that the Business Central Server instance uses to connect to the Business Central database in SQL Server. The encryption key must be installed on the computer where the Business Central Server is installed and also in the database in SQL Server. In a multitenant deployment, the encryption key must be installed in the application database.

To set up an encryption key, you can use one of the following methods:

- You can create and import your own encryption key by using Business Central Administration Shell cmdlets, as described in the following procedure.
- If you are configuring SQL Server authentication on a Business Central Server instance for the first time, you can use the Business Central Server Administration tool which can automatically create and install a system encryption key. If you decide to use this method, no action is required.

Create and import encryption key

1. In the Business Central Administration Shell, run the [New-NAVEncryptionkey cmdlet](#).

This creates a file that contains an encryption key. If you already have an encryption key file, you can skip this step.

2. Run the [Import-NAVEncryptionkey cmdlet](#) to install the encryption key on the Business Central Server instance and database.

On the computer running the Business Central Server instance, the encryption key file has the name BC140.key and is stored in the `%systemroot%\ProgramData\Microsoft\Microsoft Dynamics NAV\[version]\Server\Keys`. In the database, the encryption key is registered in the `dbo.ndopublicencryptionkey` table. In a multitenant deployment, the encryption key is registered in the application database.

Configure SQL Authentication on the Database

This section describes how to configure a Business Central database to use SQL Server Authentication with a Business Central Server instance. You can complete the steps in this procedure by using SQL Server Management Studio or Transact-SQL.

IMPORTANT

In a deployment where the Business Central Server instance is configured as a multitenant server instance, you must complete the following procedure on the application database and tenant database.

Configure SQL Server Authentication on the database in SQL Server

1. Configure the SQL Server instance (Database Engine) that hosts the Business Central database to use SQL Server Authentication.

To use SQL Server authentication, you configure the database instance to mixed authentication mode (SQL Server and Windows Authentication). For more information, see [Change Server Authentication Mode](#).

2. In the SQL Server instance, create a login that uses SQL Server authentication.

For more information, see [Create a Login](#).

3. Map the login to a user in the Business Central database, and add the user to the **db_owner** role of the Business Central database.

For more information, see [Create a Database User](#).

Configure SQL Server Authentication on the Business Central Instance (Non-Multitenant)

You configure the Business Central Server instance with the login credentials (user name and password) of the user account in the Business Central database in SQL Server that you want to use for authentication. You can do this using the Business Central Server Administration tool or Business Central Administration Shell.

Configure SQL Authentication on a server instance using Business Central Server Administration tool

1. Open the Business Central Server Administration tool.
2. In the console tree, which is the left pane, expand the node for the computer that contains the Business Central Server instance, and then select the Business Central Server instance.
3. In the **Actions** pane, choose **Database Credentials**.
4. On the **Database Credentials** page, choose the **Edit** button.
5. Set the **Database Authentication Type** to **SQL Authentication**.
6. In the **Database User Name** field, type the login name for the database user that you want to use to access the Business Central database in SQL Server.
7. In the **Password** field, type the login password for the database user that you want to use to access the Business Central database in SQL Server.
8. Choose the **Save** button, and then on the **Enable Encryption on SQL Server Connections** dialog box, choose the **OK** button.

Encryption keys are used to help secure the login credentials over the connection between the Business Central Server instance and the Business Central database in SQL Server.

9. On the **Information** dialog box about encryption, choose the **OK** button.

This dialog box is to inform you to enable encryption on SQL Server connections, which is disabled by default.

10. If you want to enable encryption on SQL Server connections, in the **Action** pane, choose **Configuration**,

and then choose the **Edit** button. In the **Database** tab, select **Enable Encryption on SQL Connections**, choose the **Save** button, and then the **OK** button.

11. Restart the server instance.

Configure SQL Authentication on a server instance using Business Central Administration Shell

- If you are modifying an existing Business Central Server instance, run the [Set-NAVServerConfiguration cmdlet](#).

Use the *DatabaseCredentials* parameter to provide the login credentials of the database user that you want to use to access the application database.

- If you are creating a new Business Central Server instance, run the [New-NAVServerInstance cmdlet](#).

Use the *DatabaseCredentials* parameter to provide the login credentials of the database user that you want to use to access the application database.

Configure SQL Server Authentication on Business Central Instance in a Multitenant Deployment

This section describes how to configure a Business Central database to use SQL Server Authentication with a Business Central Server instance. You can complete the steps in this procedure by using SQL Server Management Studio or Transact-SQL.

To configure a SQL Server Authentication on a Business Central Server instance, you set up the server instance with the login credentials (user name and password) for the user accounts for the application and tenant databases in SQL Server. You can do this using the Business Central Server Administration tool or Business Central Administration Shell.

Configure SQL Authentication using Business Central Server Administration tool

1. Open the Business Central Server Administration tool.
2. In the console tree, which is the left pane, expand the node for the computer that contains the Business Central Server instance, and then select the Business Central Server instance.
3. Configure SQL Server Authentication with the application database as follows:
 - a. In the **Actions** pane, choose **Database Credentials**.
 - b. On the **Database Credentials** page, choose the **Edit** button.
 - c. Set the **Database Authentication Mode** to **SQL Server Authentication**.
 - d. In the **Database User Name** field, type the login name for the database user that you want to use to access the Business Central application database in SQL Server.
 - e. In the **Password** field, type the login password for the database user that you want to use to access the Business Central database in SQL Server.
 - f. Choose the **Save** button, and then on the **Enable Encryption on SQL Server Connections** dialog box, choose the **OK** button.

Encryption keys are used to help secure the login credentials over the connection between the Business Central Server instance and the Business Central database in SQL Server.

- g. On the **Information** dialog box about encryption, choose the **OK** button.

This dialog box is to inform you to enable encryption on SQL Server connections, which is disabled by default.

- h. If you want to enable encryption on SQL Server connections, in the **Action** pane, choose **Configuration**, and then choose the **Edit** button. In the **Database** tab, select **Enable Encryption on SQL Connections**, choose the **Save** button, and then the **OK** button.
4. To configure SQL Server Authentication with the tenant database, mount the tenant to the Business Central Server instance and specify the login credentials (user name and password) for the database user that you want to use to access the Business Central tenant database in SQL Server.

If the tenant is already mounted to the Business Central Server instance, you must dismount the tenant, and mount it again.

For more information see [Mount or Dismount a Tenant](#).

5. Restart the server instance.

Configure SQL Authentication using Business Central Administration Shell

1. Configure SQL Server Authentication with the application database as follows:

- If you are modifying an existing Business Central Server instance, run the [Set-NAVServerConfiguration cmdlet](#).

Use the *DatabaseCredentials* parameter to provide the login credentials of the database user that you want to use to access the application database.

- If you are creating a new Business Central Server instance, run the [New-NAVServerInstance cmdlet](#).

Use the *DatabaseCredentials* parameter to provide the login credentials of the database user that you want to use to access the application database.

2. To configure SQL Authentication with the tenant database, run the [Mount-NAVTenant cmdlet](#).

Use the *DatabaseCredentials* parameter to provide the login credentials of the database user that you want to use to access the tenant database.

See Also

[Installation Considerations for Microsoft SQL Server](#)

[Deployment](#)

[Installing Business Central Using Setup](#)

Business Central Performance Counters

3/31/2019 • 9 minutes to read

The following table describes the performance counters that are available in Business Central for monitoring Business Central Server instances.

Client session counters

These counters pertain to sessions from the clients, NAS, and web services, to the server instance.

COUNTER	DESCRIPTION
# Active sessions	<p>Number of active sessions on the Business Central Server instance.</p> <p>An active session is a connection to the Business Central Server instance from a Business Central client, such as the Dynamics NAV Client connected to Business Central or Business Central Web client, and Web services (OData and SOAP).</p>
Server operations/sec	<p>Number of operations that have started on the Business Central Server per second.</p> <p>An operation is a call to the Business Central Server instance from a Business Central client to run Business Central objects.</p> <p>Note: OData and SOAP requests are not included.</p>
Average server operation time (ms)	Average duration of server operations in milliseconds.

SQL Server connection counters

These counters pertain to the connection from the server instance to the SQL Server instance and databases.

COUNTER	DESCRIPTION
# Mounted tenants	Number of tenants that are mounted on the Business Central Server instance. This counter is relevant with a multitenant server instance, where tenants are often mounted and dismounted.
# Open connections	<p>The current number of open connections from the Business Central Server instance to Business Central databases on SQL Servers.</p> <p>The value is always equal to the sum of the # Open tenant connections counter and the # Open application connections counter. -We recommend that you use these counters instead.</p>

COUNTER	DESCRIPTION
# Open application connections	<p>Current number of open application connections from the Business Central Server instance to the Business Central application database on SQL Servers.</p> <p>Because all connections are to only one application database, you will see failures when the total number of connections for all server instances exceeds the maximum number of connections allowed to the database.</p>
# Open tenant connections	<p>Current number of open tenant connections from the Business Central Server instance to Business Central tenant databases on SQL Servers.</p> <p>If there are multiple tenant databases, you cannot see the distribution of opened connections per database (or database pool).</p> <p>With Azure SQL Database, connections are denied if the throttling limit is reached. The limit depends on the database configuration. Be aware that in clusters, other server instances will also have connections to the same database, so the total load on a database requires that you look at multiple server instances.</p>
% Query repositioning rate	Percentage of queries that are re-executed when fetching the query result.
Hard throttled connections	Number of connections that were hard-throttled.
Soft throttled connections	Number of connections that were soft-throttled.
Transient errors	Number of transient errors.
Heartbeat time (ms)	<p>The time that it takes to complete a single write to a system table. Conceptually, this counter measures the time it takes to call the application database server to update the 'last active' field the dbo.Service Instance table for the Business Central Server instance. Every 30 seconds, the instance writes a record to indicate that the instance is "alive."</p> <p>You can use this counter to indicate if there is network latency between the Business Central Server and the database.</p>
# Preferred connection total requests	Count of the total number of requests to the preferred connection cache. The preferred connection cache contains requests from the SQL connection pool that was last used by a Business Central user.
% Preferred connection cache hit rate	Percentage of hits in the preferred connection cache, compared to the total number of requests.

Data and caching counters

These counters pertain to the data caching on the server instance.

COUNTER	DESCRIPTION
# Calculated fields cache total requests	Count of the total number of requests to the calculated fields cache. The calculated fields cache contains the results of CALCFIELDS method (Record) calls.
% Calculated fields cache hit rate	Percentage of hits in the calculated fields cache, compared to the total requests to the calculated fields cache.
# Command cache total requests	Count of the total number of requests to the command cache. The command cache contains the results of all SQL commands.
% Command cache hit rate	Percentage of hits in the command cache, compared to the total requests to the command cache.
# Primary key cache total requests	Count of the total number of requests to the primary key cache. The primary key cache contains the results of requests to get a record by using its primary key.
% Primary key cache hit rate	Percentage of hits in the primary key cache, compared to the total requests to the primary key cache.
# Result set cache total requests	Count of the total number of requests to the result set cache. The result set cache contains result sets that are returned from SQL Server.
% Result set cache hit rate	<p>Percentage of hits in the result set cache, compared to the total requests to the result set cache.</p> <p>The value also depends on the usage pattern and which parts of the application are used. For example, the SELECTLATESTVERSION method will clear the cache, which results in a lower hit rate.</p> <p>In general, reading frequently updated values will lower the hit rate because the cache synchronization across Business Central Server instances will remove stale values, which causes re-reads.</p>
# Rows in all temporary tables	Count of number of rows in all temporary tables.

Scheduled task counters

These pertain to tasks that are run by Task Scheduler.

COUNTER	DESCRIPTION
# Available tasks	Remaining number of tasks that can potentially run simultaneously before the maximum number of tasks is reached. The value of this counter is the value the Maximum # of tasks counter minus the value of the # Running tasks counter.

COUNTER	DESCRIPTION
# of task errors/sec	Number of errors per second that are caused by running tasks. The task are causing errors in AL or exceptions on the server instance. If the value is greater than zero for an extended period of time, this typically indicates a failing task that keeps getting rescheduled.
# Running tasks	Number of tasks that are currently running on the server instance. The value is limited to the value of the Maximum # of tasks counter.
Average task execution time	<p>The average time (in ticks) that tasks have taken to complete. Task execution time is counted regardless of whether the task completed successfully or raised an error.</p> <p>There is no general rule for what the normal operations level is. To analyze this counter, look for large spikes to identify long-running tasks.</p> <p>Note: A tick is the smallest unit that the your system uses for time measurements, and it is typically determined by the operating system. For example, in Windows, a single tick represents one hundred nanoseconds, which means that there are 10,000 ticks in a millisecond. Tick durations can differ bewteen systems, so be aware of this fact when comparing absolute values across systems.</p>
Maximum # of tasks	The maximum number of tasks that can run simultaneously. This value is defined by the Maximum Concurrent Running Tasks (TaskSchedulerMaxConcurrentRunningTasks) setting in the server instance configuration. Therefore, this value is constant until the server instance setting is changed and the instance is restarted.
Total # Pending tasks	The total number of tasks in the shared task list that are waiting to be picked up by server instances connected to this application database. The tasks counted are those that are ready and have been scheduled to run now or earlier and that are not currently running.
Total # Running tasks	Total number of tasks in the shared task list that are currently running by any server instance connected to this application database.
Time (ms) since the list of running tasks last had capacity for new tasks	The time (ms) since the list of running tasks last had capacity for new tasks.

For more information about task scheduler, see [Task Scheduler](#).

See Also

[Set up Performance Counters in Windows Performance Monitor](#)

[Create a Data Collector Set From Template](#)

[Optimizing SQL Server Performance with Business Central](#)

Monitoring Business Central Server Events

3/31/2019 • 2 minutes to read

You can monitor events on Business Central Server to diagnose conditions and troubleshoot problems that affect operation and performance.

Event Logging Overview

Business Central uses Event Tracing for Windows (ETW), which is a subsystem of Windows operating systems. ETW provides a tracing mechanism for events that are raised by an application or service. ETW enables you to use industry standard tools such as Windows Performance Monitor, PerfView, Event Viewer, and Windows PowerShell to dynamically collect data on trace events that occur on the Business Central Server.

Events that occur on Business Central Server instances are recorded in Windows Event logs on the Business Central Server computer. Dynamics 365 Business Central uses channels on all events. Event channels provide a way to collect and view events from a specific provider, which in this case is Business Central Server, and group the events according to predefined types, such as admin, operational, and debug. For example, in Event Viewer, Business Central Server instance events are collected in the Admin, Operational, and Debug channel logs for Business Central in the Applications and Services Logs.

For more general information about ETW and event channels, see [Event Tracing for Windows](#) and [Event Logs and Channels in Windows Event Log](#).

Monitoring Business Central Server Event Traces

Event tracing provides detailed information about what is occurring on the Business Central Server and application when users work with Business Central. This can help you identify and analyze problems or conditions that affect performance. Event tracing enables you to dynamically monitor Business Central Server without having to restart the server or Business Central clients. By using industry-standard tools for event tracing, you can start and stop event tracing sessions, and then view the trace event data from a stored log file.

You can use event tracing to track the following operations on Business Central Server instances:

- Running Business Central reports, queries, and XMLports.
- Execution of SQL statements by Business Central Server.
- Execution of AL functions.
- Telemetry.
- Windows event log events.

Event Trace Monitoring Tools

There are various industry-standard tools that you can use to collect event trace data. The procedures in this section use Windows Performance Monitor, PerfView, Event Viewer, and Windows PowerShell to illustrate how you can collect and view event trace data. For details about how to use these tools and others, refer to the documentation available with the tool. For an overview of some of the tools, see [Tools for Monitoring Performance Counters and Events](#).

Get Started

TASK	FOR MORE INFORMATION, SEE
Review the list of trace events that are available for monitoring Business Central Server instances.	Business Central Trace Events List
Collect event trace data in an event trace log (.etl) file. Use the event trace monitoring tool to start an event trace session.	Use Performance Monitor to Collect Event Trace Data Use PerfView to Collect Event Trace Data Use Logman to Collect Event Trace Data
View event trace data that is contained in an .etl file.	Use PerfView to View Event Trace Data
Use Event Viewer to collect and view events	Monitoring Business Central Server Events by Using Event Viewer
Use Windows PowerShell to view event trace data	Monitoring Business Central Server Events by Using Windows PowerShell
Turn off or limit the amount of telemetry trace events emitted based on the severity level.	Turn Off or Limit Telemetry Trace Events

See Also

[Business Central Server Trace Events](#)

[Business Central Server Admin and Operational Events](#)

Business Central Server Trace Events

3/31/2019 • 8 minutes to read

This article provides an overview of the trace events that are generated by Business Central server instance.

Overview

- There are two event trace providers that publish different trace events to the event log: **Microsoft-Dynamics365BusinessCentral-Server** and **Microsoft-Dynamics365BusinessCentral-Common**. The **Microsoft-Dynamics365BusinessCentral-Common** provider is strictly for telemetry trace events. All other events use **Microsoft-Dynamics365BusinessCentral-Server**. You typically need to specify the event trace provider in the monitoring tool that you are using.
- There are several types of trace events for each event trace provider, including: Windows event viewer, SQL traces, service calls, AL function calls, and telemetry. Trace event types are identified by a keyword with a corresponding decimal and hexadecimal value. The keywords and values enable you to collect data on specific trace events. Some tools support the hexadecimal values only and other tools support both hexadecimal and decimal.
- For some trace events, there is separate event for when an operation starts and when it stops. This enables you to determine the duration of the operation. Some monitoring tools, such as PerfView, will automatically return the duration in the stop event.
- Some monitoring tools might interpret and display the collected event trace differently than others. For more information, see [Event Trace Data](#).

Windows Event Viewer Trace Events

Windows Event Viewer trace events track errors, warnings, and information messages that provide information about the condition or state of Business Central Server instances. These events can be viewed in the **Dynamics365BusinessCentral** channel logs and **Application** log of Event Viewer on the computer that is running Business Central Server. For more information, see [Monitoring Business Central Server Events Using Event Viewer](#).

EVENT TRACE PROVIDER	KEYWORD	DECIMAL VALUE	HEXADECIMAL VALUE
Microsoft-Dynamics365BusinessCentral-Server	EventViewer	1	0x1

For a list and description of EventViewer trace events, see [Business Central Server Admin and Operational Events](#).

SQL Trace Events

SQL trace events track a specific set of SQL statements that are executed from the Business Central Server instance against the Business Central database on SQL Server.

EVENT TRACE PROVIDER	KEYWORD	DECIMAL VALUE	HEXADECIMAL VALUE
Microsoft-Dynamics365BusinessCentral-Server	SQLTracing	2	0x2

The event data that is collected includes: session ID, tenant ID, the Business Central user, and the SQL statement. For more information, see [Event Trace Data](#).

The following table lists the SQL trace events.

ID	EVENT (TASK/OPCODE)	WHAT IS TRACED
1	ExecuteScalar/Start	SQL statements that query a database table and return a single field from a row in the query result.
2	ExecuteScalar/Stop	SQL statements that query a database table and return a single field from a row in the query result.
3	ExecuteNonQuery/Start	SQL statements that return a number of rows from a database table
4	ExecuteNonQuery/Stop	SQL statements that return a number of rows from a database table
5	ExecuteReader/Start	SQL statements that return a set of rows from a database table.
6	ExecuteReader/Stop	SQL statements that return a set of rows from a database table.
7	ReadNextResult/Start	SQL statements that return the next result from a database query.
8	ReadNextResult/Stop	SQL statements that return the next result from a database query.
9	ReadNextRow/Start	SQL statements that return the next row in database table.
10	ReadNextRow/Stop	SQL statements that return the next row in database table.
11	BeginTransaction/Start	SQL statements that start a database transaction.
12	BeginTransaction/Stop	SQL statements that start a database transaction.
13	Prepare/Start	SQL statements that create a prepared version of the command on an instance of SQL Server.

ID	EVENT (TASK/OPCODE)	WHAT IS TRACED
14	Prepare/Stop	SQL statements that create a prepared version of the command on an instance of SQL Server.
15	OpenConnection/Start	SQL statements that open connection to the database from the connection pool.
16	OpenConnection/Stop	SQL statements that open connection to the database from the connection pool.
17	Commit/Start	SQL statements that commit a database transaction.
18	Commit/Stop	SQL statements that commit a database transaction.
19	Rollback/Start	SQL statements that cancel the changes in a pending database transaction.
20	Rollback/Stop	SQL statements that cancel the changes in a pending database transaction.

Service Call Trace Events

Service call trace events track when Business Central clients run the Business Central objects: Queries, Reports, and XMLports.

EVENT TRACE PROVIDER	KEYWORD	DECIMAL VALUE	HEXADECIMAL VALUE
Microsoft-Dynamics365BusinessCentral-Server	ServiceCall	4	0x4

The event data that is collected includes: session ID, tenant ID, Business Central user, and the Business Central object ID. For more information, see [Event Trace Data](#).

The following table lists the service call trace events.

ID	EVENT (TASK/OPCODE)	WHAT IS TRACED
300	RunQuery/Start	Business Central Query objects that are opened and closed.
301	RunQuery/Stop	Business Central Query objects that are opened and closed.
302	RunReport/Start	Business Central Report objects that are opened and closed.

ID	EVENT (TASK/OPCODE)	WHAT IS TRACED
303	RunReport/Stop	Business Central Report objects that are opened and closed.
310	RunXmlPort/Start	Business Central XMLport objects that are opened and closed.
311	RunXmlPort/Stop	Business Central XMLport objects that are opened and closed.
500	OpenSession	Dynamics NAV Client connected to Business Centrals and Business Central Web clients establish a connection to the Business Central Server instance.
501	CloseSession	Dynamics NAV Client connected to Business Centrals and Business Central Web clients establish a connection to the Business Central Server instance.

AL Methods Trace Events

AL function tracing events track the execution of AL functions and statements on the Business Central Server instance.

EVENT TRACE PROVIDER	KEYWORD	DECIMAL VALUE	HEXADECIMAL VALUE
Microsoft-Dynamics365BusinessCentral-Server	ALTracing	8	0x8

The event data that is collected includes: session ID, tenant ID, Business Central user, ALAL function, AL statements, and line number. For more information, see [Event Trace Data](#).

IMPORTANT

If the Business Central Server instance is not configured for full AL function tracing, then only root-level AL function will be traced. Statements and AL functions that are called from functions will not be traced. By default, the Business Central Server instance is not configured for full AL function tracing. For information about how to specify full AL function tracing, see [Configuring Business Central Server](#).

The following table lists the AL function tracing events.

ID	EVENT (TASK/OPCODE)	WHAT IS TRACED				
400	ExecuteALFunction/Start	AL functions that are called.		401	ExecuteALFunction/Stop	AL functions that are called.

ID	EVENT (TASK/OPCODE)	WHAT IS TRACED				
402	ExecuteALFunctionFailed	Errors that occur when executing AL functions. The errors can be caused by exceptions or ERROR Function (Dialog) calls.				
403	ExecuteALFunction	AL statements that are executed. Important: This trace event is only traced when the Business Central Server is configured to full AL function tracing.				

Telemetry Trace Events

Telemetry trace events can provide data about operations in the application and how it is being used in production. These events include both system telemetry trace events and user-defined, custom telemetry trace events.

EVENT TRACE PROVIDER	KEYWORD	DECIMAL VALUE	HEXADECIMAL VALUE
Microsoft-Dynamics365BusinessCentral-Common	TelemetryTracing	32	0x20

Custom telemetry trace events are emitted from the application. These are events that are sent by [SENDTRACETAG method](#) calls from inside the application. For more information about custom telemetry trace events, see [Instrumenting an Application for Telemetry](#).

Some of the important event data that is collected for both system and custom telemetry trace events includes: tag, category, message, dataclassification. For more information about this data, see [Event Trace Data](#).

Telemetry events can have one of the following event IDs, based on the data classification and verbosity (or severity level):

DATA CLASSIFICATION	VERBOSITY	ID
All except CustomerContent and EndUserIdentifiableInformation	Critical	700
	Error	701

DATA CLASSIFICATION	VERBOSITY	ID
	Informational	702
	Informational	703
	Verbose	704
	Warning	705
	Informational	706
CustomerContent or EndUserIdentifiableInformation	Critical	707
	Error	708
	Informational	709
	Informational	710
	Verbose	711
	Warning	712

NOTE

Event IDs 703, 706, and 710 are used only for system telemetry trace events. All other IDs are used for both system and custom events.

Event Trace Data

The following table lists the arguments that make up the data collected for trace events. When viewing event trace data, the way that the arguments are interpreted and displayed can vary depending on the tool that you use.

ARGUMENT	DESCRIPTION	TRACE EVENT TYPE				
category	Specifies the category of the telemetry trace event.	Telemetry (TelemetryData)				

ARGUMENT	DESCRIPTION	TRACE EVENT TYPE				
connectionType	Specifies the RoleTailored client that has established the connection to the Business Central server instance. Values include Dynamics NAV Client connected to Business Central and Business Central Web client.	Service calls (ServiceCall)				
dataclassification	Specifies the RoleTailored client that has established the connection to the Business Central server instance. Values include Dynamics NAV Client connected to Business Central and Business Central Web client.	Service calls (ServiceCall)				
failureMessage	Includes the error message that is returned when a AL function fails.	AL function trace events (ALTracing)				
functionName	Specifies the AL function that was executed.	AL function trace events (ALTracing)				

ARGUMENT	DESCRIPTION	TRACE EVENT TYPE				
lineNumber	Specifies the line number of the statement in the AL code of the Business Central object that was executed.	AL function trace events				
message	Specifies the error, warning, or information message text that was issued for a trace event	Windows event log trace events (EventViewer) Telemetry (TelemetryData)				
objectId	Specifies the ID of the Business Central object that was executed in the session.	Service calls trace events (ServiceCall) AL function trace events (ALTracing)				
objectType	Specifies the Business Central object type that executed by a AL function or statement. Values include the following: CodeUnit, Page, Query, Report, Table, and XMLport.	AL function trace events (ALTracing)				

ARGUMENT	DESCRIPTION	TRACE EVENT TYPE				
sessionId	Specifies the ID that is assigned to the session that is used by the operation. Each operation establishes a session with the Business Central Server instance from a connection in the Business Central Server's connection pool.	All				
sqlStatement	Specifies the SQL statement that was executed on the session.	SQL trace events (SQLTracing)				
statement	Specifies the AL statement that was executed on the session.	AL function trace events		tag	Specifies the ID of the telemetry trace event. For system telemetry trace events, this ID is autogenerated. For custom telemetry trace events, it is user-defined.	Telemetry (TelemetryData)

ARGUMENT	DESCRIPTION	TRACE EVENT TYPE				
tenantId	Specifies the ID of the tenant database that is mounted on the Business Central Server instance. If the Business Central Server instance is not configured for multitenancy, then the value is empty. For more information about multitenancy, see Multitenant Deployment Architecture .	All				
userName	Specifies the Business Central user account that is logged on to the session.	All				

See Also

[Monitoring Business Central Server Events](#)

[Classifying Data](#)

Business Central Server Admin and Operational Events (EventViewer) List

3/31/2019 • 11 minutes to read

Events have the source `BusinessCentralServer$[ServerInstance]`. Each event has a unique ID and is assigned to a task category. The source, IDs, and task categories enable you to filter the events that display in Event Viewer. For a description of the task categories, see [Task Categories](#).

The following table lists the events that are generated by the Business Central Server.

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	MESSAGE/REMARKS						
112	Error	12	Operational	<p>Message: Fatal Sql error. The connection can no longer be used.</p> <p>Remarks: An error occurred on the connection from the Business Central Server instance to the SQL database and the connection could not be established.</p> <p>This</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	error MESSAGE/REMARKS						
				<p>error MESSAGE/REMARKS caused by one of the following reasons:</p> <ul style="list-style-type: none">- The Business Central Server has been stopped.- The SQL server connection settings are incorrect- A network failure has occurred.- A hardware failure has occurred on the server or on your computer. <p>To resolve this issue, try to restart the Business Central Server or see Troubleshooting: A failed</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	MESSAGE/REMARKS						
				total Message occurred. The connection to SQL server cannot be established.						
113	Information	12	Operational	<p>Message: Object Change Listener is listening on SQL Server '{0}' in Database '{1}'.</p> <p>Remarks: Occurs when the Business Central Server instance has established a connection to the SQL database. The Change Listener object listens for changes to application objects in the Business Central database.</p>		200	Error	12	Admin	<p>Remarks: This event ID is used for various errors that occur on Business Central Server instances.</p> <p>These events indicate that the Business Central Server instance is not operating. View the details of each message to determine the cause of the problem.</p>

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	MESSAGE/REMARKS						
201	Information	12	Operational	<p>Remarks:</p> <p>This event ID is used for various information messages that occur on Business Central Server instances.</p> <p>These events are typical conditions and are for information only.</p>						
202	Warning	12	Admin	<p>Remarks:</p> <p>This event ID is used for various warning messages that occur on Business Central Server instances.</p> <p>These events indicate that an unexpe</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	<p> cted MESSA GE/RE MARKS occurre </p>						
				<p> d on the Busines s Central Server instanc e. In most cases, the Busines s Central Server instanc e will still be operati onal. </p> <p> View the details of each messag e to determ ine the cause of the proble m. </p>						
205	Inform ation	8	Operati onal	<p> Messa ge: 'Micros oft Dynam ics NAV Data Service' is listenin g to request s at net.tcp: //[Serv er]: [Port]/[ServerI nstanc e]/ODa ta </p> <p> Remark s: Indicat es that </p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	the MESSAGE/REPORT for the listening ports						
				<p>OData web services has been opened on the Business Central Server instance and it is ready to handle OData requests.</p> <p>Typically, this condition occurs when the Business Central Server instance starts.</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	MESSAGE/REMARKS						
206	Information	11	Operational	<p>Message: 'Microsoft Dynamics NAV Data Service' at net.tcp://[Server]:[Port]/[ServerInstance]/OData has stopped.</p> <p>Remarks: Indicates that the listening port for OData web services on the Business Central Server instance has been closed.</p> <p>Typically, this condition occurs when the Business Central Server instance is stopped.</p>						
207	Information	8	Operational	Message						

				starts.						
EVENT ID	Inform Level ation	TASK CATEGORY	CHANNEL Operational	MESSAGE/REMARKS						
				<p>'Microsoft Dynamics NAV Business Web Services' at net.tcp://[Server]:[Port]/[ServerInstance]/WS/Services has stopped.</p> <p>Remarks: Indicates that the listening port for SOAP web services has been closed.</p> <p>Typically, this condition occurs when the Business Central Server instance is stopped.</p>						
209	Information	8	Operational	<p>Message: 'Microsoft Dynamics NAV Service' is</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	listenin MESSAGE TO REMARKS						
				net.tcp: //([Server]: [Port])/[ServerI nstanc e]/Servi ce Remark s: Indicat es that the listenin g port for Dynam ics NAV Client connec ted to Busines s Central has opened and it is ready for Dynam ics NAV Client connec ted to Busines s Central connec tions. Typicall y, this conditi on occurs when the Busines s Central Server instanc e starts.						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	MESSAGE/REMARKS						
210	Information	11	Operational	<p>Message: 'Microsoft Dynamics NAV Service' at net.tcp://[Server]:[Port]/[ServerInstance]/Service has stopped.</p> <p>Remarks: The listening port for Dynamics NAV Client connected to Business Central has closed.</p> <p>Typically, this condition occurs when the Business Central Server instance is stopped.</p>		211	Information	12	Operational	<p>Remarks: This event ID is used for various information messages that are generated by Microsoft Dynamics NAV Application Server (NAS) and background sessions.</p> <p>Because NAS does not have a user interface, events are used to provide operational information to administrators or developers.</p>
214	Information	8	Operational	<p>Message: 'Microsoft Dynamics</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	NAV MESSAGE/REMARKS						
				<p>listening to requests at net.tcp://[Server]:[Port]/[ServerInstance]/ManagementService.</p> <p>Remarks:</p> <p>Indicates that the listening port for Business Central Server Administration tool has been opened on the Business Central Server instance.</p> <p>Typically, this condition occurs when the Business Central Server instance starts.</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	MESSAGE/REMARKS						
215	Information	11	Operational	<p>Message: 'Microsoft Dynamics NAV Service' at net.tcp://[Server]:[Port]/[ServerInstance]/ManagementService has stopped.</p> <p>Remarks: Indicates that the listening port for the Business Central Server Administration tool has closed.</p> <p>Typically, this condition occurs when the Business Central Server instance is stopped.</p>						
216	Error	13	Admin	<p>Remarks: This event</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	ID is used for various errors that occur when the Business Central Server cannot start or establish a connection to the Business Central database on SQL Server.						
				These events are caused by unhandled exceptions that are thrown the Business Central Server instances and indicate that an unrecoverable condition has occurred.						
				The errors can be caused by an incorrect configu						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	ration MESSAGE/REMARKS						
				Central Server or the Microsoft SQL Server connection. To resolve errors, verify the Business Central Server and SQL Server configuration. For more information, see Configuring Business Central Server and Troubleshooting: SQL Server Connection Problems .						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	MESSAGE/REMARKS						
217	Information	13	Operational	<p>Remarks:</p> <p>This event ID is used for various information messages that occur when the Business Central Server starts and establishes a connection to the Business Central database on SQL Server.</p> <p>These events are caused by exceptions that are thrown by the Business Central Server instances. These events are typical conditions and are for information</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	only MESSAGE/REMARKS						
218	Warnin g	13	Admin	Remark						
				<p>s:</p> <p>This event ID is used for various warnings that occur when the Business Central Server starts and establishes a connection to the Business Central database on SQL Server.</p> <p>These events are caused by handled exceptions that are thrown by the Business Central Server instances.</p> <p>Typically, the Business Central Server will continue to operate, but</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	MESSAGE PRESENTATION ADDRESS						
				As the problem that is described in the event message.						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	MESSAGE/REMARKS						
219	Information	12	Operational	<p>Message: Microsoft Dynamics NAV Application Server for tenant '[TenantID]' is scheduled to start with the following configuration: Company: [CompanyName], Codeunit: [StartUpCodeUnitID], Method: [StartUpMethod], Arguments: [StartUpArguments]</p> <p>Remarks: Refers to NAS service only. Indicates that the NAS service is scheduled to start.</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	MESSAGE/REMARKS						
220	Information	12	Operational	<p>Message: Microsoft Dynamics NAV Application Server for tenant '[TenantID]' has completed.</p> <p>Remarks: Refers to NAS service only. Indicates that the NAS service has started successfully.</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	MESSAGE/REMARKS						
221	Error	12	Admin	<p>Message: The Microsoft Dynamics NAV Application Server session for tenant '[TenantID]' has failed and will be restarted. Reason : [Message]</p> <p>Remarks: Refers to NAS service only. Indicates that an exception has occurred and NAS service did not start. NAS service will attempt to start again.</p>						
222	Error	12	Admin	<p>Message: The Microsoft Dynamics NAV</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	Application Message/Remarks						
				for tenant '[TenantID]' has permanently failed and will not be restarted. Reason : [Message]						
				Remarks: Refers to NAS service only. Indicates that an exception has occurred and NAS service did not start. The NAS service will not be restarted because the maximum number of times that service can attempt to restart has been met. This value is specified by						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	configuration MESSAGE/REMARKS						
				are specified in the CustomSettings.xml file. For more information about the CustomSettings.xml file, see Configuring Business Central Server .						
224	Information	12	Operational	<p>Message: The service has completed configuration and is ready.</p> <p>Remarks: Occurs when the Business Central Server is started and is ready for use.</p>		225	Information	12	Operational	<p>Message: The service is shutting down.</p> <p>Remarks: Occurs when Business Central Server is stopped.</p>
226	Information	12	Operational	Message: The NAV application was mounted from						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	MESSAGE/REMARKS						
				<p>databaseName'] on database server '[SQLServerInstance]'. Remarks: Refers to Business Central Server instances that are used in a multi-tenant environment. Indicates that the Business Central Server instance is connected to the Business Central application in the specified application database. Typically, this condition occurs when the Business Central</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	Server Message or the Remarks						
				Mount - NAVApplication cmdlet is run.						
227	Error	12	Admin	<p>Message: The NAV application could not be mounted for database '[DatabaseName]' on database server '[SQLServerInstance]' due to the following error: [Message].</p> <p>Remarks: Refers to Business Central Server instances that are configured for multitenancy.</p> <p>Indicates that the Business Central Server instance</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	MESSAGE/REMARKS						
				<p>Business Central application in the specified application database.</p> <p>Verify that Business Central Server is configured to use the correct application database. For more information, see Migrating to Multitenancy.</p>						
228	Information	12	Operational	<p>Message: Tenant '[Tenant]' was mounted from database '[DatabaseName]' on database server '[SQLServerInstance]'.</p> <p>Remarks: Refers to the</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	Business MESSAGE/REMARKS						
				Business Central Server instances that are configured for multitenancy. Indicates that the Business Central Server is connected to the tenant that is in the specified tenant database. Typically, this condition occurs when the Business Central Server instance starts or when the Mount - NAVTenantcmdlet is run.						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	MESSAGE/REMARKS						
229	Error	12	Admin	<p>Message: Tenant '[TenantID]' could not be mounted due to the following error: [Message]</p> <p>Remarks: Refers to Business Central Server instances that are configured for multitenancy.</p> <p>Occurs when the Business Central Server instance cannot connect to the tenant database.</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	MESSAGE/REMARKS						
230	Information	12	Operational	<p>Message: Tenant '[Tenant]' was dismounted.</p> <p>Remarks: Refers only to Business Central Server instances that are used in a multitenant environment.</p> <p>Typically, this condition occurs when the [Dismount-NAV] Tenant cmdlet is run.</p>						
231	Error	12	Admin	<p>Remarks: This event ID is used for various errors that occur when authenticating a Business Central user who is</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	MESSAGE/REMARKS						
				trying to log in to the Business Central Server from a RoleTailored client. This event is caused by an error in the authentication system. This event is only relevant when the Business Central Server instance is configured for NavUserPassword or Access Control Service credential types.						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	MESSAGE/REMARKS						
232	Information	12	Operational	<p>Message: A user successfully authenticated against the server.</p> <p>Remarks: This event occurs when a user successfully logs on to the Business Central Server instance from a RoleTailored client.</p> <p>This event is only relevant when the Business Central Server instance is configured for NavUserPassword or Access Control Service credential types.</p>						
232	Information	12	Operational	<p>Message:</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	MESSAGE/REMARKS						
				<p>A user provides an invalid credential. Authentication was not successful.</p> <p>Remarks: This event occurs when a user provides an invalid user name or password when the user logs on to the Business Central Server instance from a RoleTailored client.</p> <p>This event is only relevant when the Business Central Server instance is configured for NavUserPassword or Access Control Service.</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	credential MESSAGE/REMARKS						
233	Information	12	Admin	<p>Messag e: The session attempted to write to table '[Table Name]'; but the write operation was rejected because it exceeds the optional table limit of the license. The license only permits writing to [Number] optional tables per session . The session has already written to the following tables: '[Table Name]'; '[Table Name]'; 'and [Table Name]'.</p> <p>Remarks: This event pertain</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	MESSAGE/REMARKS						
				<p>s to the limited user license that is used on the Business Central solution. A limited user license specifies how many optional tables a session can write to. For more information about licensing for Business Central, see Microsoft Dynamics ERP Licensing Guide.</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	MESSAGE/REMARKS						
700-706	Critical, Error, Warning, Information	33	Admin	Telemetry events. You can configure the lowest level of telemetry events to be recorded in the event log by changing the Diagnostic Trace Level setting in the Business Central Server instance configuration. For more information, see Configuring Business Central Server .						
1000	Error	12	Operational	Certificate monitoring has permanently failed and will not be restarted. Reason :						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	[Message] MESSAGE/REMARKS						
				<p>s: An unhandled exception occurred that prevents the certificate from being monitored.</p> <p>Note: Event IDs 1000 through 1006 refer to the security certificate that is used by the Business Central Server instance to protect communications with client or web services. For more information, see Using Security Certificates.</p>						
1001	Information	12	Operational	<p>Message: Configuration setting 'ClientServicesCertificates'</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	ateThumbprint has been updated. It will not take effect until the service is restarted.						
				Remark: Occurs when the security certificate that is used by Business Central Server has been replaced. The thumbprint is set by the ClientServicesCertificateThumbprint parameter in the CustomSetting.xml file for the Business Central Server.						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	MESSAGE/REMARKS						
1002	Error	12	Operational	<p>Message: The service certificate is valid from [Date] to [Date] only.</p> <p>Remarks: Occurs when the security certificate that is used by the Business Central Server is not valid for use on the current date.</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	MESSAGE/REMARKS						
1003	Warning	12	Operational	<p>Message: The service certificate is close to its expiration date.</p> <p>Remarks: Occurs for the first time 30 days before the expiration date of the security certificate that is used on the Business Central Server, and then one time each day until the certificate is replaced or renewed.</p>						

EVENT ID	LEVEL	TASK CATEGORY	CHANNEL	MESSAGE/REMARKS						
1006	Information	12	Operational	<p>Message: Configuration setting 'ClientServicesCertificateThumbprint' has been updated. It will not take effect until the service is restarted.</p> <p>Remarks: Occurs when a new security certificate is applied on Business Central Server. The thumbprint is set by the <i>ClientServicesCertificateThumbprint</i> parameter in the CustomSetting.xml file for the Business</p>						

				Central MESSA Server GE/REM ARKS						
EVENT ID	LEVEL	TASK CATEGO RY	CHANN EL							

Task Categories

Task categories logically classify events according to the operations that they perform. In Event Viewer, you can sort, include, or exclude events in the Windows Application log based on the task categories. A task category is defined by a decimal number. The following table lists the task categories that are associated with Business Central Server events.

TASK CATEGORY	DESCRIPTION
8	Service connects to the Business Central Server instance.
11	Service disconnects from the Business Central Server instance.
12	Information, warning, or error message from the Business Central Server instance.
13	Exception thrown by Business Central Server.

See Also

[Monitoring Business Central Server Events Using Event Viewer](#)

[Monitoring Business Central Server Events](#)

Monitoring Business Central Server Events Using Event Viewer

3/31/2019 • 5 minutes to read

Events that occur on the Business Central Server instances can be recorded in event logs on the computer that is running Business Central Server. You can view the events by using Event Viewer.

About Business Central Server Events in Event Viewer

Events that occur on Business Central Server instances are recorded in the event channels specific to Business Central and also in the general Windows Application log. Event channels provide a way to collect and view events from a specific event trace provider. This differs from the Windows Application log which contains system-wide events from multiple publishers (applications and components).

Business Central channel logs

In the Event Viewer console tree, open **Applications and Services Logs > Microsoft > Dynamics365BusinessCentral**.

Server folder

The **Server** folder contains events from the event trace provider called **Microsoft-Dynamics365BusinessCentral-Server**. The events are recorded in the following logs:

LOG	DESCRIPTION
Admin	<p>Includes events that target end users and IT administrators. These events typically indicate a problem that requires action to resolve the problem. An example of an admin event is a tenant database failing to mount on the Business Central Server instance.</p> <p>For a list and description of these events, see Business Central Server Admin and Operational Events.</p>
Operational	<p>Includes events that provide information about an operation that occurred on Business Central Server instances. These events are typically ordinary operating events that do not require any action but can be used to analyze and diagnose a problem. An example of an operational event is the shutting down of the Business Central Server instance.</p> <p>For a list and description of these events, see Business Central Server Admin and Operational Events.</p>
Debug	<p>Includes the trace event types: SQL (SQLTracing), service calls (ServiceCalls), and AL function calls (ALTracing). For more information about the different trace events and others ways to monitor them, see Business Central Server Trace Events and Monitoring Business Central Server Events.</p> <p>Note: In Event Viewer, this log is hidden and disabled by default. For information about how to show and enable this log, see Enable Business Central Debug Logs in Event Viewer.</p>

Common folder

The **Common** folder contains telemetry events from the event trace provider called **Microsoft-Dynamics365BusinessCentral-Common**. This folder contains strictly telemetry events, which have IDs 700-707. The telemetry events are recorded in the following logs:

LOG	DESCRIPTION
Admin	<p>Includes custom telemetry trace events that are emitted from the application. These are events that are sent by SENDTRACETAG method calls from inside the application.</p> <p>For more information, see Instrumenting an Application for Telemetry.</p> <p>Note The Dynamics NAV Server instance includes a configuration setting called Diagnostic Trace Level (<code>TraceLevel1</code> in the customsettings.config file) that enables you to specify the lowest severity level of telemetry events to be recorded in the event log, or even turn off telemetry event logging altogether. If you do not see the expected events, then verify the Dynamics NAV Server instance configuration with an administrator. For information, see Configuring Business Central Server.</p>
Operational	Not applicable.
Debug	<p>Includes system telemetry trace events that occur.</p> <p>Note: In Event Viewer, this log is hidden and disabled by default. For information about how to show and enable this log, see Enable Business Central Debug Logs in Event Viewer.</p>

Application log

The Application log includes admin and operational type events (errors, warnings, and information messages) that occur on the Business Central Server instance.

To view the **Application** log, in the console tree, choose **Windows Logs, Applications**.

The events in this log are the same events that are recorded in the **Admin** and **Operation** logs in the **Dynamics365BusinessCentral > Server** channel. Therefore, you can consider the **Application** log to be a secondary log for these events. Unless you are using System Center Operations Manager to monitor Business Central Server events, you can disable logging Business Central Server events to the Windows Application log and rely on **Applications and Services Logs** instead. For more information, see [Disable Logging Events to the Windows Application Log](#).

NOTE

Trace events are not included in this log.

Filtering Dynamics Server Events in Event Viewer

By default, the Business Central Server logs contain events of all levels (error, warning, and information) for all Business Central Server instances. You can use the filtering functionality that is available in Event Viewer to display only Business Central Server instance events that meet specific criteria. For example, if you have several Business Central Server instances, you can filter logs to show only events from a specific Business Central Server

instance. For more information, see the following example.

Example

Your Business Central Server is running several instances that are configured with multiple tenants. In Event Viewer, you want to view only errors that occurred in the last 24 hours on the tenant *MyTenant1* of the Business Central Server instance *MyNavServerInstance1*.

To filter the event log

1. For example, in the console tree of Event Viewer, choose **Applications and Services Logs > Microsoft > Dynamics365BusinessCentral > Server**.
2. Select the **Admin** log.
3. In the **Action** pane, choose **Filter Current Log**.

The **Filter Current Log** window opens.

4. On the **Filter** tab, set the **Logged** drop-down list to **Last 24 hours**.
5. In the **Error Level** section, select the **Error** check box.
6. Choose the **XML** tab.

XML similar to the following is displayed:

```
<QueryList>
  <Query Id="0" Path="Microsoft-Dynamics365BusinessCentral-Server/Admin">
    <Select Path="Microsoft-Dynamics365BusinessCentral-Server/Admin">
      *[System[(Level=2) and TimeCreated[timediff(@SystemTime) <= 604800000]]]
    </Select>
  </Query>
</QueryList>
```

`Microsoft-Dynamics365BusinessCentral-Server` indicates that Business Central Server is the provider of the events in the log.

7. Select the **Edit** query manually check box, and then choose the **Yes** button.
8. In the `<Select Path="Microsoft-Dynamics365BusinessCentral-Server/Admin">` element, after `*[System[(Level=2) and TimeCreated[timediff(@SystemTime) <= 86400000]]]`, add the following lines:

```
and
*[EventData[Data[@Name='tenantId'] and Data = 'MyTenant1']]
and
*[EventData[Data[@Name='serverInstanceName'] and Data='MyNavServerInstance1']]
```

The complete XML should look similar to the following XML:

```
<QueryList>
  <Query Id="0" Path="Microsoft-Dynamics365BusinessCentral-Server/Admin">
    <Select Path="Microsoft-Dynamics365BusinessCentral-Server/Admin">
      *[System[(Level=2) and TimeCreated[timediff(@SystemTime) <= 604800000]]]
      and
      *[EventData[Data[@Name='tenantId'] and Data = 'MyTenant1']]
      and
      *[EventData[Data[@Name='serverInstanceName'] and Data='MyNavServerInstance1']]
    </Select>
  </Query>
</QueryList>
```

9. Choose the **OK** button.

The **Admin** log displays only errors that occurred in the last 24 hours on tenant *Tenant1* and Business Central Server instance *MyNavServerInstance1*. The applied filter can be removed. Alternatively, you can save it as a custom view. For more information about filtering in Event Viewer, see [Filter Displayed Events](#) and [Advanced XML filtering in the Windows Event Viewer](#).

See Also

[Monitoring Business Central Server Events](#)

[Business Central Server Trace Events](#)

[Monitoring Business Central Server](#)

[Monitoring Business Central Server Using Performance Counters](#)

[Windows Event Viewer](#)

How to: Use Performance Monitor to Collect Event Trace Data

3/31/2019 • 3 minutes to read

This topic describes how to use Windows Performance Monitor to collect event trace data for Business Central Server. To collect trace event data, you create a Data Collector Set, and then start the Data Collector Set.

Create a Data Collector Set for collecting Business Central trace event data

1. Start Windows Performance Monitor.
 - Choose **Start**, in the **Search** box, type **perfmon**, and then choose the related link.
2. In the navigation tree, expand **Data Collector Sets**, right-click **User-defined**, choose **New**, and then choose **Data Collector Set**.
3. On the **Create new Data Collector Set Wizard** page, enter a name for the new data collector set, select **Create manually (Advanced)**, and then choose the **Next** button.
4. On the **What type of data do you want to include** page, select the **Event trace data** check box, and then choose the **Next** button.
5. On the **Which event trace providers would you like to enable** page, choose the **Add** button to add a provider.
6. In the **Event Trace Providers** list, select **Microsoft-Dynamics365BusinessCentral-Server**, and then choose the **OK** button.
7. If you want to collect data for all trace events, choose the **Next** button. If you want to collect data on specific trace events, do the following:
 - a. In the **Properties** list, select **Keywords (Any)**, and then choose the **Edit** button.
 - b. On the **Property** page, in the **Manual** box, type the keyword decimal value for the trace event. For a list of keyword values for trace events, see [Business Central Server Trace Events](#).

For example, if you want to collect data on service call trace events, then type **4**. If you want to collect data on more than one trace event, add the keyword values for each trace event and then use the sum in the **Manual** box. For example, if you want to collect data on service calls (keyword decimal value = 4) and AL functions (keyword decimal value = 8), then use the value **12**.

NOTE

Performance Monitor will automatically convert the value to hexadecimal, such as 0x4 or 0xC. You can also enter the keyword hexadecimal values directly.

- c. Choose the **OK** button, and then **Next** button.
8. On the **Where would you like the data to be stored** page, set the **Root directory** box to the folder where you want to save the event trace log file that is generated when you run the Data Collector Set.
 9. Choose the **Finish** button to complete the wizard

The new Data Collector Set appears under **User Defined** in the navigation pane.

Complete the next procedure to increase trace buffer settings to make sure that events are not dropped when collecting trace event data.

Change the Data Collector Set trace buffers

1. In the navigation pane, select the new Data Collector Set.
2. In the main pane, right-click the **DataCollector01** item, and then choose **Properties**.
3. In the **Properties** dialog box, choose the **Trace Buffers** tab.
4. Set the following properties.

PROPERTY	RECOMMENDED MINIMUM VALUES
Buffer size	128
Minimum buffers	50
Maximum buffers	50

You might have to adjust these properties based on the monitoring sessions and expected number of events that will be collected. If a large number of events are collected, then the trace buffer size and count might have to be increased.

5. Choose the **OK** button to save and close the **Properties** dialog box.

To start and stop a Data Collector Set

- To start to collect data, right-click the Data Collector Set, and then choose **Start**.
- To stop collecting data, right-click the Data Collector Set, and then choose **Stop**.

For information about how to schedule a time to start and stop collecting data, see [Schedule Data Collection in Windows Performance Monitor](#).

The collected event trace data is stored in an event trace log (.etl) file in the location that you specified. You can view the data in the log file by using various industry-standard tools, such as PerfView. For information about how to use PerfView to view the event trace data, see [Use PerfView to View Event Trace Data](#).

See Also

[Monitoring Business Central Server Events](#)
[Use PerfView to View Event Trace Data](#)
[Instrumenting an Application for Telemetry](#)

How to: Use PerfView to Collect Event Trace Data

3/31/2019 • 2 minutes to read

This topic describes how to use PerfView to collect event trace data for Business Central Server. When you collect event trace data, the data is stored in an event trace log (.etl) file in a location that you choose.

To install PerfView

- Go to <http://go.microsoft.com/fwlink/?LinkID=313428>, and then follow the instructions to download and install PerfView.

To collect event trace data

1. Open PerfView.exe.
2. On the **Collect** menu, choose **Collect**.

The **Collecting data over a user specified interval** dialog box appears.

3. Set the **Data file** field to the path and name of the log file in which to store the trace event data. The file name must have the .etl file name extension.
4. Choose **Advanced options**.

The upper part of the **Advanced options** area includes check boxes and fields that specify the providers from which to collect event trace data.

5. In the **Additional providers** field, type **Microsoft-Dynamics365BusinessCentral-Server**.
 - If you want to filter on a specific trace event, include a colon after **Microsoft-Dynamics365BusinessCentral-Server**, followed by the hexadecimal keyword value for the trace event. For example, to collect trace events data on service call trace events only, then type **Microsoft-Dynamics365BusinessCentral-Server:0x4**.
 - If you want to collect data on more than one trace event, add the keyword values for each trace event and then use the sum in the field. For example, if you want to collect data on service calls (keyword value = 0x4) and AL function traces (keyword value = 0x8), then type **Microsoft-Dynamics365BusinessCentral-Server:0xC** in the field. In hexadecimal, the sum of 0x4 and 0x8 is 0xC.
6. Clear the check boxes above the **Additional providers** field for any providers that you do not want to collect data for.
7. To start recording data, choose the **Start Collection** button.
8. To stop recording data, choose the **Stop Collection** button.

The collected event trace data is stored in an event trace log (.etl) file in the location that you specified. You can view the data in the log file by using various industry-standard tools, such as PerfView. For information about how to use PerfView to view the event trace data, see [Use PerfView to View Event Trace Data](#).

To view event trace data from an event trace log file

1. Open PerfView.exe.
2. In PerfView, use the left pane to locate the .etl file that you want to view.

The left pane displays the current directory and the files that PerfView is set up to browse. To change a directory, choose a subdirectory from the list or type the directory (for example, c:\PerfLogs) in the text box at the top of the pane.

3. Double-click the .etl file that you want to view.

Several items appear in the left pane under the .etl file that you selected.

4. To view the event traces, double-click **Events**.

The **Events** window opens to display the contents of the .etl file. Trace events are listed in the left pane.

5. To view details about a trace event, double-click the trace event.

See Also

[Monitoring Business Central Server Events](#)

[Business Central Server Trace Events](#)

[Instrumenting an Application for Telemetry](#)

How to: Use LogMan to Collect Event Trace Data

3/31/2019 • 2 minutes to read

This article describes how to use logman to collect event trace data for Business Central Server. Logman (logman.exe) comes with the Windows Operating System. You can use it to create and manage event trace session and performance logs from the command prompt.

This article provides a brief introduction to using logman to collect trace event data for Business Central Server and telemetry events. For more detailed information about logman, see [Logman](#).

Collect event trace data

You can collect Business Central Server trace event data from two different trace event providers: **Microsoft-Dynamics365BusinessCentral-Server** and **Microsoft-Dynamics365BusinessCentral-Common**. **Microsoft-Dynamics365BusinessCentral-Server** is used for trace events like SQL traces, AL function traces, and session calls. **Microsoft-Dynamics365BusinessCentral-Common** is used for telemetry events.

Data that is collected with logman is stored in an event trace log (.etl) file.

The following steps give you an example of how to use logman.

1. Open the command prompt, and change to the directory that contains the `logman.exe` file.

This is typically `C:\Windows\System32`

2. At the command prompt, run one of the following commands to create a trace data collector.

For telemetry trace events:

```
logman create trace MyTelemetryTraceData -p Microsoft-Dynamics365BusinessCentral-Common -o c:\perflogs\MyTelemetryTraceData.etl
```

For server trace events:

```
logman create trace MyServerTraceData -p Microsoft-Dynamics365BusinessCentral-Server -o c:\perflogs\MyServerTraceData.etl
```

These commands will create event log files named `MyTelemetryTraceData.etl` and `MyServerTraceData.etl` in the `c:\perflogs` folder of your computer.

3. To start the trace session, run one of the following commands.

For telemetry trace events:

```
logman start MyTelemetryTraceData
```

For server trace events:

```
logman start MyServerTraceData
```

4. To stop the trace session, run one of the following commands.

For telemetry trace events:

```
logman stop MyTelemetryTraceData
```

For server trace events:

```
logman stop MyServerTraceData
```

The data is now stored in an .etl file.

View trace event data

There are various industry tools available for viewing data in .etl files.

For example, from the command line, you can use the [tracert command](#) to create dump files, summary, and report files. The following code creates files for the MyTelemetryTraceData_000001.etl file:

```
tracert c:\perflogs\MyTelemetryTraceData_000001.etl -o c:\perflogs\MyTelemetry-dmp.xml -of XML -summary  
c:\perflogs\MyTelemetry-summary.txt -report c:\perflogs\MyTelemetry-rpt.xml
```

You can also use PerView. For more information, see [Use PerfView to View Event Trace Data](#).

See Also

[Monitoring Business Central Server Events](#)

[Business Central Server Trace Events](#)

[Instrumenting an Application for Telemetry](#)

Monitoring Business Central Server Events with PowerShell

3/31/2019 • 3 minutes to read

Events that occur on the Business Central Server instances are recorded in event logs on the computer that is running Business Central Server. You can view the events by using Windows PowerShell as described in this article.

PowerShell Get-WinEvent Cmdlet

You can use the Get-WinEvent cmdlet of Windows PowerShell to view Business Central Server instance events and trace events in the event logs and event tracing log files on the Business Central Server computer. The Get-WinEvent cmdlet retrieves the same events that can be viewed in Event Viewer under **Applications and Services Logs > Microsoft > Dynamics365BusinessCentral** (see [Monitoring Business Central Server Events Using Event Viewer](#)).

The Get-WinEvent cmdlet includes several parameters that enable you to filter the events that you view and specify how the events are displayed. Windows PowerShell enables you can create scripts that perform complex operations for extracting and displaying specific event data. For more information about the Get-WinEvent cmdlet, see [Get-WinEvent](#).

For more information about installing and getting started with Windows PowerShell, see [Getting Started with Windows PowerShell](#).

To use the Get-WinEvent Cmdlet to view events

1. If you want to view events in a **Debug** log, ensure that the log is enabled. The **Admin** and **Operational** logs are enabled by default.

For information, see [To enable the Business Central Server Debug Log from Windows PowerShell](#).

2. On the computer that is running Business Central Server, start Window PowerShell.

For more information, see [Starting Windows PowerShell](#).

3. At the command prompt, enter the `Get-WinEvent` command. The following table provides some simple example commands.

TO VIEW	COMMAND
Events in the all Dynamics365BusinessCentral > Server logs	<code>Get-WinEvent -ProviderName Microsoft-Dynamics365BusinessCentral-Server</code>
Events in the all Dynamics365BusinessCentral > Common logs	<code>Get-WinEvent -ProviderName Microsoft-DynamicsNav-Common</code>
Events in the Dynamics365BusinessCentral > Server > Admin log	<code>Get-WinEvent -LogName Microsoft-Dynamics365BusinessCentral-Server/Admin</code>
Events in the Dynamics365BusinessCentral > Common > Admin log	<code>Get-WinEvent -LogName Microsoft-DynamicsNav-Common/Admin</code>

TO VIEW	COMMAND
Events in the Business Central Server Operational log	<code>Get-WinEvent -LogName Microsoft-Dynamics365BusinessCentral-Server/Operational</code>
Trace events in the Business Central Server Debug log	<code>Get-WinEvent -LogName Microsoft-Dynamics365BusinessCentral-Server/Debug -Oldest</code>

To enable the Debug Logs from Windows PowerShell

There are two debug logs for Business Central: **Microsoft-Dynamics365BusinessCentral-Server/Debug** and **Microsoft-DynamicsNav-Common/Debug**.

1. On the computer that is running Business Central Server, start Windows PowerShell as an administrator.
2. At the command prompt, run the following commands:

```
wevtutil.exe set-log "Microsoft-Dynamics365BusinessCentral-Server/Debug" /q:true /e:true
```

```
wevtutil.exe set-log "Microsoft-DynamicsNav-Common/Debug" /q:true /e:true
```

TIP

You can also enable the Debug log from Event Viewer. For more information, see [Enable Analytic and Debug Logs](#).

Filtering Business Central Server Events

You can filter the events that you view in a Business Central Server log by setting the *FilterXPath* parameter of the `Get-WinEvent` cmdlet. The following examples illustrate how you can use the *FilterXPath* parameter to filter the Business Central Server events.

Example 1

The following example uses the `Get-WinEvent` cmdlet to view errors in the Business Central Server Admin log for the tenant *MyTenant1* on the server instance *MyNavServerInstance1*.

```
Get-WinEvent -LogName 'Microsoft-Dynamics365BusinessCentral-Server/Admin' -FilterXPath "[System[(Level=2)]] and *[EventData[Data[@Name='tenantId'] and (Data = 'MyTenant1')]] and *[EventData[Data[@Name='serverInstanceName'] and Data='MyNavServerInstance1']]" | Format-List -Property Message-
```

Example 2

The following is an example of a Windows PowerShell script that you can create and run to view trace events in the Business Central Server Debug log. The script returns the start and stop AL function trace events that take more than four seconds to execute on the tenant *MyTenant1* of the server instance *MyNavServerInstance1*.

```

$maxAllowedSeconds = 4

$xPath = "[System[(EventID = 400 or EventID = 401)]] and " +
    "[EventData[Data[@Name='tenantId'] and (Data = 'MyTenant1')]] and " +
    "[EventData[Data[@Name='serverInstanceName'] and Data='MyNavServerInstance1']]"

$events = Get-WinEvent -LogName 'Microsoft-Dynamics365BusinessCentral-Server/Debug' -FilterXPath $XPath -
Oldest -MaxEvents 10000

Write-Host "List of AL functions that took more than $maxAllowedSeconds seconds to execute :" -
ForegroundColor DarkYellow

for($i = 0; $i -lt $events.Length; $i+=2)
{
    $seconds = ($events[$i + 1].TimeCreated - $events[$i].TimeCreated).Seconds

    if ($seconds -ge $maxAllowedSeconds )
    {
        Write-Host $events[$i].Message `r`n -ForegroundColor Magenta
    }
}

```

You can create the script by using, for example, Notepad or Windows PowerShell Integrated Scripting Environment (ISE). You save the script as .ps1 file type, and then run it from the Windows PowerShell.

See Also

[Monitoring Business Central Server Events](#)

[Business Central Server Trace Events](#)

[Monitoring Business Central Server](#)

[Monitoring Business Central Server Using Performance Counters](#)

[Event Viewer](#)

Turn Off or Limit Telemetry Trace Events

6/25/2019 • 2 minutes to read

The application and platform can emit many telemetry trace events, which can be collected using various event trace tools. For example, telemetry trace events are recorded in the Business Central Server channel logs, which you can see in Event Viewer, under **Applications and Services Logs > Microsoft > Dynamics365BusinessCentral > Common > Admin**.

The number of events can place a large demand on the logging resources on the computer running the Business Central Server instance. To help alleviate this demand, the Business Central Server instance includes a configuration setting called **Diagnostic Trace Level** (`TraceLevel` in the `customsettings.config` file) that enables you to specify the lowest severity level of customer telemetry trace events that are emitted from the application, or even turn off telemetry events altogether. Custom telemetry trace events have IDs from 700-712.

To configure the **Diagnostic Trace Level** setting, you can use the Business Central Server Administration tool, modify the Business Central Server instance configuration file (`CustomSettings.config`) directly, or use the [Set-NAVServerConfiguration cmdlet](#) of the Business Central Administration Shell.

TIP

Custom telemetry events are generated by calls to the `SENDTRACETAG` method in code. For more information, see [Instrumenting an Application for Telemetry](#).

Use the Business Central Server Administration tool

1. To start the Business Central Server Administration tool, select **Start**, and in the **Search programs and files** box, type **Microsoft Dynamics365 Business Central Administration**, and then choose the related link.
2. In the left pane, under **Console root**, select the Business Central Server instance.
3. In the center pane, select the **Edit** button.
4. Under **General**, set the **Diagnostic Trace Level**:

You use this setting to filter out lower-level events from being emitted. For example, if you set this setting to **Error**, only **Error** and **Critical** events will be emitted.

Set to **Off** if you do not want to emit telemetry trace events.

5. Select the **Save** button, and then select the **OK** button.

You must restart the Business Central Server instance for the changes to take effect.

6. To restart, the Business Central Server instance, in the left pane, select the Business Central computer.

Unless you are administering a remote computer, this is Business Central (local).

7. In the center pane, right-click an instance, and then select **Restart**.

Modify the CustomSettings.config file

1. Open the `CustomSettings.config` file for the Business Central Server instance in a text editor, such as Notepad.

By default, the file is located in the C:\Program Files\Microsoft Dynamics 365 Business Central\140\Service folder or C:\Program Files\Microsoft Dynamics 365 Business Central\140\Service\Instances\
<instancename> folder (for multitenant installations).

2. Set the **TraceLevel** setting to **Critical**, **Error**, **Warning**, **Normal** (this corresponds to the **Information** level), **Verbose**, or **Off**.
3. Save the file, and then restart the Business Central Server instance.

Use the Business Central Administration Shell

1. Start the Business Central Administration Shell.
2. At the command prompt, run the following command:

```
Set-NAVServerConfiguration -ServerInstance MyServerInstance -KeyName TraceLevel -KeyValue level -ApplyTo All
```

Substitute `MyServerInstance` with the name of the Business Central Server instance and `level` with either `Critical`, `Error`, `Warning`, `Normal`, `Verbose`, or `Off`.

For more information about how to use the Business Central Administration Shell, see [Business Central PowerShell Cmdlets](#) and [Set-NAVServerConfiguration Cmdlet](#).

See Also

[Monitoring Business Central Server Events Using Event Viewer](#)

[Monitoring Business Central Server Events](#)

[Configuring Business Central Server](#)

Monitoring Long Running SQL Queries using the Event Log

3/31/2019 • 2 minutes to read

Microsoft Dynamics NAV 2017 is the first version that allows long running SQL queries to be logged to the Windows Event Log. The queries are logged when the application communicates with the database and the call to the database takes too long.

Defining Long Running SQL Queries

The time logged in long running SQL queries is the time spent on the called database as seen from the server. There are multiple reasons that can cause this delay, such as the database waiting for a lock to be released, or the database executing an operation that performs badly due to missing indexes.

The threshold of when a query is logged is controlled in the configuration value of the *SqlLongRunningThreshold* key. The default value is 1000 milliseconds (ms). For more information about *SqlLongRunningThreshold*, see [Configuring Business Central Server](#), database settings section.

Changing Configuration Values

With Business Central, some of the configuration values for the server can be changed in the memory of the server, without doing a server restart. To change the threshold dynamically to 2000 ms, run the Business Central Administration Shell as Administrator and then type the following PowerShell cmdlet:

```
Set-NAVServerConfiguration -ServerInstance <ServerInstanceName> -KeyName SqlLongRunningThreshold -KeyValue 2000 -ApplyTo Memory
```

See Also

[Troubleshooting: Using the Event Log to Monitor Long Running SQL Queries](#)

[Troubleshooting: Analyzing Long Running SQL Queries Involving FlowFields by Disabling SmartSQL](#)

[Set-NAVServerConfiguration](#)

[Tools for Monitoring Performance Counters and Events](#)

[Monitoring Business Central Server Using Performance Counters](#)

[Monitoring Business Central Server Events](#)

Optimizing SQL Server Performance with Business Central

3/31/2019 • 2 minutes to read

The following articles describe how to optimize performance in Dynamics 365 Business Central when accessing data from the SQL Server database.

[Setting SQL Compatibility Level to Optimize Database Performance](#)

[Data Access](#)

[Table Keys and Performance](#)

[Bulk Inserts](#)

[AL Database Methods and Performance on SQL Server](#)

[Query Objects and Performance](#)

[Configuring Query Hints for Optimizing SQL Server Performance with Business Central](#)

[Troubleshooting: Analyzing Long Running SQL Queries Involving FlowFields by Disabling SmartSQL](#)

[Troubleshooting: Using Query Store to Monitor Query Performance in Business Central](#)

[Troubleshooting: Using the Event Log to Monitor Long Running SQL Queries in Business Central](#)

See Also

[Installation Considerations for Microsoft SQL Server](#)

[Microsoft SQL Server documentation](#)

Setting SQL Compatibility Level to Optimize Database Performance

3/31/2019 • 2 minutes to read

If your Business Central database is running on Azure SQL Database or SQL Server 2016 or later, set the database's compatibility level to match the database server. This will equip the database with the latest optimization features of Azure SQL Database or SQL Server. This is particularly relevant for demonstration databases that are installed by using the Business Central Setup because the default compatibility level matches SQL Server 2014.

To change the compatibility level

You change the compatibility level of the database by using SQL Server Management Studio. There are two ways to do this:

- Open the database properties, select the **Options** page, and then set the **Compatibility Level**:

For more information, see [View or Change the Compatibility Level of a Database](#).

- Run the following query:

```
ALTER DATABASE <database name> SET COMPATIBILITY_LEVEL = { 140 | 130 }
```

where:

- `<database name>` is the name of the database to be modified.
- `140` sets the database to be compatible with SQL Server 2017
- `130` sets the database to be compatible with SQL Server 2016 and Azure SQL Database

For more information, see [ALTER DATABASE \(Transact-SQL\) Compatibility Level](#).

NOTE

The compatibility level for Azure SQL Database is subject to change. Refer to Azure SQL Database documentation for latest compatibility level.

See Also

[Optimizing SQL Server Performance](#)
[Microsoft SQL Server documentation](#)

Data Access

3/31/2019 • 9 minutes to read

Data that is needed in the client goes through the following path from the Business Central Server to the SQL Server database:

1. If the data is cached in the Business Central Server data cache, it is returned.
2. If the data is not cached in the Business Central Server data cache, it is fetched from SQL Server over the network as follows:
 - a. If the data resides in SQL Servers data cache, it is returned.
 - b. If the data does not reside in SQL Servers data cache, it is fetched from storage and returned.

Business Central Server data caching

In Business Central, the data cache is shared by all users who are connected to the same Business Central Server instance. This means that after one user has read a record, a second user who reads the same record gets it from the cache. In earlier versions of Business Central, the data cache was isolated for each user.

The following AL functions utilize the cache system:

- GET
- FIND
- FINDFIRST
- FINDLAST
- FINDSET
- COUNT
- ISEMPY
- CALCFIELDS

There are two types of caches, global and private:

- Global cache is for all users connected to a Business Central Server instance.
- Private cache is per user, per company, in a transactional scope. Data in a private cache for a given table and company is flushed when a transaction ends.

The cache that is used is determined by the lock state of a table. If a table is not locked, then the global cache is queried for data; otherwise, the private cache is queried.

Results from query objects are not cached.

For a call to any of the **FIND** functions, 1024 rows are cached. You can set the size of the cache by using the **Data Cache Size** setting in the Business Central Server configuration file. The default size is 9, which approximates a cache size of 500 MB. If you increase this number by one, then the cache size doubles.

You can bypass the cache by using the [SELECTLATESTVERSION method \(Database\)](#).

Business Central synchronizes caching between Business Central Server instances that are connected to the same database. By default, the synchronization occurs every 30 seconds.

You can set the cache synchronization interval by using the *CacheSynchronizationPeriod* parameter in the CustomSettings.config file. This parameter is not included in the CustomSetting.config file by default, so you must add it manually using the following format:

```
<add key="CacheSynchronizationPeriod" value="hh:mm:ss" />
```

For example, to set the interval to 50 seconds, set the `value` to `"00:00:50"`. For more information about the CustomSettings.config file, see [Configuring Business Central Server](#).

Business Central Server connections to SQL Server

Starting from Microsoft Dynamics NAV 2013, the Business Central Server uses ADO.NET to connect to the SQL Server database. Installations of Microsoft Dynamics NAV 2009 and earlier uses ODBC to connect to the SQL Server database.

The ADO.NET interface is a managed data access layer that supports SQL Server connection pooling, which can dramatically decrease memory consumption by Business Central Server. SQL Server connection pooling also simplifies deployment of the Business Central three-tier architecture for deployments where the three tiers are installed on separate computers. Specifically, administrators are no longer required to manually create SPNs or to set up delegation when the client, Business Central Server, and SQL Server are on separate computers.

There is no longer a one-to-one correlation between the number of client connections and the number of SQL Server connections. In earlier versions of Business Central, each SQL Server connection could consume up to 40 MB of memory. Additionally, memory allocation is now in managed memory, which is generally more efficient than unmanaged memory.

Records are retrieved using Multiple Active Result Sets (MARS). methods such as NEXT, FIND('-'), FIND('+'), FIND('>'), and FIND('<') are generally faster with MARS than the server cursors that earlier versions of Business Central used.

Data read/write performance

AL functions COUNT and AVERAGE formulas can use SIFT indexes. For more information, see [CALCSUMS method \(Record\)](#) and [CALCFIELDS method \(Record\)](#). MIN and MAX formulas use SQL Server MIN and MAX functions exclusively.

RecordIds and SQL Variant columns in a table do not prevent the use of BULK inserts. For more information, see [Bulk Inserts](#).

In most cases, filtering on FlowFields issues a single SQL statement. In earlier versions of Business Central, filtering on FlowFields issued an SQL statement for each filtered FlowField and for each record in the table in order to calculate the filtered FlowFields. The exceptions in Business Central in which filtering on FlowFields does not issue a single SQL statement are as follows:

- You use the `ValuelsFilter` option on a field and the field has a value.
- A second predicate is specified on a source field and the field that is used for the second predicate has a value. For example, when you specify the [CalcFormula Property](#) for a FlowField, you can specify table filters in the **Calculation Formula** window. If you specify two or more filters on the same source field, then filtering does not issue a single SQL statement.
- You specify **Validated** for the [SecurityFiltering Property](#) on a record. This value for the **SecurityFiltering** property means that each record that is part of the calculation must be verified for inclusion in the security filter.

In most cases, calling the FIND or NEXT functions after you have set the view to include only marked records issues a single SQL statement. In earlier versions of Business Central, calling FIND or NEXT functions that have marked records issued an SQL statement for each mark. There are some exceptions if many records are marked. For more information, see [MARKEDONLY method \(Record\)](#).

Using SQL Server table partitioning

As of Microsoft Dynamics NAV 2018, the use of SQL Server table and index partitioning is a supported configuration. The data of partitioned tables and indexes is divided into units that can be spread across more than one filegroup in a SQL Server database. All partitions of a single index or table must reside in the same database. The table or index is treated as a single logical entity when queries or updates are performed on the data. Prior to SQL Server 2016 SP1, partitioned tables and indexes were not available in every edition of SQL Server.

Partitioning large tables or indexes can have the following manageability and performance benefits:

- You can perform maintenance operations on one or more partitions more quickly. The operations are more efficient because they target only these data subsets, instead of the whole table. For example, you can choose to rebuild one or more partitions of an index.
- You might be able to improve query performance, based on the types of queries you frequently run and on your hardware configuration. When SQL Server performs data sorting for I/O operations, it sorts the data first by partition. SQL Server accesses one drive at a time, and this might reduce performance. To improve data sorting performance, stripe the data files of your partitions across more than one disk by setting up a RAID (redundant array of independent disks). In this way, although SQL Server still sorts data by partition, it can access all the drives of each partition at the same time.
- You can use partitioning to distribute parts of tables to different IO sub systems. For example, you could archive data for old transactions on slow and inexpensive disks and keep current data on solid-state drives (SSD). You can improve performance by enabling lock escalation at the partition level instead of a whole table. This can reduce lock contention on the table.

For more general information about partitioned tables and indexes in SQL Server, see [Partitioned Tables and Indexes](#).

How Business Central supports partitioning

If you have altered tables in a Business Central database to make them partitioned tables, the synchronization engine, which is responsible for mapping the logical metamodel to physical tables, will respect this configuration during upgrades. After a schema upgrade, even if tables have been dropped and recreated, the partitioning strategy applied to the original tables will be added to the upgraded tables. You can create a partitioned table or index in SQL Server by using SQL Server Management Studio or Transact-SQL.

NOTE

For partitioning to work, the partition column must be part of the clustering key on the table.

Table Partitioning Example

This example uses Transact-SQL to change table **G_L Entry** to be partitioned on the **Posting Date** field, with data partitioned on the year, and where all partitions are aligned to the PRIMARY file group.

1. In SQL query editor, create a partition function that creates partitions that divide on year (this can be used for partitioning multiple tables):

```
CREATE PARTITION FUNCTION [DataHistoryPartitionmethod] (datetime)
AS RANGE LEFT FOR VALUES (
    '20151231 23:59:59.997',
    '20161231 23:59:59.997',
    '20171231 23:59:59.997',
    '20181231 23:59:59.997' )
GO
```

2. Create a partition scheme that maps partitions to file groups. In this example, all partitions are mapped to the PRIMARY file group (this can be used for partitioning multiple tables):

```
CREATE PARTITION SCHEME DataHistoryPartitionScheme
AS PARTITION DataHistoryPartitionmethod ALL TO ([PRIMARY])
GO
```

3. In the Dynamics NAV Development Environment, add the **Posting Date** field to the primary key.

For more information, see [Table Keys](#).

4. In the Transact-SQL Editor, partition table **G_L Entry** by using the previously defined partition scheme:

```
ALTER TABLE [dbo].[G_L Entry]
DROP CONSTRAINT [G_L Entry$0]
GO

ALTER TABLE [dbo].[G_L Entry]
ADD CONSTRAINT [G_L Entry$0] PRIMARY KEY CLUSTERED
(
[$companyId], [Entry No_], [Posting Date]
)
ON DataHistoryPartitionScheme( [Posting Date] )
GO
```

TIP

SQL Server Management Studio includes the **Create Partition Wizard** to help you create partitioning functions, partitioning schemes, as well as changing a table to be partitioned. For more information, see [Create Partitioned Tables and Indexes](#).

Using SQL Server data compression

As of Business Central April 2019, it is possible to configure data compression directly in table metadata by using the [CompressionType property](#) in AL or CSIDE. Previously, compression could only be configured in SQL Server. You use data compression to help reduce the size of selected tables in the database. In addition to saving space, data compression can help improve performance of I/O-intensive workloads because the data is stored in fewer pages and queries will read fewer pages from disk. This is especially useful if your storage system is based on disks and not SSD.

However, extra CPU resources are required on the database server to compress and decompress the data while data is exchanged with the Business Central Server.

With the **CompressionType** property, you can configure row or page type compression or configure the table not to use compression. With these compression settings, Business Central table synchronization process will make changes to the SQL Server table, overwriting the current compression type, if any. You can choose to control data compression directly on SQL Server by setting the **CompressionType** property to **Unspecified**, in which case table synchronization process will not control the data compression.

To evaluate whether a table is a good candidate to compress, you can use the stored procedure

`sp_estimate_data_compression_savings` in SQL Server. For more information, see

[sp_estimate_data_compression_savings \(Transact-SQL\)](#).

Because SQL Server supports data compression on the partition level, you can combine SQL Server data compression with table partitioning (see the previous section) to achieve flexible data archiving on historical parts of a large table, without having the CPU overhead on the active part of the table.

NOTE

Prior to SQL Server 2016 SP1, compression was not available in every edition of SQL Server.

For more general information about table compression in SQL Server, see [Data Compression](#). For guidance on strategy, capacity planning, and best practices for data compression, see [Data Compression: Strategy, Capacity Planning and Best Practices](#).

See Also

[Query Objects and Performance](#)

Table Keys and Performance in Business Central

3/31/2019 • 2 minutes to read

When you write AL code that searches through a subset of the records in a table, you must consider what keys are defined for the table and write code that optimizes for the keys. For example, the entries for a specific customer are usually a small subset of a table containing entries for all the customers.

Defining Keys to Improve Performance

The time that it takes to complete a loop through a subset of records depends on the size of the subset. If a subset cannot be located and read efficiently, then performance deteriorates.

To maximize performance, you must define the keys in the table that support the code that you run. You must then specify these keys correctly in your code.

For example, to retrieve the entries for a specific customer, you apply a filter to the Customer No. field in the Cust. Ledger Entry table. To run the code efficiently on Microsoft SQL Server, you must define a key in the table that has Customer No. as the first field.

The table could have the following keys:

- Entry No.
- Customer No., Posting Date

The following is an example of code that finds a subset of records.

```
SETRANGE("Customer No.", '1000');  
IF FIND('-') THEN  
  REPEAT  
  UNTIL NEXT = 0;
```

SQL Server automatically chooses which index to use in order to retrieve data in the most efficient way. SQL Server calculates the cost of retrieving data using different indexes and then chooses the path that has the smallest cost. For Business Central, that calculation is based only on the statistical distribution of values in a column.

For example, if a table contains 1000 rows and a column in the table contains either the value 0 or the value 1, then that column is said to have a low selectivity. If instead a column contained the values ranging from 1 to 500 then the column is said to have a high selectivity. In the following code example, SQL Server chooses an index that contains the HighSelectivityColumn and then sorts the rows by the LowSelectivityColumn.

```
SETCURRENTKEY("LowSelectivityColumn");  
SETFILTER("LowSelectivityColumn", '1');  
SETFILTER("HighSelectivityColumn", '777');  
FIND('-')
```

See Also

[Data Access](#)

[Bulk Inserts](#)

[AL Database Methods and Performance on SQL Server](#)

Bulk Inserts

3/31/2019 • 2 minutes to read

By default, Business Central automatically buffers inserts in order to send them to Microsoft SQL Server at one time.

By using bulk inserts, the number of server calls is reduced, thereby improving performance.

Bulk inserts also improve scalability by delaying the actual insert until the last possible moment in the transaction. This reduces the amount of time that tables are locked; especially tables that contain SIFT indexes.

Application developers who want to write high performance code that utilizes this feature should understand the following bulk insert constraints.

Bulk Insert Constraints

If you want to write code that uses the bulk insert functionality, you must be aware of the following constraints.

Records are sent to SQL Server when the following occurs:

- You call `COMMIT` to commit the transaction.
- You call `MODIFY` or `DELETE` on the table.
- You call any `FIND` or `CALC` methods on the table.

Records are not buffered if any of the following conditions are met:

- The application is using the return value from an `INSERT` call; for example, "`IF (GLEntry.INSERT) THEN`".
- The table that you are going to insert the records into contains any of the following:
 - BLOB fields
 - Fields with the **AutoIncrement** property set to **Yes**

The following code example cannot use buffered inserts because it contains a `FIND` call on the **GL/Entry** table within the loop.

```
IF (JnlLine.FIND('-')) THEN BEGIN
    GLEntry.LOCKTABLE;
    REPEAT
        IF (GLEntry.FINDLAST) THEN
            GLEntry.NEXT := GLEntry."Entry No." + 1
        ELSE
            GLEntry.NEXT := 1;
        // The FIND call will flush the buffered records.
        GLEntry."Entry No." := GLEntry.NEXT ;
        GLEntry.INSERT;
    UNTIL (JnlLine.FIND('>') = 0)
END;
COMMIT;
```

If you rewrite the code, as shown in the following example, you can use buffered inserts.

```
IF (JnlLine.FIND('-')) THEN BEGIN
    GLEntry.LOCKTABLE;
    IF (GLEntry.FINDLAST) THEN
        GLEntry.Next := GLEntry."Entry No." + 1
    ELSE
        GLEntry.Next := 1;
    REPEAT
        GLEntry."Entry No.":= GLEntry.Next;
        GLEntry.Next := GLEntry."Entry No." + 1;
        GLEntry.INSERT;
    UNTIL (JnlLine.FIND('>') = 0)
END;
COMMIT;
// The inserts are performed here.
```

Disabling Bulk Inserts

Disabling bulk inserts can be helpful when you are troubleshooting failures that occur when inserting records. To disable bulk inserts, you set the *BufferedInsertEnabled* parameter in the CustomSettings.config file of the Business Central Server to **FALSE**. For more information, see [Configuring Business Central Server](#).

See Also

[Data Access](#)

[Table Keys and Performance](#)

[AL Database Methods and Performance on SQL Server](#)

[Query Objects and Performance](#)

AL Database Methods and Performance on SQL Server

3/31/2019 • 5 minutes to read

This topic describes the relationship between basic database functions in AL and SQL statements.

AL and SQL Statements

GET, FIND, and NEXT

The AL language offers several methods to retrieve record data. In Dynamics 365 Business Central, records are retrieved using multiple active result sets (MARS). Generally, retrieving records with MARS is faster than with server-side cursors. Additionally, each function is optimized for a specific purpose. To achieve optimal performance you must use the method that is best suited for a given purpose.

- **Record.GET** is optimized for getting a single record based on primary key values.
- **Record.FIND** is optimized for getting a single record based on the primary keys in the record and any filter or range that has been set.
- **Record.FIND('-')** and **Record.FIND('+')** are optimized for reading primarily from a single table when the application might not read all records. **FIND('-')** is implemented by issuing a self-tuning TOP X call, where X can change over time, based on statistics of the number of rows read.

The following are examples of scenarios in which you should use the **FIND('-')** function to achieve optimal performance:

- Before you post a general journal batch, you must check all journal lines for validity and verify that all lines balance. After the first line when an error is found, you do not have to retrieve the rest of the rows.
- If you want to fulfill multiple outstanding orders from a recent purchase but you do not know how many orders are covered by the purchase.
- **Record.FINDSET(ForUpdate, UpdateKey)** is optimized for reading the complete set of records in the specified filter and range. The *UpdateKey* parameter does not influence the efficiency of this method in Dynamics 365 Business Central, such as it did in Microsoft Dynamics NAV 2009.

FINDSET is not implemented by issuing a TOP X call.

- **Record.FINDFIRST** and **Record.FINDLAST** are optimized for finding the single first or last record in the specified filter and range.
- **Record.NEXT** can be called at any time. However, if **Record.NEXT** is not called as part of retrieving a continuous result set, then Business Central calls a separate SQL statement in order to find the next record.

Dynamic Result Sets

Any result set that is returned from a call to the find methods discussed in the previous section is dynamic. That means that the result set is guaranteed to contain any changes that you make further ahead in the result set. However this feature comes at a cost. If any modifications are made to a table which is being traversed, then Business Central might have to issue an extra SQL statement to guarantee that the result set is dynamic.

The following code shows how records are most efficiently retrieved. **FINDSET** is the most efficient method to use because this example reads all records.

```

IF FINDSET THEN
  REPEAT
    // Insert statements to repeat.
  UNTIL NEXT = 0;

```

CALCFIELDS, CALCSUMS, and COUNT

Each call to **CALCFIELDS**, **CALCFIELD**, **CALCSUMS**, or **CALCSUM** functions that calculates a sum requires a separate SQL statement unless the client has calculated the same sum or another sum that uses the same SumIndexFields or filters in a recent operation, and therefore, the result is cached.

Each **CALCFIELDS** or **CALCSUMS** request should be confined to use only one SIFT index. The SIFT index can only be used if:

- All requested sum-fields are contained in the same SIFT index.
- The filtered fields are part of the key fields specified in the SIFT index containing all the sum fields.

If neither of these requirements is fulfilled, then the sum will be calculated directly from the base table.

In Dynamics 365 Business Central, SIFT indexes can be used to count records in a filter provided that a SIFT index exists that contains all filtered fields in the key fields that are defined for the SIFT index.

SETAUTOCALCFIELDS

It is a common task to retrieve data and request calculation of associated FlowFields. The following example traverses customer records, calculates the balance, and marks the customer as blocked if the customer exceeds the maximum credit limit. Note the Customer record and associated fields are imaginary.

```

IF Customer.FINDSET() THEN REPEAT
  Customer.CALCFIELDS(Customer.Balance)
  IF (Customer.Balance > MaxCreditLimit) THEN BEGIN
    Customer.Blocked = True;
    Customer.MODIFY();
  END
  ELSE IF (Customer.Balance > LargeCredit) THEN BEGIN
    Customer.Caution = True;
    Customer.MODIFY();
  END;
UNTIL Customer.NEXT = 0;

```

In Dynamics 365 Business Central, you can do this much faster. First, we set a filter on the customer. This could also be done in Business Central 2009, but behind the scenes the same code as mentioned earlier would be executed. In Dynamics 365 Business Central, setting a filter on a record is translated into a single SQL statement.

```

Customer.SETFILTER(Customer.Balance, '>%1', LargeCredit);
IF Customer.FINDSET() THEN REPEAT
  Customer.CALCFIELDS(Customer.Balance)
  IF (Customer.Balance > MaxCreditLimit) THEN BEGIN
    Customer.Blocked = True;
    Customer.MODIFY();
  END
  ELSE IF (Customer.Balance > LargeCredit) THEN BEGIN
    Customer.Caution = True;
    Customer.MODIFY();
  END;
UNTIL Customer.NEXT = 0;

```

In the previous example, an extra call to CALCFIELDS still must be issued for the code to be able to check the value

of Customer.Balance. In Dynamics 365 Business Central, you can optimize this further by using the new **SETAUTOCALCFIELDS** method.

```
Customer.SETFILTER(Customer.Balance,'>%1', LargeCredit);
Customer.SETAUTOCALCFIELDS(Customer.Balance)
IF Customer.FINDSET() THEN REPEAT
  IF (Customer.Balance > MaxCreditLimit) THEN BEGIN
    Customer.Blocked = True;
    Customer.MODIFY();
  END
  ELSE IF (Customer.Balance > LargeCredit) THEN BEGIN
    Customer.Caution = True;
    Customer.MODIFY();
  END;
UNTIL Customer.NEXT = 0;
```

INSERT, MODIFY, DELETE, and LOCKTABLE

Each call to **INSERT**, **MODIFY**, or **DELETE** functions requires a separate SQL statement. If the table that you modify contains SumIndexes, then the operations will be much slower. As a test, select a table that contains SumIndexes and execute one hundred **INSERT**, **MODIFY**, or **DELETE** operations to measure how long it takes to maintain the table and all its SumIndexes.

The **LOCKTABLE** function does not require any separate SQL statements. It only causes any subsequent reading from the table to lock the table or parts of it.

See Also

[Table Keys and Performance](#)

[Bulk Inserts](#)

[GET Method \(Record\)](#)

[FIND Method \(Record\)](#)

[NEXT Method \(Record\)](#)

[FINDSET Method \(Record\)](#)

[FINDFIRST Method \(Record\)](#)

[FINDLAST Method \(Record\)](#)

[CALCFIELDS Method \(Record\)](#)

[CALCFIELD Method \(FieldRef\)](#)

[CALCSUMS Method \(Record\)](#)

[CALCSUM Method \(FieldRef\)](#)

[SETAUTOCALCFIELDS Method \(Record\)](#)

[INSERT Method \(Record\)](#)

[MODIFY Method \(Record\)](#)

[DELETE Method \(Record\)](#)

[LOCKTABLE Method \(Record\)](#)

Query Objects and Performance

3/31/2019 • 3 minutes to read

A *query* is an object in Dynamics 365 Business Central that you use to specify a set of data that you want to read from the Business Central database. You can query the database to retrieve one or more fields from a single table or multiple tables. You can specify how to join the tables in the query. You can specify totaling methods on fields, such as sums and averages. This topic describes how to design queries and table keys in the most efficient way.

FlowFields in Queries

A sub-query is automatically added to the SQL statement to retrieve each FlowField in a query. This allows Business Central to retrieve all the data in one request.

IMPORTANT

You cannot use a FlowField on a virtual table in a query because this cannot be converted automatically into a SQL statement.

Covering Indexes

When you use a query to select a subset of fields in a table, you should consider taking advantage of the covering index strategy. A *covering index* is the index that contains all output fields required by the operation performed on that index. A covering index data access strategy can greatly improve performance because the database must retrieve only data from the index instead of finding data by using the index and then retrieving the data in the clustered index. A covering index data access strategy can be used when the following conditions are true:

- All columns in a given data item are part of a single Business Central key.
- All columns that are used in the Dataltem table filters are part of the same Business Central key.
- If two Dataltems are linked, then the field on the parent Dataltem that links the two Dataltems (the **Reference Field** on the **DataltemLink** property), must be part of the same Business Central key as the columns in the child Dataltem.

The SQL Server optimizer automatically chooses a covering index strategy whenever possible.

For more information about SQL Server covering indexes, see [SQL Server Optimization](#).

For more information about SQL Server clustered and non-clustered indexes, see [Types of Indexes](#).

Covering SIFT Indexes

Similar to how indexes can be used to retrieve data for a query, SIFT indexes can be used to retrieve data for a query that contains totals. SIFT totals are maintained after each insert, modify, or delete call, and so some or all of the totals are already calculated. A SIFT index can be used when the following conditions are true:

- The query contains at least one aggregated column with **Method Type** set to **Totals** and with **Method** set to either **Sum**, **Count**, or **Average**.
- If a Dataltem contains an aggregated column, then all columns under that Dataltem must be aggregated columns, must use either the **Sum**, **Count**, or **Average** method, and must be part of a SumIndexField defined on a single Business Central key.

- In a query in which you have aggregations but not on all DataItems, then for the DataItems without aggregations, the columns are part of a SumIndexField.
- All non-aggregated columns under the DataItem that have aggregation are part of the key fields defined for the same SIFT index.
- All columns that are used in the DataItem table filters are part of the same Business Central key.
- If two DataItems are linked, then the field on the parent DataItem that links the two DataItems (the **Reference Field** in the **DataItemLink** property) must be part of the same Business Central key as the columns in the child DataItem.

Business Central Server automatically use a SIFT index for query objects whenever possible.

Differences Between Query and Record Result Sets

Business Central does not do any caching for query result sets. When you run a query, Business Central always gets the data directly from SQL Server.

Query result sets are not guaranteed to be dynamic, whereas record result sets are always dynamic. This means that if you insert or modify data in result set row that you have not yet looped through, then it is not guaranteed that the query result set includes those changes.

Enabling and Disabling Selected Query Hints

SQL Server query optimizer will try to select the best execution plan for SELECT, INSERT, UPDATE, and DELETE statements. Most of the time, query optimizer makes the right choice. [Query hints](#) are strategies that can be enforced by the SQL Server query processor to override any execution plan that the query optimizer might select for a query. The Business Central Server instance includes configuration settings that let you enable or disable the use of the selected query hints on the database.

For more information, see [Configuring Query Hints for Optimizing SQL Server Performance with Business Central](#).

See Also

[Query Object](#)

[Optimizing SQL Server Performance with Business Central](#)

Troubleshooting: Using Query Store to Monitor Query Performance in Business Central

3/31/2019 • 2 minutes to read

What is SQL Server Query Store?

The SQL Server Query Store feature provides you with insight on database query plan choice and performance. It simplifies database performance troubleshooting by helping you quickly find performance differences caused by query plan changes. Query Store automatically captures a history of queries, plans, and runtime statistics, and retains these for your review.

Where is SQL Server Query Store available?

SQL Server Query Store is available in SQL Server (starting with SQL Server 2016) and in Azure SQL Database.

What are the common scenarios for using the Query Store feature?

Query Store keeps a history of compilation and runtime metrics throughout query execution. With this information, you can get some answers on questions about your workload, such as:

- What was the last n queries executed on the database?
- What were the number of executions for each query?
- Which queries had the longest average execution time within last hour?
- Which queries had the biggest average physical IO reads in last 24 hours, with corresponding average row count and execution count?

Do you want to read more?

Please visit the SQL Server documentation for more information on setup, configuration and usage of Query Store:

[Monitoring Performance By Using the Query Store](#)

[Operating the Query Store in Azure SQL Database](#)

See Also

[Installation Considerations for Microsoft SQL Server](#)

Troubleshooting: Using the Event Viewer to Monitor Long Running SQL Queries in Business Central

3/31/2019 • 2 minutes to read

This topic shows how you can use the Event Viewer to monitor long running SQL queries and decide which ones can be candidates for optimization.

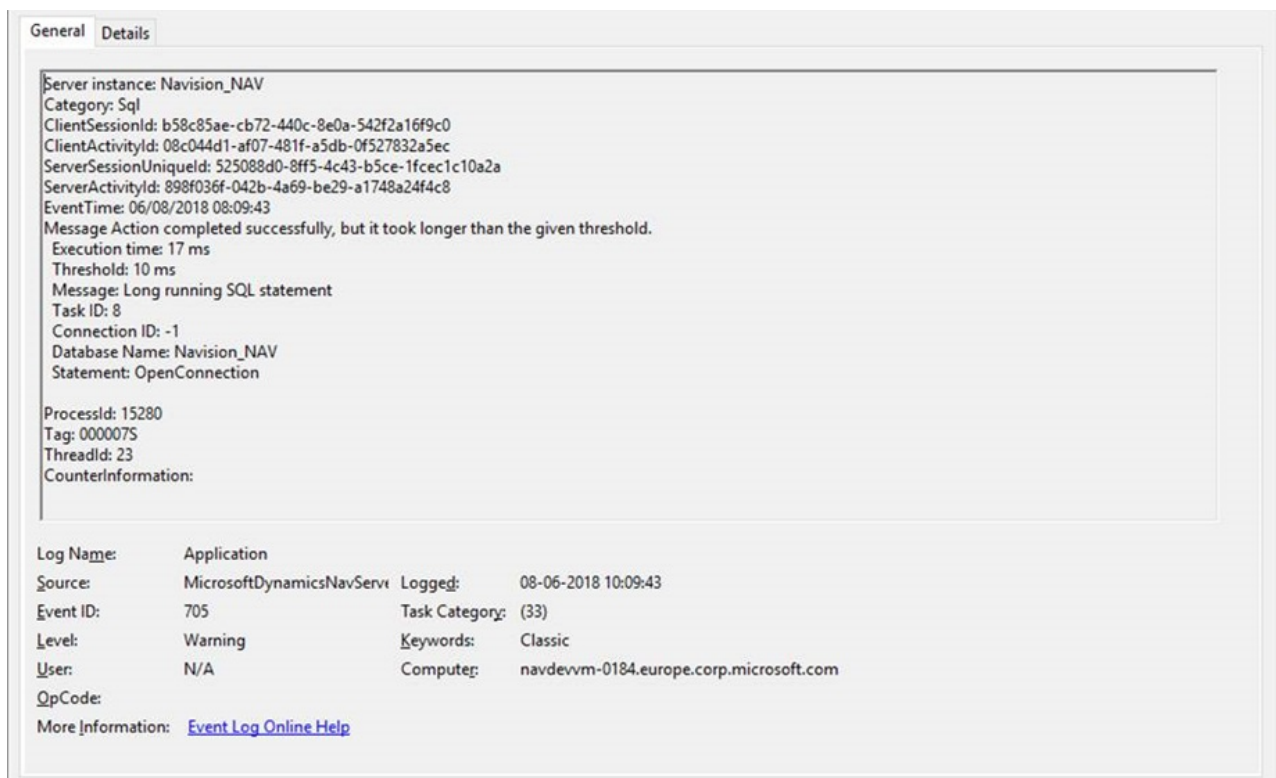
Resolution

Identifying long running SQL queries can be a good starting point when doing a performance analysis. To find which SQL queries performed slower than expected, open the Event Viewer and go to the Windows Logs Application.

NOTE

The SQL queries that exceed the set threshold will be displayed in the Application window of the Event Viewer as *Warning*.

If the value of the [SqlLongRunningThreshold](#) key was set to the default value of 1000 milliseconds, you will see the message: "Action completed successfully, but it took longer than the given threshold." for actions that took longer than that. To meet your performance expectations in production, you can set the threshold to a different value without doing a server restart. For more information on how you can do this, see [Monitoring Long Running SQL Queries using the Event Log](#).



The screenshot shows the Windows Event Viewer with the 'Details' tab selected. The event is a warning (yellow triangle) with the following details:

- Server instance: Navision_NAV
- Category: Sql
- ClientSessionId: b58c85ae-cb72-440c-8e0a-542f2a16f9c0
- ClientActivityId: 08c044d1-af07-481f-a5db-0f527832a5ec
- ServerSessionUniqueId: 525088d0-8ff5-4c43-b5ce-1fcec1c10a2a
- ServerActivityId: 898f036f-042b-4a69-be29-a1748a24f4c8
- EventTime: 06/08/2018 08:09:43
- Message: Action completed successfully, but it took longer than the given threshold.
- Execution time: 17 ms
- Threshold: 10 ms
- Message: Long running SQL statement
- Task ID: 8
- Connection ID: -1
- Database Name: Navision_NAV
- Statement: OpenConnection
- ProcessId: 15280
- Tag: 0000075
- ThreadId: 23
- CounterInformation:

Below the details, there is a table with the following information:

Log Name:	Application
Source:	MicrosoftDynamicsNavServe
Event ID:	705
Level:	Warning
User:	N/A
OpCode:	
More Information:	Event Log Online Help

At the bottom, there is a table with the following information:

Log Name:	Application
Source:	MicrosoftDynamicsNavServe
Event ID:	705
Level:	Warning
User:	N/A
OpCode:	
More Information:	Event Log Online Help

In some cases, you can see what caused the delay by looking at the SQL statement that was generated by the code listed in the `AL CallStack` column. The code below shows which AL method generated a slow performing query.

```
Server instance: Navision_NAV
Category: Sql
ClientSessionId: 00000000-0000-0000-0000-000000000000
ClientActivityId: 828c9342-891a-4631-8eb3-a1da7304fdc9
ServerSessionUniqueId: 24b32889-9be9-439f-b86c-9615d5e51319
ServerActivityId: 19bf285d-a8f2-42b6-a4c0-4afe9fb5b4b4
EventTime: 06/08/2018 08:10:15
Message Action completed successfully, but it took longer than the given threshold.
  Execution time: 33 ms
  Threshold: 10 ms
  Message: Long running SQL statement
  Task ID: 3
  Connection ID: 2
  Database Name: Navision_NAV
  Statement: SELECT "2161"."timestamp","2161"."User","2161"."Default Execute Time","2161"."Current Job Queue
Entry" FROM "SQLDATABASE".dbo."CRONUS International Ltd_$Calendar Event User Config_" "2161" WITH(UPDLOCK)
WHERE ("2161"."User"=@0) OPTION(OPTIMIZE FOR UNKNOWN)
    AppObjectType: CodeUnit
    AppObjectId: 2160
    AL CallStack: "Calendar Event Mangement"(CodeUnit 2160).GetCalendarEventUserConfiguration line 2
"Calendar Event Mangement"(CodeUnit 2160).FindJobQueue line 1
"Calendar Event Mangement"(CodeUnit 2160).FindOrCreateJobQueue line 1
"Calendar Event Mangement"(CodeUnit 2160).CreateOrUpdateJobQueueEntry line 1
"Calendar Event"(Table 2160).Schedule line 12
"Calendar Event"(Table 2160).OnInsert(Trigger) line 1
"Calendar Event Mangement"(CodeUnit 2160).CreateCalendarEventForCodeunit line 6
"Create Telemetry Cal. Events"(CodeUnit 1352).OnRun(Trigger) line 5

ProcessId: 15280
Tag: 000007L
ThreadId: 10
CounterInformation:
```

See Also

[Troubleshooting: Analyzing Long Running SQL Queries Involving FlowFields by Disabling SmartSQL](#)

[Monitoring Long Running SQL Queries using the Event Log](#)

[Tools for Monitoring Performance Counters and Events](#)

[Business Central Server Administration Tool](#)

[Troubleshooting: Using Query Store to Monitor Query Performance in Business Central](#)

[SQL Trace](#)

Session Timeout Settings and Configuration

3/31/2019 • 8 minutes to read

When you start a client, like the Business Central Web client or Dynamics NAV Client connected to Business Central, a connection is established with the Business Central Server instance and a corresponding session is added on Business Central Server.

Business Central Server includes several timeout settings that determine when a session closes as a result of inactivity over the client connection, lost client connection, or closing of the client. To help you configure the timeout settings, this document provides an overview of how the session timeouts work and answers some basic questions about session behavior.

Session timeout settings overview

This section provides an overview of the settings that are available in Business Central to control when a Business Central Server session for a Business Central client connection times out and closes. Some of the settings are set on Business Central Server and others are set for the Dynamics NAV Client connected to Business Central or for the Business Central Web client. For more details about using these settings, see the other sections in this topic.

Business Central Server timeout settings

The following table describes the session timeout settings that are used by Business Central Server.

SETTING	DESCRIPTION	REMARKS
ClientServicesReconnectPeriod	The amount of time during which a client can reconnect to an existing session on Business Central Server before a session closes.	For more information, see Configuring How Long a Session Remains Open after the Client Connection is Lost .
ClientServicesIdleClientTimeout	The interval of time that a Business Central client connection can remain inactive before the session is closed.	For more information, see Configuring How Long a Session Remains Open When the Client Connection is Inactive .
ClientServicesKeepAliveInterval	Specifies the interval (in seconds) between keep-alive messages that are sent from the Dynamics NAV Client connected to Business Central to Business Central Server.	This setting is also used, in part, to define the reconnect period when a connection is lost. For more information, see Keeping inactive sessions alive .

These settings are available in the CustomSettings.config file of Business Central Server. For more information about this file, see [Configuring Business Central Server](#).

Business Central Web client timeout settings

The following table describes the session timeout settings that are used by the Business Central Web client.

SETTING	DESCRIPTION	REMARKS
SessionTimeout	Specifies the amount of time that session remains open when there is no activity over the connection from the Business Central Web client to Business Central Server.	For more information, see Configuring Business Central Server .

This setting is available in the `navsettings.json` configuration file of the Business Central Web Server. For more information about this file, see [Configuring Web Server](#).

Configuring How Long a Session Remains Open When the Client Connection is Inactive

Inactivity on a connection is when the Business Central client is not sending messages to Business Central Server. Controlling when a session will timeout and close because of inactivity is different for the Dynamics NAV Client connected to Business Central and the Business Central Web client.

Configuring the inactive session timeout for the Dynamics NAV Client connected to Business Central

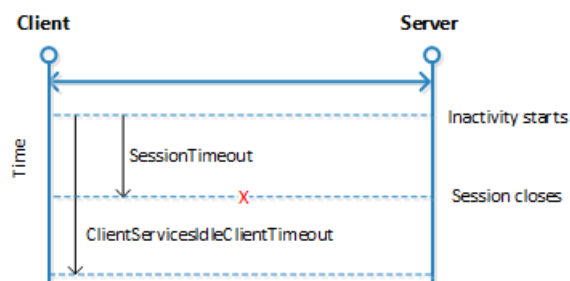
When the Dynamics NAV Client connected to Business Central is inactive, the session will remain open until the time period that is specified by the `ClientServicesIdleClientTimeout` setting has passed, provided that the client has not been stopped or the connection to Business Central Server has not been lost. The default value of the `ClientServicesIdleClientTimeout` setting is **MaxValue**, which means that there is no time limit so the session will remain active indefinitely.

Configuring the inactive session timeout for the Business Central Web client

There are two settings that control when a Web client session closes because of inactivity on a connection:

- `ClientServicesIdleClientTimeout` setting on Business Central Server.
- `SessionTimeout` setting on the Business Central Web Server.

The session closes according to the setting that has the shortest time period. By default, the `ClientServicesIdleClientTimeout` setting is set to **MaxValue**, which means no time limit, and the `SessionTimeout` setting is 00:20:00 (20 minutes). This means that when client connection is inactive, a session will close after 20 minutes. The following figure illustrates the timeout behavior:



The `SessionTimeout` setting enables you to set the Business Central Web client inactive session timeout different than for the Dynamics NAV Client connected to Business Central, which is only controlled by the `ClientServicesIdleClientTimeout` setting. Typically, you will set the inactive session timeout period on Business Central Web client connections shorter than for the Dynamics NAV Client connected to Business Central.

Keeping inactive sessions alive

To keep an inactive session alive, the Dynamics NAV Client connected to Business Central uses the Windows Communication Framework (WCF) reliable sessions feature. When the Dynamics NAV Client connected to Business Central is inactive, reliable sessions automatically sends messages from the Dynamics NAV Client connected to Business Central to Business Central Server. You control the interval of the keep-alive messages by setting the `ClientServicesKeepAliveInterval` setting on the Business Central Server. The default value of the `ClientServicesKeepAliveInterval` setting is 120 seconds (2 minutes).

For most installations, the `ClientServicesKeepAliveInterval` setting default value sufficient for keeping sessions open until the `ClientServicesIdleClientTimeout` setting period elapses. However, when Business Central Server is installed behind a load balancer, which is the case on Microsoft Azure, you might have to adjust the value the `ClientServicesKeepAliveInterval` setting to prevent sessions from closing before the expected session timeout. A load balancer typically has an idle timeout setting that it uses to determine whether to redirect connections.

However, you want a stable connection between the Dynamics NAV Client connected to Business Central and Business Central Server. If there is no activity on the client connection for duration of the load balancer's idle timeout setting, then the load balancer might redirect the client connection to another server. To avoid this condition, we recommend that you set the *ClientServicesKeepAliveInterval* to half the value of the load balancer's idle timeout setting.

NOTE

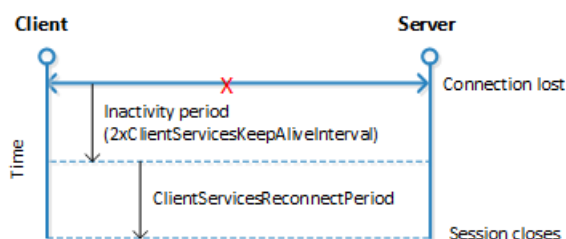
The idle timeout on Azure is around 4 minutes, so the default setting of *ClientServicesKeepAliveInterval* (2 minutes) should be sufficient.

Configuring How Long a Session Remains Open after the Client Connection is Lost

Occasionally, a Business Central client can lose the network connection to Business Central Server. You can use *ClientServicesReconnectPeriod* setting on Business Central Server to control how long a session remains open after the connection is lost to allow time for the client to reconnect to the session.

The time a session remains open actually depends two settings: *ClientServiceKeepAliveInterval* and *ClientServicesReconnectPeriod*. The *ClientServiceKeepAliveInterval* setting is used to specify an initial inactivity period. The initial inactivity period is equal to two times the *ClientServiceKeepAliveInterval* setting value. After this initial inactivity period, the session remains open for the time period that is specified *ClientServicesReconnectPeriod* setting. By default, the *ClientServiceKeepAliveInterval* setting is 120 seconds (2 minutes) and the *ClientServicesReconnectPeriod* setting is 10 minutes. This means that Business Central Server waits approximately 14 minutes for the client to reconnect before closing the session.

The following figure illustrates the reconnect session timeout behavior.



The process that occurs when a client does not reconnect to the session is explained as follows:

1. The connection is lost and the initial inactivity period starts (default is 4 minutes).
2. After the initial inactivity period, the service channel enters a faulted state.

When the service channel is in the faulted state, Business Central Server considers the session with the client as orphaned and waits for it to reconnect.

3. If the client does not reconnect within the time period that is specified by the *ClientServicesReconnectPeriod* setting (default is 10 minutes), then Business Central Server closes the session.
4. The session is then removed from the **Active Session** table in the Business Central.

FAQ

This section answers some typical questions about session timeout.

How long does Business Central Server wait when the Dynamics NAV Client connected to Business Central is inactive before closing a session??

With Dynamics NAV Client connected to Business Central, by default, Business Central Server will wait indefinitely

as long as the client has not been stopped or the connection to Business Central Server has not been lost. With the Business Central Web client, the session will remain active for 20 minutes. The Dynamics NAV Client connected to Business Central and Business Central Web client include configuration settings that you can use to change the inactivity timeout period. For more information, see [Configuring How Long a Session Remains Open When the Client Connection is Inactive](#).

What happens to the session if I end the Dynamics NAV Client connected to Business Central by using Task Manager?

If the Dynamics NAV Client connected to Business Central is waiting for a response from Business Central Server, as is the case with a modal dialog, then the session remains open until the time period that is specified by the *ClientServicesReconnectPeriod* setting expires. When the Window Client process is ended, the service channel will enter a faulted state. Business Central Server considers the session with the Microsoft Dynamics NAV client as orphaned and waits for it to reconnect.

What happens to the session if the client loses the connection to Business Central Server?

By default, it will take approximately 14 minutes for the Business Central Server to close the current session. The time it takes to close the session is in part determined by the *ClientServicesReconnectPeriod* setting on Business Central Server plus an initial 10 minute inactivity period. For more information, see [Configuring How Long a Session Remains Open after the Client Connection is Lost](#).

What happens if the session is still active when Business Central Server tries to close it?

1. The server stops any executing threads when the next statement is to be executed and the current call stack is aborted so any uncommitted transactions will be rolled back.
2. The server cancels any callbacks to the client (similar to waiting for the response to a Confirm dialog).
3. The session is closed, and then removed from the Active Session table.

Preparing Dynamics 365 for Sales for Integration

3/31/2019 • 3 minutes to read

This article describes how to set up and configure Dynamics 365 for Sales for integrating with Business Central. You must complete the following tasks:

1. Create a user for connecting to and synchronizing data from Business Central.
2. Install the Business Central integration solution for Dynamics 365 for Sales.

This task is optional. You only need to complete this task if you want the functionality that is provided by the Business Central integration solution.

IMPORTANT

To perform the tasks in this topic, you must have the System Administrator security role or equivalent privileges in Dynamics 365 for Sales.

Create a Dynamics 365 for Sales User for Connecting to Business Central

As a minimum, this must be a non-interactive user account that has the required privileges to write, read, modify, and delete data in the entities that will be integrated with Business Central.

You will use this user account to set up the connection to Dynamics 365 for Sales from Business Central.

IMPORTANT

You should not use this account to sign in to Dynamics 365 for Sales to modify entities records that are integrated with Business Central because the changes will be ignored by integration synchronization jobs in Business Central.

Create the connection user

- For more information about how to create users in Dynamics 365 for Sales, see <http://go.microsoft.com/fwlink/?LinkID=616518>.

Install the Business Central Integration Solution

Business Central includes a solution that enables users to access coupled records in Business Central, such as customers and items, from records in Dynamics 365 for Sales, such as accounts and products. The solution adds a link on the Dynamics 365 for Sales record pages that opens the coupled Business Central record. The solution is also used to display information from Business Central in a part on certain entity records in Dynamics 365 for Sales, such as accounts. Installing this solution is optional.

1. From Business Central installation media (DVD), copy either the DynamicsNAVIntegrationSolution_v8.zip or DynamicsNAVIntegrationSolution_v9.zip file to your computer.

These files are located in the **CrmCustomization** folder. This file is the solution package.

Use the zip version that matches the Dynamics 365 for Sales SDK version. Use DynamicsNAVIntegrationSolution_v8.zip for legacy services running CRM or Dynamics 365 for Sales version 8.x and earlier. Use DynamicsNAVIntegrationSolution_v9.zip for Dynamics 365 for Sales versions

9.0 and later.

2. In Dynamics 365 for Sales, import the DynamicsNAVIntegrationSolution.zip as a solution.

This step adds the **Business Central Connection** entity and **Business Central Account Statistics** entity in the system and additional items such as Business Central integration security roles.

For more information about how to manage solutions in Dynamics 365 for Sales, <http://go.microsoft.com/fwlink/?LinkID=616519>.

3. (Optional) Set up the **Business Central Connection** entity to display in the **Settings** area of Dynamics 365 for Sales.

This enables Dynamics 365 for Sales users who are assigned the **Business Central Admin** role to modify the entity in Dynamics 365 for Sales. For more information about how to modify entities in Dynamics 365 for Sales, see <http://go.microsoft.com/fwlink/?LinkID=616521>.

4. Assign the **Business Central Integration Administrator** role to the Business Central connection user.
5. Assign the **Business Central Integration User** role to all users who require the use of the features provided by the Business Central integration solution.

If you install the Business Central integration solution after you have set up the connection to Dynamics 365 for Sales from in Business Central, you must modify the connection setup to point to the URL of the Business Central Web client.

Development in AL

4/4/2019 • 3 minutes to read

Extensions are a programming model where functionality is defined as an addition to existing objects and defines how they are different or modify the behavior of the solution. This section explains how you can develop extensions using the development environment for Dynamics 365 Business Central.

If you are new to building extensions, we recommend that you read this document to get an understanding of the basics and terms you will encounter while working. Next, follow the [Getting Started with AL](#) to set up the tools.

TIP

If you are looking for the C/SIDE documentation, visit our [Dynamics NAV library](#).

Understanding objects in the development environment

All functionality in Dynamics 365 Business Central is coded in objects. The extension model is object-based; you create new objects, and extend existing objects depending on what you want your extension to do. Table objects define the table schema that holds data, page objects represent the pages seen in the user interface and codeunits contain code for logical calculations and for the application behavior. These objects are stored as code, known as AL code, and are saved in files with the `.al` file extension. The AL Language extension also supports the multi-root functionality which allows you to work with multiple AL folders within one workspace. For more information on how to group a set of disparate project folders into one workspace, see [Working with multiple AL project folders within one workspace](#).

NOTE

A single `.al` file may contain multiple objects.

There are two other special objects which are specifically used for building extensions. Table extension objects and page extension objects are used for defining additive or overriding changes to table or page objects. For example, an extension for managing a business that sells organic food may define a table extension object for the Item table that contains two additional fields, `Organic` and `Produced Locally`. The `Organic` and `Produced Locally` fields are not usually present in the Item table, but through the table extension these data fields will now be available to store data in and to access from code. You can then use the page extension object to display the fields that you added to the table object.

NOTE

Extension objects can have a name with a maximum length of 30 characters.

You have several options for creating new objects with the AL Language extension for Visual Studio Code. For more information about the objects that you can create for your extension,

see [AL Development Environment](#).

Developing extensions in Visual Studio Code

Using the AL Language extension for Visual Studio Code, you will get the benefits of a modern development environment along with seamless publishing and execution integration with your Dynamics 365 Business Central tenant. For more information on getting up and running, see [Getting Started with AL](#).

Visual Studio Code and the AL Language extension lets you do the following tasks:

- Create new files for your solution
- Get assistance with creating the appropriate configuration and setting files
- Use code snippets that provide templates for coding application objects
- Get compiler validation while coding
- Press Ctrl+F5 to publish your changes and see your code running

For more information, see [Visual Studio Code Docs](#).

TIP

If you have previous experience working with the C/SIDE development environment and need an overview of some of the changes between the two development environments, see [Differences in the Development Environments](#).

Designer

The Designer works in the client itself allowing design of pages using a drag-and-drop interface. The Designer allows building extensions in the client itself by rearranging fields, adding fields, and previewing the page design. For more information, see [Using Designer](#).

Compiling and deploying

Extensions are compiled as .app package files. The .app package file can be deployed to the Dynamics 365 Business Central server. An .app package contains the various artifacts that deliver the new functionality to the Dynamics 365 Business Central deployment as well as a manifest that specifies the name, publisher, version, and other attributes of the extension. For information about the manifest, see [JSON Files](#).

Submitting your app

When all development and testing is done, you can submit your extension package to AppSource. Before you submit the extension package, we encourage you to read the checklist to help facilitating the validation. For more information, see [Checklist for Submitting Your App](#).

See Also

[Getting Started with AL](#)

[Getting Started Developing Connect Apps for Dynamics 365 Business Central](#)

[Keyboard Shortcuts](#)

[AL Development Environment](#)

[FAQ for Developing in AL](#)

Getting Started with AL

5/24/2019 • 3 minutes to read

To get started writing extensions for Dynamics 365 Business Central you will need a Dynamics 365 Business Central tenant, Visual Studio Code, and the AL Language extension. Visual Studio Code is a cross platform editor that you will use for coding and debugging.

Steps to set up a sandbox environment and Visual Studio Code

Go through the following steps to set up a sandbox environment. With this you get sample code that compiles and runs with just a few commands.

NOTE

If you want to create a container-based sandbox, see [Get started with the Container Sandbox Development Environment](#). For information about which sandboxes you can choose, see [Choosing Your Dynamics 365 Business Central Development Sandbox Environment](#).

1. Sign up for a [Dynamics 365 Business Central sandbox](#).
2. Download [Visual Studio Code](#).
3. Download the [AL Language extension](#).
4. Press **Ctrl+,** to open the **user settings** window; here you can modify the [telemetry settings](#).
5. Press **Alt+A, Alt+L** to trigger the **AL Go!** command, and then choose **Microsoft cloud sandbox**.

NOTE

If you want to change your configuration at a later point in time, choose **Add Configuration** on the right side, and then choose one of the available options.

6. Enter the credentials you provided for the signup, and then symbols will automatically start downloading. To manually download the symbols, press **Ctrl+Shift+P** and select **AL: Download symbols**.
7. Press **F5** to deploy and run the extension on your online sandbox tenant.

You now have a HelloWorld sample that compiles and runs. The JSON files in the project are automatically updated with the settings that allows you to press **F5** to build and deploy the solution.

Tips and tricks

- Use **Ctrl+Space** to activate IntelliSense.
- Always use the `.al` extension on new files.
- Use the built-in [snippets for code](#) by starting typing `t` and pick from the list.
- Create objects within the right object ranges, see [Object Ranges in Dynamics 365 Business Central](#).
- Build and get inspired by our sample library on [GitHub](#).

- Use **Ctrl+Shift+P** to clear the credentials cache if you want to deploy against a different environment.

JSON file settings

There are two JSON files in the project; the `app.json` file and the `launch.json` file. The files are automatically generated for your project. For more information, see [JSON files](#).

Telemetry settings

By default, Visual Studio Code is set up with a telemetry system to enable that data and errors are sent to Microsoft. If you do not want to send telemetry data, you can change the `telemetry.enableTelemetry` setting from `true` to `false`.

To modify the telemetry setting, press **Ctrl+,** in Visual Studio Code and choose the **user settings** window, which opens the `settings.json` file, and then add `telemetry.enableTelemetry` and set it to `false`.

```
"telemetry.enableTelemetry": false,
```

TIP

The `settings.json` file contains user and workspace settings, these options can be modified to suit your preference. If you want to modify Visual Studio Code editor options and functional behavior settings, see [User and Workspace Settings](#).

The symbol file

The symbol file contains metadata of the application. This is what your extension is being built on, and therefore the symbol file must be present. If it is not present, you will be prompted to download it. For more information about the platform symbol file, see [Symbols](#).

Installing and publishing an extension

To make your extension available to users, the package must be published to a specific Microsoft Dynamics 365 Business Central Server instance. The extension can be installed for one or more tenants. For more information about how to install and publish an extension, see [How to: Publish and Install an Extension](#).

Controlling user access to publishing extensions

The access to publishing extensions is controlled on a user or user group basis by the **D365 EXTENSION MGT** permission set.

NOTE

If you add new permission sets and want to control the access to developing and publishing extensions, you must include indirect read and write permissions to the NavApp table (read – for downloading symbols, write – for publishing the app) in the permission set.

To prohibit a user from publishing, just remove the user from the **D365 EXTENSION MGT** permission set.

Next steps

Now that you have the tools and the HelloWorld example up and running, you might want to try to create a small sample app in AL. This walkthrough guides you through how to create a simple app adding objects, code, and publishing the app to your tenant. For more information, see [Building Your First Sample Extension With Extension Objects, Install Code, and Upgrade Code](#).

See Also

[AL Development Environment](#)

[FAQ for Developing in AL](#)

[Syntax](#)

[Building Your First Sample Extension With Extension Objects, Install Code, and Upgrade Code](#)

Choosing Your Dynamics 365 Business Central Development Sandbox Environment

3/31/2019 • 2 minutes to read

To get started developing for Dynamics 365 Business Central it is important to understand the different options you have at hand. You can either choose to run a sandbox environment deployed as a Dynamics 365 Business Central service, or you can run a container-based image either hosted as an Azure VM or locally. Both options provide the AL development tools; the container-based sandbox additionally provides access to the C/SIDE development tools. You can also choose to run a sandbox environment with production data using the **Business Central Admin Center**. For more information, see [Business Central Admin Center](#).

NOTE

When you publish an app to the online sandbox for testing, it is published within the scope of the service node that is hosting the sandbox. Upgrading the sandbox to a new version means that the sandbox is moved to another node that is running the new version. All apps are removed before the sandbox is moved because they will not be available on the new node. However, the data of the app is not removed, so you only have to re-publish and install the app to make it available. Apps that are published to the production environment are published within a global scope and downloaded to the service node and installed during the upgrade, which means that they will not disappear.

Sandbox Overview

The following topic outlines the most important capabilities on the offered development sandbox environments for Dynamics 365 Business Central.

CAPABILITY	ONLINE SANDBOX	CONTAINER SANDBOX
Deployment	Dynamics 365 Cloud Service managed by Microsoft	Azure VM or on-premises managed by ISV/VAR
Production data	Manually uploaded using Rapid Start packages. Or, available through the Business Central Admin Center .	Manually uploaded using Rapid Start packages
Production services	Manually configured	Not available
Cost	Part of the Business Central subscription	Locally hosted - free, Azure-hosted - cost incurred
Development	Full capabilities of the development environment. Designer functionality, such as: Add/Remove components, Move components, Set/clear Freeze pane, Edit captions	Full capabilities of the development environment. Designer functionality, such as: Add/Remove components, Move components, Set/clear Freeze pane, Edit captions
Tools	Visual Studio Code, Designer	Visual Studio Code, Designer, on-premise tools such as SQL Server Management Studio, and C/SIDE.

CAPABILITY	ONLINE SANDBOX	CONTAINER SANDBOX
Debugging	Enabled	Enabled
Database access	No	Yes
Extensions	Must be manually installed.	Must be manually installed.
From AppSource	Available.	Not available.
From File	Not available.	Available.
From Visual Studio Code	Available.	Available.

Getting Started

Based on the overview above and the requirements for your development environment, you can get started with a sandbox by following the links below:

- [Online Sandbox with Demo Data](#)
- [Online Sandbox with Production Data](#)
- [Container Sandbox](#)

See Also

[Getting Started with AL](#)

[Keyboard Shortcuts](#)

[AL Development Environment](#)

Building Your First Sample Extension With Extension Objects, Install Code, and Upgrade Code

5/3/2019 • 16 minutes to read

This walkthrough will guide you through all the steps that you must follow to create a sample extension in AL. New objects and extension objects will be added to the base application for a simple reward feature for customers. Every section of this exercise includes code that serves for installing, customizing, or upgrading this sample extension. The final result can be published and installed on your tenants.

About this walkthrough

This walkthrough illustrates the following tasks:

- Developing a sample extension with a table, a card page, and a list page.
- Deploying the sample extension to your development sandbox environment.
- Using the Dynamics 365 Business Central Designer to modify visual aspects of the extension.
- Creating extension objects that can be used to modify page and table objects.
- Initializing the database during the installation of the extension.
- Upgrading and preserving data during the upgrade of the extension.

Prerequisites

To complete this walkthrough, you will need:

- The Dynamics 365 Business Central tenant.
- Visual Studio Code.
- The AL Language extension for Visual Studio Code.

For more information on how to get started with your first extension for Dynamics 365 Business Central, see [Getting Started](#).

Rewards extension overview

The extension enables the ability to assign one of three reward levels to customers: GOLD, SILVER, and BRONZE. Each reward level can be assigned a discount percentage. Different types of objects available within the AL development environment will build the foundation of the user interface, allowing the user to edit the information. If you look for another option to update the layout of a page, you can use the Designer drag-and-drop interface. Additionally, this exercise contains the install code that will create the base for the reward levels. The upgrade code is run to upgrade the extension to a newer version and it will change the BRONZE level to ALUMINUM. Following all the steps of this walkthrough allows you to publish the extension on your tenant and create a possible new feature for your customers.

Reward table object

The following code adds a new table **50100 Reward** for storing the reward levels for customers. The table consists of three fields: **Reward ID**, **Description**, and **Discount Percentage**. For example, the **Description** field

must contain a value of type text and it cannot exceed the limit of 250 characters. The second field contains three properties that are used to set the range of the discount percentage assigned to every customer. Properties can be created for every field, depending on the scope.

TIP

Type `ttable` followed by the Tab key. This snippet will create a basic layout for a table object.

```
table 50100 Reward
{
    DataClassification = ToBeClassified;

    fields
    {
        // The "Reward ID" field represents the unique identifier
        // of the reward and can contain up to 30 Code characters.
        field(1;"Reward ID";Code[30])
        {
            DataClassification = ToBeClassified;
        }

        // The "Description" field can contain a string
        // with up to 250 characters.
        field(2;Description;Text[250])
        {
            // This property specified that
            // this field cannot be left empty.
            NotBlank = true;
        }

        // The "Discount Percentage" field is a Decimal numeric value
        // that represents the discount that will
        // be applied for this reward.
        field(3;"Discount Percentage";Decimal)
        {
            // The "MinValue" property sets the minimum value for the "Discount Percentage"
            // field.
            MinValue = 0;

            // The "MaxValue" property sets the maximum value for the "Discount Percentage"
            // field.
            MaxValue = 100;

            // The "DecimalPlaces" property is set to 2 to display discount values with
            // exactly 2 decimals.
            DecimalPlaces = 2;
        }
    }

    keys
    {
        // The field "Reward ID" is used as the primary key of this table.
        key(PK;"Reward ID")
        {
            // Create a clustered index from this key.
            Clustered = true;
        }
    }
}
```

For more information about table properties, see [Table Properties](#).

Reward card page object

The following code adds a new page **50101 Reward Card** for viewing and editing the different reward levels that are stored in the new **Reward** table. Pages are the primary object that a user will interact with and have a different behavior based on the type of page that you choose. The **Reward Card** page is of type Card and it is used to view and edit one record or entity from the **Reward** table.

TIP

Use the snippet `tpage, Page of type card` to create the basic structure for the page object.

```
page 50101 "Reward Card"
{
    // The page will be of type "Card" and will render as a card.
    PageType = Card;

    // The page will be part of the "Tasks" group of search results.
    UsageCategory = Tasks;

    // The source table shows data from the "Reward" table.
    SourceTable = Reward;

    // The layout describes the visual parts on the page.
    layout
    {
        area(content)
        {
            group(Reward)
            {
                field("Reward Id";"Reward ID")
                {
                    // ApplicationArea sets the application area that
                    // applies to the page field and action controls.
                    // Setting the property to All means that the control
                    // will always appear in the user interface.
                    ApplicationArea = All;
                }

                field(Description;Description)
                {
                    ApplicationArea = All;
                }

                field("Discount Percentage";"Discount Percentage")
                {
                    ApplicationArea = All;
                }
            }
        }
    }
}
```

For more information about the types of pages in AL, see [Pages Overview](#).

Reward list page object

The following code adds the **50102 Reward List** page that enables users to view the contents of the **Reward** table and edit specific records by selecting them and viewing them in the **Reward Card** page.

TIP

Use the snippet `tpage, Page of type list` to create the basic structure for the page object.

```
page 50102 "Reward List"
{
    // Specify that this page will be a list page.
    PageType = List;

    // The page will be part of the "Lists" group of search results.
    UsageCategory = Lists;

    // The data of this page is taken from the "Reward" table.
    SourceTable = Reward;

    // The "CardPageId" is set to the Reward Card previously created.
    // This will allow users to open records from the list in the "Reward Card" page.
    CardPageId = "Reward Card";

    layout
    {
        area(content)
        {
            repeater(Rewards)
            {
                field("Reward ID";"Reward ID")
                {
                    ApplicationArea = All;
                    ToolTip = 'Specifies the level of reward that the customer has at this point.';
                }

                field(Description;Description)
                {
                    ApplicationArea = All;
                }

                field("Discount Percentage";"Discount Percentage")
                {
                    ApplicationArea = All;
                }
            }
        }
    }
}
```

After you have created the objects, update the **startupObjectId** in the launch.json file to 50102, the ID of the **Reward List** page and select the Ctrl+F5 shortcut to see the new page in your sandbox environment. You will be asked to sign in to your Business Central.

TIP

Information about your sandbox environment and other environments is stored as configurations in the launch.json file. For more information, see [JSON Files](#).

Designer

Dynamics 365 Business Central Designer works in the browser and allows modifying the current page. It enables users to add existing table fields, move fields around, or remove fields from the page. Users can make changes to display the information they need, where they need it by using drag-and-drop components.

To show how the Designer changes the design of a page, you begin by adding two new fields to the **Reward** table.

These fields will be used later on to exemplify the Designer's properties.

```
field(4;"Minimum Purchase";Decimal)
{
    MinValue = 0;
    DecimalPlaces = 2;
}

field(5;"Last Modified Date";Date)
{
    // The "Editable" property sets a value that indicates whether the field can be edited
    // through the UI.
    Editable = false;
}
```

The **Last Modified Date** field requires constant changes to remain accurate. To keep it updated, triggers will be used. Triggers are predefined methods that are executed when certain actions happen. They are added by default when you use the `ttable` template, and now you can add code to the triggers.

```
// "OnInsert" trigger executes when a new record is inserted into the table.
trigger OnInsert();
begin
    SetLastModifiedDate();
end;

// "OnModify" trigger executes when a record in the table is modified.
trigger OnModify();
begin
    SetLastModifiedDate();
end;

// "OnDelete" trigger executes when a record in the table is deleted.
trigger OnDelete();
begin
end;

// "OnRename" trigger executes when a record in a primary key field is modified.
trigger OnRename();
begin
    SetLastModifiedDate();
end;

// On the current record, the value of the "Last Modified Date" field to the current
// date.
local procedure SetLastModifiedDate();
begin
    Rec."Last Modified Date" := Today();
end;
```

From this point, changes to the **Reward Card** page can be done either manually by adding the code below in Visual Studio Code or by using the Designer's functions. Both ways lead to the same results, but the Designer speeds up the process.

```
field("Minimum Purchase";"Minimum Purchase")
{
    ApplicationArea = All;
}

field("Last Modified Date";"Last Modified Date")
{
    ApplicationArea = All;
}
```

Using the F6 key shortcut in Visual Studio Code launches the browser and enters the Designer.

NOTE

Every time you start designing, you create a new extension and the changes you make in the Designer will apply to all users.

To add the same fields and customize the **Reward Card** page, follow the next steps:

- Choose the purple box to the right of the **Last Modified Date** field and select **Remove**.
- Navigate to the **Reward Card** page by choosing **+ new**.
- Select **More** from the Designer bar.
- Select **Field** from the Designer bar to show the list of available fields.
- Drag the **Minimum Purchase** and **Last Modified Date** fields from the list onto the page in the **General group**.
- Choose the **General** in the group caption to enable the value to be edited. Change the caption to **Info** and press **Enter**.

After making these adjustments, finish up your design by choosing **Stop Designing**, which allows you to name the extension with an option to download code, and save the extension for the tenant. If you choose not to download the code at the end, you can still pull the changes via the F7 key shortcut from Visual Studio Code. You can also uninstall the extension by opening the **Extension Management** page.

For more information about Designer, see [Designer](#).

Customer table extension object

The **Customer** table, like many other tables, is part of the Dynamics 365 Business Central service and it cannot be modified directly by developers. To add additional fields or to change properties on this table, developers must create a new type of object, a table extension. The following code creates a table extension for the **Customer** table and adds the `Reward ID` field.

TIP

Use the snippet `ttableext` to create a basic structure for the table extension object.

```

tableextension 50103 "Customer Ext" extends Customer
{
    fields
    {
        field(50100;"Reward ID";Code[30])
        {
            // Set links to the "Reward ID" from the Reward table.
            TableRelation = Reward."Reward ID";

            // Set whether to validate a table relationship.
            ValidateTableRelation = true;

            // "OnValidate" trigger executes when data is entered in a field.
            trigger OnValidate();
            begin
                // If the "Reward ID" changed and the new record is blocked, an error is thrown.
                if (Rec."Reward ID" <> xRec."Reward ID") and
                    (Rec.Blocked <> Blocked::" ") then
                    begin
                        Error('Cannot update the rewards status of a blocked customer.')
                    end;
                end;
            end;
        }
    }
}

```

Customer card page extension object

A page extension object can be used to add new functionality to pages that are part of the Dynamics 365 Business Central service. The following page extension object extends the **Customer Card** page object by adding a field control, **Reward ID**, to the **General group** on the page. The field is added in the layout section, while in the actions section the code adds an action to open the **Reward List** page.

TIP

Use the shortcuts `tpageext` to create the basic structure for the page extension object.

```

pageextension 50104 "Customer Card Ext" extends "Customer Card"
{
    layout
    {
        // The "addlast" construct adds the field control as the last control in the General
        // group.
        addlast(General)
        {
            field("Reward ID";"Reward ID")
            {
                ApplicationArea = All;

                // Lookup property is used to provide a lookup window for
                // a text box. It is set to true, because a lookup for
                // the field is needed.
                Lookup = true;
            }
        }
    }

    actions
    {
        // The "addfirst" construct will add the action as the first action
        // in the Navigation group.
        addfirst(Navigation)
        {
            action("Rewards")
            {
                ApplicationArea = All;

                // "RunObject" sets the "Reward List" page as the object
                // that will run when the action is activated.
                RunObject = page "Reward List";
            }
        }
    }
}

```

At this point, reward levels can be created and assigned to customers. To do that, update the `startupObjectId` value in `launch.json` to 21 and select the Ctrl+F5 key to open the page.

Help links

This app is relatively straightforward, but we want users of your app to be able to get unblocked and learn more just like all other users of Business Central. First, configure your app to get context-sensitive links to Help, and then apply tooltips to the fields in your pages.

Configure context-sensitive links to Help

At an app level, you can specify where the Help for your functionality is published in the `app.json` file. Then, for each page in your app, you specify which Help file on that website is relevant for that particular page. For more information, see [Configure Context-Sensitive Help](#).

Open the `app.json` file, and then change the value of the `contextSensitiveHelpUrl` property to point at the right location on your website. In this example, you publish Help for your app at <https://mysite.com/documentation>.

```
"contextSensitiveHelpUrl": "https://mysite.com/documentation",
```

Next, you set the `ContextSensitiveHelpPage` property for the **Reward Card** and **Reward List** pages:

```
// The target Help topic is hosted on the website that is specified in the app.json file.
ContextSensitiveHelpPage = 'sales-rewards';
```

The following example illustrates the properties for the **Reward List** page after you have specified the context-sensitive Help page.

```
page 50102 "Reward List"
{
    // Specify that this page will be a list page.
    PageType = List;

    // The page will be part of the "Lists" group of search results.
    UsageCategory = Lists;

    // The target Help topic is hosted on the website that is specified in the app.json file.
    ContextSensitiveHelpPage = 'sales-rewards';

    // The data of this page is taken from the "Reward" table.
    SourceTable = Reward;

    // The "CardPageId" is set to the Reward Card previously created.
    // This will allow users to open records from the list in the "Reward Card" page.
    CardPageId = "Reward Card";

    ...
}
```

You can specify the same relative link for **Reward Card**, **Reward List**, and the customization of the **Customer** page, or you can specify different targets. For more information, see [Page-level configuration](#).

Add tooltips

Even the best designed user interface can still be confusing to some. It can be difficult to predict specifically what users will find confusing, and that is why the base application includes tooltips for all controls and actions. For more information, see [Help users get unblocked](#).

For the purposes of this walkthrough, add the following tooltip to the properties of the **Reward ID** field on all three pages:

```
ToolTip = 'Specifies the level of reward that the customer has at this point.';
```

The following example illustrates the tooltip:

```
field("Reward ID","Reward ID")
{
    ApplicationArea = All;
    ToolTip = 'Specifies the level of reward that the customer has at this point.';
}
```

If you now deploy the app, you will be able to read the tooltip text for the **Reward ID** field, and if you choose the *Learn more* link or press Ctrl+F1, a new browser tab opens the equivalent of

```
https://mysite.com/documentation/sales-rewards .
```



...TOMER CARD | WORK DATE: 4/8/2019



✓ SAVED

10000 · Adatum Corporation

New Document

Request Approval

Navigate

Customer

More options



General

Show more

No. 10000 ...

Name Adatum Corporation

Balance (\$) 0.00

Balance Due (\$) 0.00

Credit Limit (\$) 0.00

Blocked ▼

Total Sales **78,771.10**

Costs (\$) 40,255.70

Reward ID ▼

Reward ID

Specifies the level of reward that the customer has at this point.

Press Ctrl+F1 to learn more

Address & Contact

Install code

After installing the extension, the **Reward List** page is empty. This is the result of the fact that the **Reward** table is also empty. Data can be entered manually into the **Reward** table by creating new records from the **Reward List** page. However, this task slows down the process, especially because the **Reward** table should be initialized with a standard number of reward levels when the extension is installed. To solve this, install codeunits can be used. A codeunit is an object that can be used to encapsulate a set of related functionality represented by procedures and variables. An install codeunit is a codeunit with the Subtype property set to Install. This codeunit provides a set of triggers that are executed when the extension is installed for the first time and when the same version is re-installed.

In this example, the following install codeunit initializes the **Reward** table with three records representing the 'GOLD', 'SILVER', and 'BRONZE' reward levels.

TIP

Use the shortcuts `tcodeunit` and `ttrigger` to create the basic structure for the codeunit and trigger.

```

codeunit 50105 RewardsInstallCode
{
    // Set the codeunit to be an install codeunit.
    Subtype = Install;

    // This trigger includes code for company-related operations.
    trigger OnInstallAppPerCompany();
    var
        Reward : Record Reward;
    begin
        // If the "Reward" table is empty, insert the default rewards.
        if Reward.IsEmpty() then begin
            InsertDefaultRewards();
        end;
    end;

    // Insert the GOLD, SILVER, BRONZE reward levels
    procedure InsertDefaultRewards();
    begin
        InsertRewardLevel('GOLD', 'Gold Level', 20);
        InsertRewardLevel('SILVER', 'Silver Level', 10);
        InsertRewardLevel('BRONZE', 'Bronze Level', 5);
    end;

    // Create and insert a reward level in the "Reward" table.
    procedure InsertRewardLevel(ID : Code[30]; Description : Text[250]; Discount : Decimal);
    var
        Reward : Record Reward;
    begin
        Reward.Init();
        Reward."Reward ID" := ID;
        Reward.Description := Description;
        Reward."Discount Percentage" := Discount;
        Reward.Insert();
    end;
}

```

For more information about install code, see [Writing Extension Install Code](#).

Upgrade code

When you upgrade an extension to a newer version, if any modifications to the existing data are required to support the upgrade, you must write upgrade code in an upgrade codeunit. In this example, the following upgrade codeunit contains code that changes the BRONZE reward level to customer records to ALUMINUM. The upgrade codeunit will run when you run the `Upgrade-NAVApp` cmdlet.

IMPORTANT

Remember to increase the `version` number of the extension in the app.json file.

```

codeunit 50106 RewardsUpgradeCode
{
    // An upgrade codeunit includes AL methods for synchronizing changes to a table definition
    // in an application with the business data table in SQL Server and migrating existing
    // data.
    Subtype = Upgrade;

    // "OnUpgradePerCompany" trigger is used to perform the actual upgrade.
    trigger OnUpgradePerCompany();
    var
        InstallCode : Codeunit RewardsInstallCode;
        Reward : Record Reward;

        // "ModuleInfo" is the current executing module.
        Module : ModuleInfo;
    begin
        // Get information about the current module.
        NavApp.GetCurrentModuleInfo(Module);

        // If the code needs to be upgraded, the BRONZE reward level will be changed into the
        // ALUMINUM reward level.
        if Module.DataVersion.Major = 1 then begin
            Reward.Get('BRONZE');
            Reward.Rename('ALUMINUM');
            Reward.Description := 'Aluminum Level';
            Reward.Modify();
        end;
    end;
}

```

For more information about writing and running upgrade code, see [Upgrading Extension](#).

Conclusion

This walkthrough demonstrated how an extension can be developed. The main AL objects and extension objects were used to store the reward levels, to view, and to edit them. The Designer was introduced as an alternative to modify visual aspects of page objects and to customize them from the web client instead of using code. Up to this point, the table and the page objects were empty, but the install codeunits were added and allowed to initialize the **Reward** table with a standard number of reward levels when the extension was installed. An upgrade code section was also included in this exercise to create a full picture of all processes involved when an extension is built. As a result, a user is enabled to assign one of the three reward levels to a customer and to change this scenario by upgrading the version of the extension.

TIP

To try building a more advanced Customer Rewards sample extension, see [Building an Advanced Sample Extension](#).

See Also

[Developing Extensions](#)

[Getting Started with AL](#)

[How to: Publish and Install an Extension](#)

[Converting Extensions V1 to Extensions V2](#)

[Configure Context-Sensitive Help](#)

Using Designer

5/21/2019 • 7 minutes to read

When developing extensions in the AL development environment, you have a wide range of possibilities. Designer in Dynamics 365 Business Central complements the development experience in Visual Studio Code, as it provides an easy and convenient way of making immediate adjustments to your design by simply dragging and dropping the components on the page.


Here is a quick overview of capabilities in **Designer**:

FEATURES	APPLIES TO
Add components	fields and columns
Move components	fields, columns, cues, parts, actions and action groups
Remove components	fields, columns, cues, parts, actions and action groups
Change field importance, like showing in collapsed FastTab header or under Show More	fields
Exclude field from Quick Entry	fields, columns
Set freeze pane and clear freeze pane	columns
Adjust column width	columns
Edit caption	FastTab, cards, FactBox
Save extension/download code	general
Preview design	general

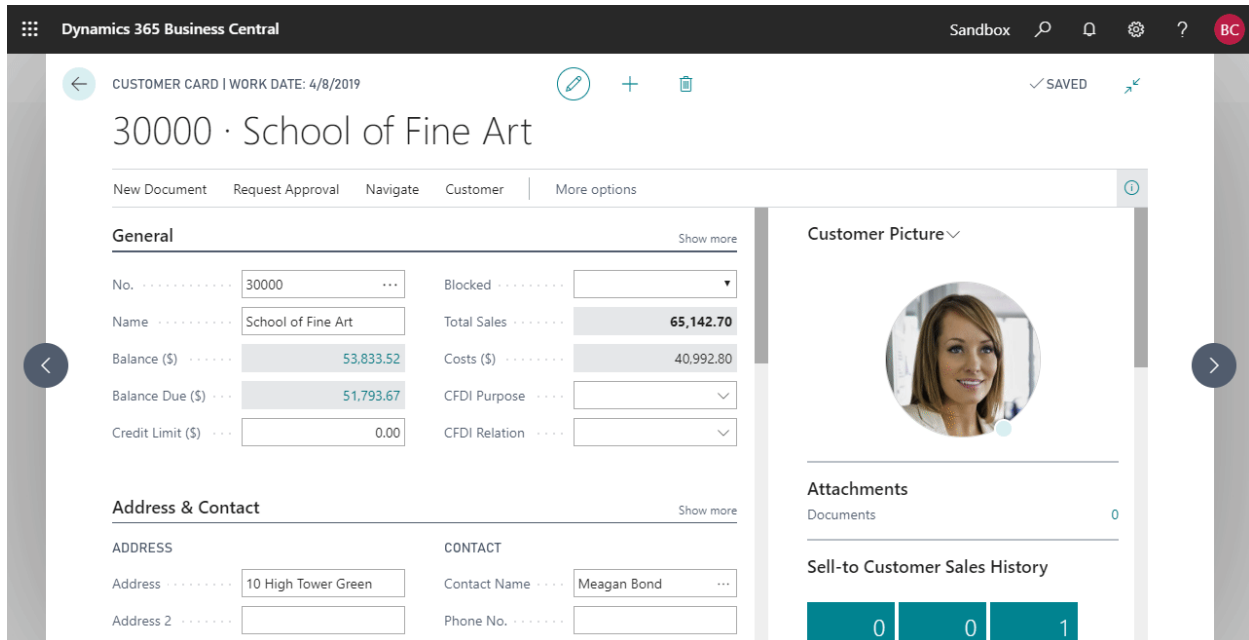
Important points to note

- Every time you start designing, you are effectively creating a new extension. Your changes are immediately visible to other users.
- The changes you make in Designer will apply to **all** users.
- You cannot remove specific fields that are bound to a page and a field must belong to an underlying table.
- You can only add fields, columns, or tiles to its applicable view from list, tall tiles, and wide tiles views. Some components cannot be moved using drag-and-drop and are restricted to the view that they are in.
- You can only add fields/columns, from a predefined list, which is based on the source table. You cannot create new ones.
- In the client, users can change the many of these settings for their workspace only by using personalization (see [Personalizing Your Workspace](#)).

Start and stop designing

In the Business Central client, you start Designer by choosing  **Designer** in the top right corner of any page that you want to make modifications to, and start designing using drag-and-drop components. In Visual Studio Code, you can start Designer by using the **F6** shortcut, which launches a browser that opens the Business Central client in Designer.

After you are done with the adjustments, finish up your design by choosing **Stop Designing**, which allows you to name the extension with an option to download code, and save the extension for the tenant. If you choose not to download the code at the end, you can still pull the code using the **F7** shortcut. You can also uninstall the extension from the **Extension Management** page or even download the source from there.





The screenshot shows the Dynamics 365 Business Central Designer interface. The top bar includes the 'Dynamics 365 Business Central' logo, a 'Sandbox' indicator, and various system icons. The main content area displays a 'CUSTOMER CARD | WORK DATE: 4/8/2019' for '30000 · School of Fine Art'. The card is divided into sections: 'General' (containing fields for No., Name, Balance, Balance Due, Credit Limit, Blocked, Total Sales, Costs, CFDI Purpose, and CFDI Relation), 'Address & Contact' (containing fields for Address, Address 2, Contact Name, and Phone No.), 'Customer Picture' (showing a profile picture of a woman), 'Attachments' (showing 0 documents), and 'Sell-to Customer Sales History' (showing 0 sales). The interface includes navigation arrows on the left and right sides.

Drag-and-drop components

In Designer, you design and modify the current page; you can add existing table fields, move fields around, remove fields from the page, hide and move actions, and more. You can make changes to display the information by using drag-and-drop components.





Working with fields

To add a field or column to a page, in the banner, choose **More**, and then choose **Field**. A pane to the right appears that lets you add fields. Here you can see all of the table fields that are available for the specific page. The table fields displayed are based on the underlying table or tables. The field can have a status of **Placed**, which means that the field already exists on the page. A status of **Ready** means that the field does not already exist on the page. To add a field, drag and drop it to the desired location.

If you want to remove a field or column, select the arrowhead indicator  or  on the component, and then choose **Remove**.

You can edit the caption of a FastTab for a group of fields by selecting the caption and start writing. Simple, clear, and plain.

Setting the freeze pane

Set freeze pane and clear freeze pane locks one or more columns to the left, even when you scroll horizontally. You can set the freeze pane, by selecting the arrowhead indicator  or  of the column that you want as the last column of the freeze pane, and then choose **Set Freeze Pane**. If you want to set the freeze pane back to its original designed location, select the arrowhead indicator  or  for the current freeze pane

column, and then choose **Clear Freeze Pane**.

Setting the Importance on Field

Fields on non-list type pages, such as card and document type pages, include Designer options for setting the importance. The following table describes the options for setting the importance in Designer and how it corresponds to the [Importance property](#) in the page code.

OPTION	DESCRIPTION	IMPORTANCE PROPERTY VALUE
Show under "Show more"	Sets the field so that appears only when the user selects Show more .	Additional
Show always	Sets the field to always display on the page (regardless of whether the user selects Show more or Show less) but not in the FastTab heading if it is collapsed.	Standard
Show when collapsed	Sets the field to always display on the page (regardless of whether the user selects Show more or Show less) and also in the header of the FastTab when the FastTab is collapsed.	Promoted

Setting the Quick Entry on Fields

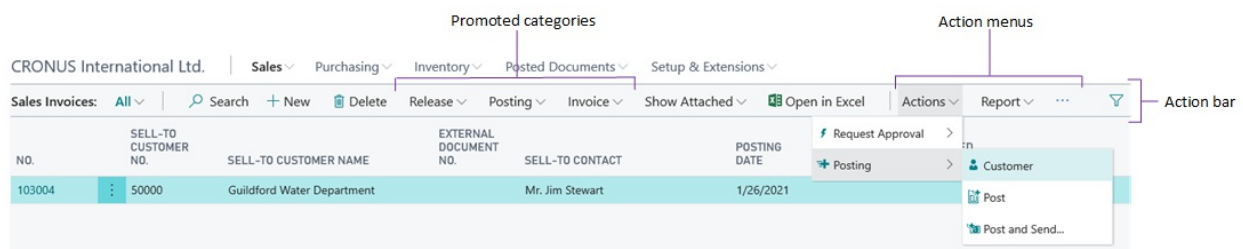
You can use Designer to set the [QuickEntry property](#) on a field. The **QuickEntry** property determines whether the field is given input focus or skipped when users navigate through fields on a page by pressing the ENTER (return) key. You use Quick Entry to help accelerate keyboard data entry by focusing only those fields a user typically needs to fill-in.

To set the QuickEntry property from Designer, select the field or column heading, and then choose either **Include in Quick Entry** (sets the **QuickEntry** property to `true`) or **Exclude from Quick Entry** (sets the **QuickEntry** property to `false`).

For more information about Quick Entry, from a user perspective, see [Accelerating Data Entry Using Quick Entry](#) in the Business Central Application Help.

Working with Actions

Designer lets you make adjustments to the actions that are defined in the action bar of a page. You can move, remove, hide, and show individual actions or action groups.



Remove, hide, and show actions and groups

Actions and actions groups that are already hidden appear dimmed. To change the state of an action or action group, select it, and then choose one of the following options:

OPTION	WHAT IT DOES
Remove	<p>This option is available for the actions that are shown in a only a promoted category alone or actions that are shown in both a promoted category and another action menu.</p> <p>Choosing Remove deletes the action from the selected location so that it no longer appears.</p> <p>If the action is only shown in the promoted category, it will automatically be shown in the action menu where it is originally defined.</p>
Hide	<p>This option is available for actions or action groups that currently are shown only in an action menu (not in a promoted category). Like Remove, choosing Hide will make the action or action group disappear from the action bar in the client. However, in Designer, the action or action group appears dimmed.</p>
Show	<p>This option appears if the action or action group has been previously hidden (dimmed). Choosing this option will make the action or action group appear in the action bar.</p>

Move actions and action groups

Designer lets you move actions within the action bar. For example, you can move an action from an action menu to a promoted category or from one promoted category to another, move an action within an action group or to a different action group.

To move an action or action group, drag and drop it to the desired location, just like with fields and columns.

- You can move individual actions into the promoted categories, but you cannot change the order of the actions in the category.
- You cannot move an action group into a promoted category.
- To move an action or action group into an empty action group, drag the action or action group to the target group and drop it in the **Drop an action here** box.

Preview design on different display targets

The display type icons let you preview the changes you made on desktop, tablet, and phone clients. This way you can make sure that your design will work on the intended display target(s). You can flip to display tablet and phone designs in portrait and landscape orientation.

Controlling User Access to Designer

Accessing Designer is controlled on a user or user group basis by the **D365 EXTENSION MGT** permission set. If a user is assigned this permission set, then Designer is available for the user in the client. To prohibit a user from using Designer, just remove the user from the **D365 EXTENSION MGT** permission set.

See Also

[Developing Extensions](#)

[Getting Started with AL](#)

[AL Development Environment](#)

Keyboard Shortcuts

3/31/2019 • 2 minutes to read

The following table provides an overview of some of the shortcut key combinations that you can use when you are working in Visual Studio Code. For a complete overview, see [Key Bindings for Visual Studio Code](#).

General in Visual Studio Code

KEYBOARD SHORTCUT	ACTION
Ctrl+Shift+P	Show All Commands
F7	Download source code
Alt+A Alt+L	Go! Generates a HelloWorld project
Ctrl+Shift+B	Package
F5	Publish
Ctrl+F5	Publish without debugging
F6	Publish and open the designer
Ctrl+F2	Update the compiler used by the service tier(s)

Editing in Visual Studio Code

KEYBOARD SHORTCUT	ACTION
Ctrl+Space	Look up suggestions for the current object
Ctrl+X	Cut
Ctrl+C	Copy
Ctrl+V	Paste
Ctrl+F2	Select all occurrences
F12	Go to definition
Alt+F12	Peek definition
Shift+F12	Show References
Ctrl+Shift+Space	Look up parameter hints

KEYBOARD SHORTCUT	ACTION
Ctrl+K Ctrl+C	Add line comment
Ctrl+K Ctrl+U	Remove line comment
Ctrl+Shift+P	Show All Commands

Errors in Visual Studio Code

KEYBOARD SHORTCUT	ACTION
F8	Move to the next error or warning
Shift+F8	Move to the previous error or warning

Compile in Visual Studio Code

KEYBOARD SHORTCUT	ACTION
Ctrl+Shift+B	Compile and build the solution
Ctrl+F5	Build and deploy

Debugging in Visual Studio Code

KEYBOARD SHORTCUT	ACTION
F5	Start debugging session

See Also

[Developing Extensions](#)

[Getting Started with AL](#)

[AL Development Environment](#)

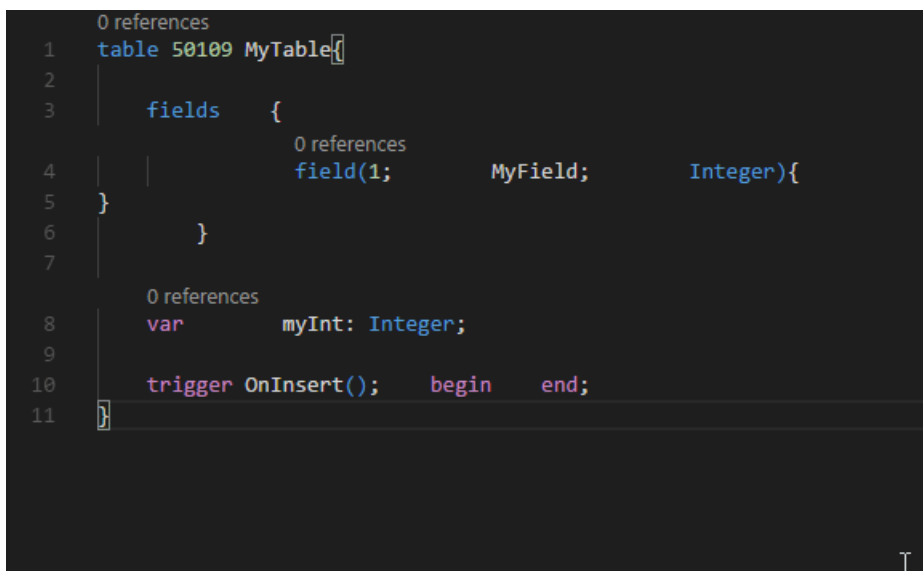
The AL Formatter

3/31/2019 • 2 minutes to read

The AL Language extension offers users the option to automatically format their source code. This capability increases the usability of the editor by allowing developers to instantly fix the indentation and formatting of their code. The auto-formatter analyzes the syntax tree of the AL code that you are formatting and, using rules developed based on the coding and style guidelines for AL, inserts and removes whitespace from key points in the document to make it more readable. The rules used by the auto-formatter cannot be configured by the user. This limitation is present to allow for a uniform style to be used throughout the community of AL developers.

Invoking the AL formatter

The auto-formatter can be invoked to format an entire AL document or a pre-selected range. In an existing project, open the document that you want to format, right-click inside the document, and select **Format Document**. In the default configuration for Visual Studio Code, the command can be run using the shortcut Alt+Shift+F.

A screenshot of the Visual Studio Code editor with a dark theme. It displays an AL code file with line numbers 1 through 11 on the left. The code is as follows:
1: `table 50109 MyTable{`
2:
3: `fields {`
4: `field(1; MyField; Integer){`
5: `}`
6: `}`
7:
8: `var myInt: Integer;`
9:
10: `trigger OnInsert(); begin end;`
11: `}`
The code is partially formatted, with some indentation and spacing. A cursor is visible at the end of line 11. The background is dark, and the text is light-colored.

To format a range, in an already opened project, open the document that you want to modify, select the specific range to format, right-click, and select **Format Selection**. In the default configuration for Visual Studio Code, the command can be run using the shortcut Ctrl+K, Ctrl+F.

```
0 references
1  table 50109 MyTable{
2
3      fields    {
4          0 references
5          field(1;    MyField;    Integer){
6      }
7
8      0 references
9      var        myInt: Integer;
10
11      trigger OnInsert();    begin    end;
12  }
```

See Also

[AL Development Environment](#)

[AL Outline View](#)

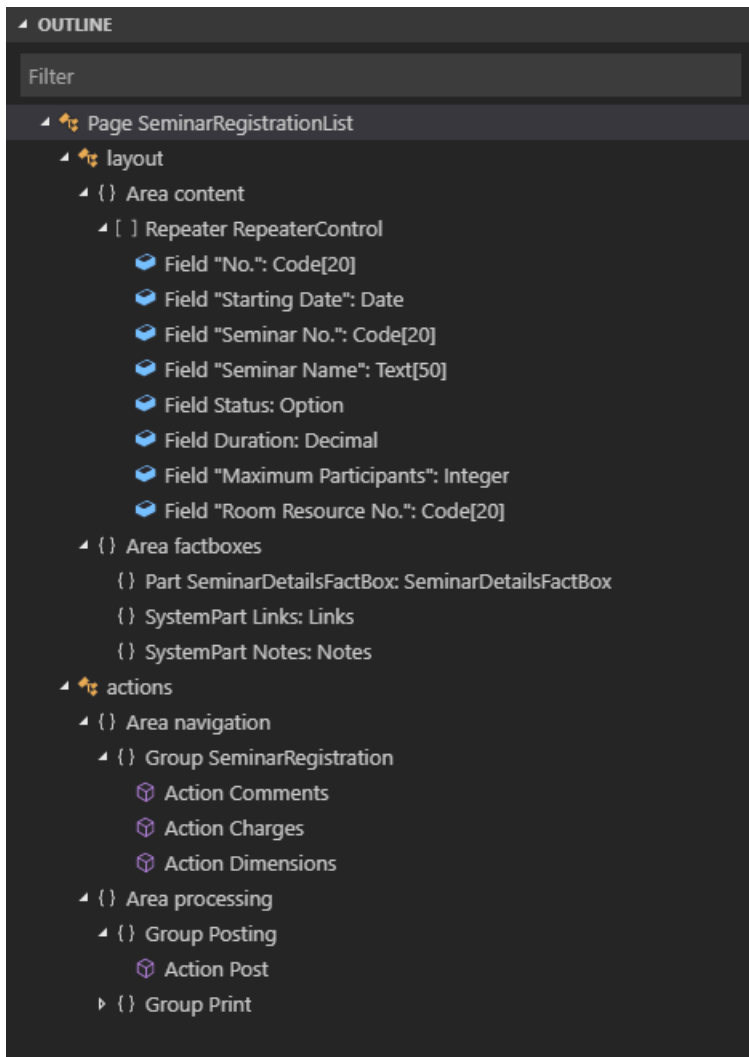
[AL Code Actions](#)

AL Outline View

3/31/2019 • 2 minutes to read

Working with the AL Language extension you have access to the **Outline** view. The **Outline** view is a separate section in the lower left corner, right under the **Explorer** view.

The **Outline** view is enabled by default and shows the symbol tree of the currently active cursor, it also allows you to filter as you type. Double-clicking on any node makes your cursor jump to the selected definition or keyword. The **Outline** view will also display any errors in your project for easy inspection.



You manage the look and feel of the **Outline** view by defining a number of settings, that are all enabled by default:

- `outline.icons` - Outline elements displayed with icons.
- `outline.problems.enabled` - Show errors and warnings on outline elements.
- `outline.problems.badges` - Badges displayed for errors and warnings.
- `outline.problems.colors` - Colors used for errors and warnings.

See Also

[AL Development Environment](#)

[AL Formatter](#)

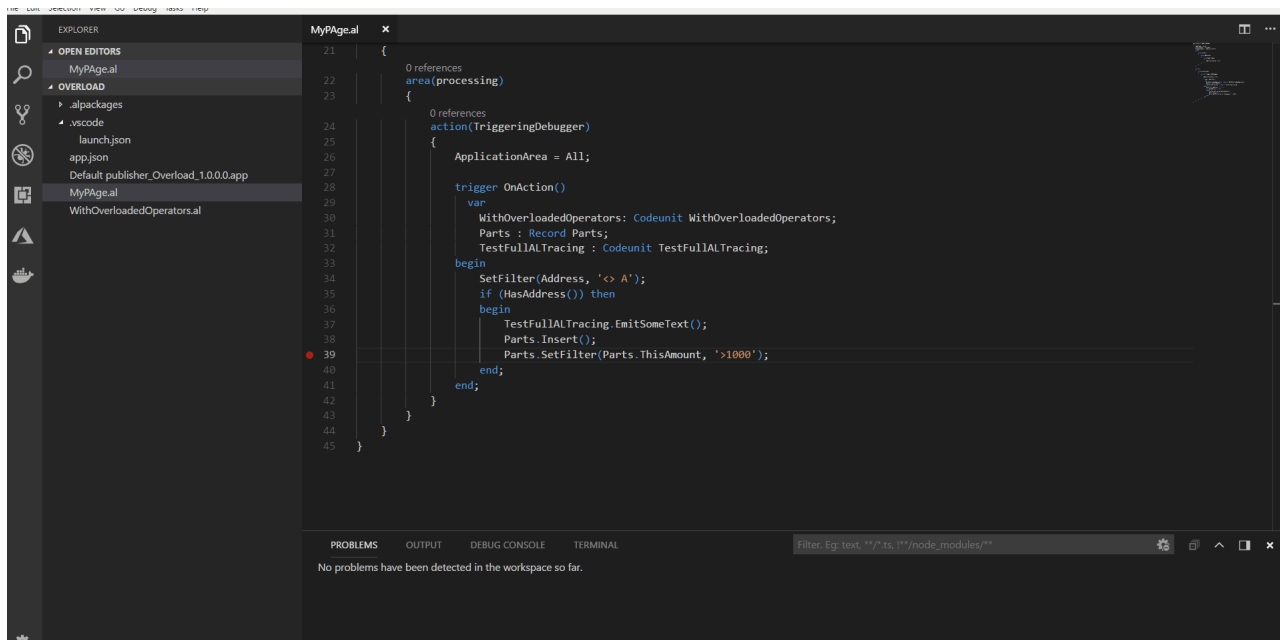
AL Code Navigation

5/24/2019 • 2 minutes to read

When you develop an AL extension, you may want to navigate around the source code frequently. To jump around the code or to access the reference code, you use the Go To Definition feature in Visual Studio Code.

Go To Definition

The Go To Definition feature navigates to the source of a type and opens the result in a new tab. You can use the `F12` shortcut key or right-click and select the Go To Definition feature from the right-click menu. The Go To Definition opens the source in the `.dal` format which contains the base application code. For example, the base application code may contain table metadata and application methods. In the following illustration, the Address type and the HasAddress type opens the `Customer.dal` file and locates the reference code of those types by using the Go To Definition feature.



With Go To Definition, you can step into the referenced code and set breakpoints on the external code and base application code. For more information, see [Debugging in AL](#).

You can always use Go To Definition on Dynamics 365 Business Central code. However, if you want to use it on other extensions, the extension package which is now referenced, when originally published, must have the `showMyCode` property set to `true`. For example, if A is referencing B you can only use the Go To Definition on types of B, if B, when it was published, had the `showMyCode` flag set to `true`. For more information, see [Security Setting and IP Protection](#).

For more information about code navigation in Visual Studio Code, see [Code Navigation](#).

See Also

[Developing Extensions](#)

[JSON Files](#)

[Debugging in AL](#)

[AL Code Actions](#)

AL Code Actions

3/31/2019 • 2 minutes to read

The AL Language extension offers users the option to help fix issues in code. Code Actions is a Visual Studio Code feature providing the user with possible corrective actions right next to an error or warning. If actions are available, a light bulb appears next to the error or warning. When the user clicks the light bulb (or presses Ctrl+.), a list of available Code Actions is presented.

In AL Language extension two code actions are available in the current version:

- Multiple IF to CASE converting code action.
- Spell check code action.

To enable AL Code Actions

1. Open the Command Palette **Ctrl+Shift+P** and choose either **User Settings** or **Workspace Settings** depending on which scope you want the code actions to apply to.
2. Enter the setting `al.enableCodeActions` to the settings file and set it to `true`: `"al.enableCodeActions": true`
3. Save the settings file. You have now enabled code actions on your project.

See Also

[AL Development Environment](#)

[AL Outline View](#)

[AL Formatter](#)

Object Ranges in Dynamics 365 Business Central

4/10/2019 • 2 minutes to read

In Dynamics 365 Business Central running in the cloud, there are three different object ranges in terms of licensing. Developing for Dynamics 365 Business Central is done using Visual Studio Code with the AL Language extension. All tenants in Dynamics 365 Business Central are able to freely use objects in the following ranges:

- 50.000-99.999
- 1.000.000-69.999.999
- 70.000.000-74.999.999

In the following each individual range is explained.

0-49.999

This range is assigned to Dynamics 365 Business Central base app functionality and must not be used.

50.000-99.999

This range is for customizations. A partner can develop an extension tailored to the individual tenant to fit the needs. The partner will develop this either by using a sandbox tenant or by obtaining a Docker image of the current release of Dynamics 365 Business Central that matches the version of the tenant. Once the development is done, the extension can be deployed to the individual tenant.

100.000-999.999

The objects in this range are mainly designed when the Microsoft team localizes Dynamics 365 Business Central for a specific country or region. These objects cannot be used by partners.

1.000.000-69.999.999

This is the Registered Solution Program (RSP) range which partners that have an ISV solution for on-premise have access to. The partner can choose to use this range for developing extensions that can be used either in Dynamics NAV on-premise or in Dynamics 365 Business Central in the cloud. When used in Dynamics 365 Business Central these extensions are obtained as apps from apps.source.microsoft.com.

70.000.000-74.999.999

Partners can obtain ranges for extension development that runs in Dynamics 365 Business Central in the cloud. This range is only available for extension development and only in Dynamics 365 Business Central. These extensions are obtained as apps from apps.source.microsoft.com.

For more information, please see the [Ready To Go](#) program.

See Also

[Ready To Go](#)

[Blog Post](#)

Adding Help Links from Pages, Reports, and XMLports

4/29/2019 • 2 minutes to read

When creating new pages, you can specify which Help file to open if the user selects the *Learn more* links in the UI of Business Central.

The context-sensitive Help link is generated based on a configuration setting in the `app.json` file and the name of the relevant Help file that you specify as part of the metadata for the page object. For more information, see [Configure Context-Sensitive Help](#).

Examples

The following examples show how you can specify the *ContextSensitiveHelpPage* property from new pages, reports, and XMLports:

```
page 50100 MyPageWithHelp
{
    ContextSensitiveHelpPage = 'sales-rewards';
}
```

```
report 50100 MyReportWithHelp
{
    requestpage
    {
        ContextSensitiveHelpPage = 'sales-rewards';
    }
}
```

```
xmlport 50100 XmlPortWithHelp
{
    requestpage
    {
        ContextSensitiveHelpPage = 'sales-rewards';
    }
}
```

In all three examples, the *ContextSensitiveHelpPage* property is set to point at the same Help files. This is because all three example objects support the same feature that is explained in the *sales-rewards* Help topic. In your app, you can choose to structure the Help differently.

See Also

[Configure Context-Sensitive Help](#)

[Translating Base App Help](#)

[JSON Files](#)

[Page Object](#)

[Report Object](#)

[XMLport Object](#)

[Table Object](#)

Working with Translation Files

5/21/2019 • 3 minutes to read

Dynamics 365 Business Central is multilanguage enabled, which means that you can display the user interface (UI) in different languages. To add a new language to the extension you have built, you must first enable the generation of XLIFF files. The XLIFF file extension is .xlf. The generated XLIFF file contains the strings that are specified in properties such as **Caption** and **Tooltip**.

NOTE

To submit an app to AppSource, you must use .xlf translation files.

IMPORTANT

You can use the .xlf translation files approach only for objects from your extension. For translating the base application you still need to use the .txt files approach. For more information, see the **Translation and Localization apps** section below.

Translation and Localization apps

The .xlf files approach cannot be used for translating the base application. If you are working on a translation or localization app (for example for a [country/region localization](#)), you must take the .txt file containing the base application translation, and place the file in the root folder of your extension. When the extension is compiled, the .txt file is then packaged with the extension.

We recommend that you use only one .txt file per language. There is no enforced naming on the .txt files, but a suggested good practice is to name it `<extensionname>.<language>.txt`.

For more information about importing and exporting .txt files, see [How to: Add Translated Strings By Importing and Exporting Multilanguage Files in Dynamics NAV](#).

Generating the XLIFF file

To enable generation of the translation file, you must add a setting in the manifest. In the app.json file of your extension, add the following line:

```
"features": [ "TranslationFile" ]
```

Now, when you run the build command (Ctrl+Shift+B) in Visual Studio Code, a `\Translations` folder will be generated and populated with the .xlf file that contains all the labels, label properties, and report labels that you are using in the extension. The generated .xlf file can now be translated.

IMPORTANT

Make sure to rename the translated file to avoid that the file is overwritten next time the extension is built.

Label syntax

The label syntax is shown in the example below for the **Caption** property:

```
Caption = 'Developer translation for %1', Comment = '%1 is extension name', locked = false, MaxLength=999;
```

NOTE

The `comment`, `locked`, and `maxLength` attributes are optional and the order is not enforced. For more information, see [Label Data Type](#).

Use the same syntax for report labels:

```
labels
{
  LabelName='LabelText',Comment='Foo',MaxLength=999,Locked=true;
}
```

And the following is the syntax for **Label** data types:

```
var
a:Label'LabelText',Comment=' Foo',MaxLength=999,Locked=true;
```

The **ML** versions of properties are **not** included in the .xlf file:

- [CaptionML](#)
- [ConstValueML](#)
- [InstructionalTextML](#)
- [OptionCaptionML](#)
- [PromotedActionCategoriesML](#)
- [ReqFilterHeadingML](#)
- [RequestFilterHeadingML](#)
- [ToolTipML](#)

The [TextConst Data Type](#) is not included in the .xlf file either.

The XLIFF file

In the generated .xlf file, you can see a `<source>` element for each label. For the translation, you will now have to add the `<target-language>` and a `<target>` element per label. This is illustrated in the example below.

```
<ding="utf-8"?>
<xliff version="1.2" xmlns="urn:oasis:names:tc:xliff:document:1.2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="urn:oasis:names:tc:xliff:document:1.2 xliff-core-1.2-transitional.xsd">
  <file datatype="xml" source-language="en-US" target-language="da-DK" original="ALProject16">
    <body>
      <group id="body">
        <trans-unit id="PageExtension 1255613137 - Property 2879900210" max-width="999" size-unit="char"
translate="yes" xml:space="preserve">
          <source>Developer translation for %1</source>
          <target>Udvikleroversættelse for %1</target>
          <note from="Developer" annotates="general" priority="2">%1 is extension name</note>
          <note from="Xliff Generator" annotates="general" priority="3">PageExtension - PageExtension</note>
        </trans-unit>
      </group>
    </body>
  </file>
</xliff>
```

NOTE

You can have only one .xlf file per language. If you translate your extension to multiple languages, you must have a translation file per language. There is no enforced naming on the file, but a suggested good practice is to name it

```
<extensionname>.<language>.xlf
```

When the extension is built and published, you change the language of Dynamics 365 Business Central to view the UI in the translated language.

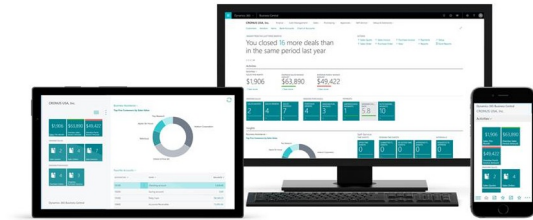
See Also

[How to: Add Translated Strings By Importing and Exporting Multilanguage Files in Dynamics NAV](#)

The "Ready to Go" Program

6/25/2019 • 3 minutes to read

Are you 'Ready to Go' for
Microsoft Dynamics 365 Business Central?



Dynamics 365 Business Central is a business management solution that helps companies connect their financials, sales, service, and operations to streamline business processes, improve customer interactions, and make better decisions.

Business Central creates multiple [opportunities for partners to provide apps or consulting services](#) on Microsoft AppSource.

The "Ready to Go" program is designed to support you in the journey of bringing offerings to market. The program contains learning, coaching, and tooling. Use the tabs below to read more about the elements of the "Ready to Go" program.

TIP

Keep on top of news, tips, tools, programs, and new capabilities by following us on the [Business Central for partners](#) blog.

- [Learning](#)
- [Coaching](#)
- [Tooling](#)
- [Resources](#)

The "Ready to Go" learning catalog is an extensive collection of materials for Dynamics 365 Business Central, including training resources, white papers, and tools for both app developers (ISVs) as well as resellers (VARs). It is designed for marketeers, business decision makers, sales and pre-sales roles, architects, consultants, and developers. [Access the "Ready to Go" learning catalog.](#)

The “Ready to Go” Online Learning Catalog

This “Ready to Go” learning catalog provides Dynamics 365 Business Central partners a single point of access to training resources. It contains readiness information for multiple roles inside organizations which resell Dynamics 365 Business Central and for those who develop apps for Microsoft AppSource. The information in this document is being updated on a weekly basis.



Materials for developers, architects, and engineers

The “Ready to Go” online learning for developers, architects, and engineers.



Materials for application consultants

The “Ready to Go” online learning for application consultants.



Materials for pre-sales roles

The “Ready to Go” online learning for pre-sales.



Materials for sales roles

The “Ready to Go” online learning for sales.



Materials for marketeers

The “Ready to Go” online learning for marketeers.



Materials for business decision makers

The “Ready to Go” online learning for business decision makers.

Getting You Started with Your Add-On App

6/17/2019 • 11 minutes to read

Build your business on Dynamics 365 Business Central

Microsoft Dynamics 365 Business Central is a business management solution that helps companies connect their financials, sales, service and operations to streamline business processes, improve customer interactions and make better decisions. With this modern business platform, you can easily and quickly tailor, extend and build applications so they fit your specific needs — with little to no code development.

AppSource is Microsoft's marketplace for your Dynamics 365 Business Central offerings and there are several reasons why going to market with Microsoft AppSource is a great idea. For example, it allows you to promote your brand, expand your reach, accelerate the customer journey and upsell your solutions and it connects you with millions of Office 365 & Dynamics 365 business users. Find more information about which opportunities you have as a partner at: aka.ms/BusinessCentralApps

You can bring two types of offerings to Microsoft AppSource:

- **Add-on Apps** (that brings your industry expertise to market), **Connect Apps** (that connect services) and **Embed Apps**.
- Or **Packaged Consulting Services** (that bring ready-made packaged engagements to market).

Guidelines for Business Central Add-on apps

To ease your journey, from the initial listing to the final publication of your **Add-on app** on AppSource, we have created two whitepapers that outlines 4 consecutive steps that you need to go through. To bring your Business Central offers to AppSource smoothly, we recommend that you check off each step as you progress. We highly recommend that you lean on the guidelines in these whitepapers to support you throughout the process of bringing your app to AppSource:

Getting you started with Add-on Apps

- STEP 1: Create and set up your accounts
- STEP 2: List your app on AppSource

Developing and publishing your Add-on App to AppSource

- STEP 3: Develop your app
- STEP 4: Initiate the validation and publication process

IMPORTANT

Please review all of the steps and follow the [Marketing Validation Checklist](#) and [Technical Validation Checklist](#).

STEP 1: Create and set up your accounts

MPN ID

All App builders and app publishers must be identifiable to Microsoft. For this reason, you need to become a member of Microsoft's Partner Network (MPN) – this will of course be off no cost to you.

- [Sign up to become a member of our Partner network here:](#)

PartnerSource Business Center (PSBC) account and your unique license file

Register as a Partner

Developing an Add-on app requires you to be known as Dynamics 365 Business Central developer and requires you to have a unique development license file with a specific object range.

To obtain an object range for developing a Microsoft Dynamics 365 Business Central, you must first have access to **PartnerSource Business Center (PSBC)**

Access to PSBC is provided by having an active:

- Solution Provider Agreement (SPA) if you are a reselling partner using the Dynamics Pricelist
- Partner Registration Agreement (PRA) if you are a non-selling partner.

The relevant contract can be requested through your local Regional Operations Center (ROC) Contracts and Agreements Team below:

- mbscon@microsoft.com : If you are based in Europe, the Middle East, or Africa.
- mbsagree@microsoft.com : If you are based in the Americas.
- mbslques@microsoft.com : If you are based in the Asia Pacific region.

Object Ranges

When you develop a Microsoft Dynamics 365 Business Central App for Microsoft AppSource, you will need to request access to an object range which holds a certain number of objects with which you can build your solution. In order to avoid overlap between objects used in different solution, each partner is assigned a number of objects in a unique object range. For example, a partner could get assigned the object range 70,001,000 – 70,001,999 which will give them 1000 numbered objects which they can use to develop Microsoft Dynamics 365 Business Central solutions.

Requesting the correct license file and object range

Depending on where you will deploy your Dynamics 365 Business Central solution (cf. on premise or in the Cloud) you can use different licensing methods and object ranges. There are currently 2 available ranges which you can request. Both have some characteristics which you need to keep in mind:

- The **RSP Object Range (1-69 million)**: This object range is tied to [the RSP Program details](#). The program details specify that you have to pay quarterly for used objects. However, if you comply with the [Certified for Microsoft Dynamics \(CFMD\)](#) program requirements, one of the benefits of the program is that the quarterly fees on object costs will be waived. This object range can both be implemented on-premises, partner hosted (in C/AL or AL format) or in the Business Central SAAS Service (AL Only format). The RSP Program page describes the process on how to request the RSP object range.
- The **App Object Range (70-75 million)**: This object range was originally designed to run in the Business Central service only for Microsoft Appsource Apps.
Today you can implement apps developed in this range both on-premises, partner hosted and in the Business Central SAAS Service. This object range is free of charge, the only requirement on this range is that your objects can be AL only.
You can request both object ranges [here](#).

You will have to make a choice which object range is best for you. Some partners desire to have 1 product line which can be implemented everywhere, others want to build a new SAASified app separate from their legacy solutions.

Developer account

A developer account enables you to submit apps and add-ins to Microsoft's marketplaces, including the Windows

Store, Office Store, Azure Marketplace, and Microsoft AppSource. Note, you only need one developer account per company (not one per app submission).

- [Register \(or check if you already have access\) here:](#)
- A one-time registration fee applies

Choosing a primary contact email and publisher display name

When registering for a Microsoft Developer Account you will be asked to provide an "E-mail address" and a "Publisher display name". When choosing your display name and primary email account, please take the following into account:

- Email: To ease submission and avoid missing vital communications we recommend you provide a companywide email/dev center account that can be shared across multiple users so that several people can manage your portal submission.

Although, if you prefer a singular account, where you can add multiple users through the portal, then this is possible too.

Publisher Display name: A display name refers to how you want your company name displayed on your app in **Microsoft AppSource**.

See example to the right, where the display name is highlighted in yellow.

Consistency is key!

It is key that you use the same display name and email throughout your app submission, as you will be asked to provide them in several different touch points. such as for example:

- To sign into the Cloud Partner Portal (and creating your app offer)
- To enroll and sign in to the Collaborate tool and "Ready to Go" platform

Access to the Cloud Partner Portal and being set up as a publisher

The Cloud Partner Portal is the place where you first will submit your App for the marketing validation and later submit it for publication.

In order to be set up as a publisher and access the Cloud Partner Portal, you need to email Ryan Weigel at rweigel@microsoft.com , and provide him the following information:

- Your Publisher display name (which will be displayed on AppSource)
- The Email account you chose as the primary contact (which is needed when logging into the Cloud Partner Portal)
- Your MPN ID

Get onboarded to tools and programs

Create your sandbox environment

You have 3 options to work develop against the current version of Microsoft Dynamics 365 Business Central.

- [If your add-on is lightweight, it might be sufficient to use a sandbox environment](#)
- [If you want to setup a Docker-based development environment on Azure, you can use:](#)
- [If you want to setup a local Docker-based developer environment, you can download a PowerShell script available at](#)

The "Ready to Go" program

There are several things to keep in mind in building an Add-on app. The **"Ready to Go" program** is designed to support you in bringing your Microsoft Dynamics 365 Business Central offers into Microsoft Appsource. The program encompasses the following three core support options that you can leverage:

- Element 1: ["Ready to Go" Online learning](#)
- Element 2: ["Ready to Go" Coaching](#)
- Element 3: ["Ready to Go" Platform](#)

If you want to have more in depth learning resources to get up to speed, then you can get a sneak-peek of the extensive set of **"Ready to Go" resources available in the online learning catalog**. We highly recommend that you either consume the materials which are built for you in the "Ready to Go" online learning catalog or get coached by one of our ISV Development Centers.

Learn more about how you can leverage the "Ready to Go" program's different support options here: aka.ms/ReadyToGo.

Register on Collaborate

Prerequisites you need to have to register on Microsoft Collaborate

- Azure Active Directory (AAD)

Note: If you have Office 365 then your company should have AAD

- AAD Global Administrator permission

To find out if your company has an AAD account, please check with your Network Administration team for your company.

How your Global Administrator must register for Collaborate

Not all people from your company can initiate the onboarding into Collaborate – only your company Global Administrator has the permission to do so.

To start the registration process, your company Global Administrator must first go to the following link:

<https://aka.ms/Collaborate>

Next, your company Global Administrator must click on the 'Get Started' option under the 'Microsoft Collaborate' header.

- Note: If you are properly registered, and have setup your above Developer Account, the registration page should autofill with your Company details. If the page is not filled automatically, please complete the form manually.

Once completed, be sure and click the 'Terms of Use' (TOU) checkbox at the bottom of the registration page. Note: You need to accept the TOU to successfully register (cf. image below).

Click the **Next** button to complete your initial registration. When successful, you will see the image below.

As the final step, to complete registration, click 'GO TO DASHBOARD'.

How to use the dashboard to add your coworkers to Microsoft Collaborate If you DO NOT want to add any coworkers:

- Please skip this part and move on to the next section called "Getting access to the available builds and engagements" to download packages through Microsoft Collaborate.

If you DO want to add coworkers, please follow the 5 steps below:

1. Log on to Microsoft Collaborate with your Global Administrator account on <https://aka.ms/Collaborate>.

2. Click on the 'Gear' Icon on the top right corner of the page and then on 'Account Settings' as shown on the image below.
3. Click on 'Users' under 'Settings' in the grey panel on the left-hand side of the page. The following image will appear.
4. Click the grey 'ADD USERS' button and leave the default choice to 'Add existing users' as-is. Now you can search for the user(s) that you want to add to Collaborate. To add them you need to select them from the menu, and then click the grey 'ADD SELECTED' button (see image below).
5. You have now successfully added your coworker to Collaborate. The added users will appear in your list of users and will now be able to log on to Microsoft Collaborate using the following link:
<https://aka.ms/Collaborate>

Getting access to the available builds and engagements

1. Register on [Microsoft Collaborate](#) by using your AAD Global Admin account (as described in detail above).
2. Once you have successfully registered (and added coworkers) on Microsoft Collaborate, your company's Azure Active Directory (AAD) global administrator should send us an email to complete the on-boarding [Dynamics 365 Business Central Programs](mailto: Dyn365BEP@microsoft.com). We need to manually assign you to the right programs and engagements. Expect a response from us within 1-2 business days.
3. In this email to [Dynamics 365 Business Central Programs](mailto: Dyn365BEP@microsoft.com), please specify the following:
 - Your 'Publisher Display Name'
 - The name(s) of the people you have added to Microsoft Collaborate
 - The email address(es) of the people you have added to Microsoft Collaborate (No personal email addresses please)
 - The roles you have given them on Collaborate (Participant, Power user)
 - MPN ID# and registered Partner Name

STEP 2: List your Add-on app on Microsoft AppSource

To list your app, you need to register it on Microsoft AppSource.

- List your app here: <https://appsource.microsoft.com/en-us/partners/list-an-app>

When listing your app you need to specify the following:

- Contact Info (First name, Last name, Email)
- Company Info (Company name, Company website, MPN Id)
- App Info: Type of offer you intend to publish (cf. app or consulting service)
- Type of app you intend to publish (cf. Dynamics 365 Business Central Add-in)
- App name

App description: Intended users of the app (cf. IT professional, Developers, Business Users)

Upon listing your app on AppSource you will receive an email from us that outlines the next steps you need to take along with a list of useful resources that can help you bring your app to AppSource quickly.

Next steps

Now that you have completed step 1 and 2 (cf. setting up your accounts and listing your offer on AppSource) you can now proceed to step 3 and 4 (cf. Developing your offer initiating the validation and publication process). Please review all of the steps and follow the [Marketing Validation Checklist](#) and [Technical Validation Checklist](#).

Useful resources

Guidelines and general information

Find general information on Add-on apps for Business Central here: aka.ms/AppSourceGo.

Monthly “Ready to Go” Office Hours call

“Ready to Go” Office Hours is a monthly call that takes place the second Tuesday of every month. The call is structured as an FAQ session, where a team of our different experts will be present to answer any technical or marketing related questions that you may have in relation to bringing your app into AppSource. Sign up for the individual calls that you want to participate in here: aka.ms/ReadyToGoOfficeHours.

GitHub

Use the GitHub forum to ask, or search, the community and Microsoft experts for questions respectively. Go to: <https://github.com/microsoft/al/issues> now and start asking away.

Follow “Ready to Go” engagements on social media

Get insights on what’s happening with Business Central – follow us on Twitter and LinkedIn.

If you have any other technical questions in relation to developing your Add-on app, then please email: d365val@microsoft.com.

Marketing Validation Checklist

6/17/2019 • 4 minutes to read

IMPORTANT

In these guidelines, you will find an outline of our requirements for marketing validation as well as examples of best practices, which you can use as inspiration, while developing the storefront details of your offer, your sales landing page and video materials.

To ease your experience with developing the storefront details of your listing, we have numerated the core elements, as they appear when you upload your content in the “Storefront Details” tab on the Cloud Partner Portal.

This image below is an example of what an offering looks like on AppSource, when the storefront details are completed according to best practices. We highly recommend that you review these guidelines. Click on any of the links under Guidance/Examples to gain more insight on that particular section or item within the marketing validation process.

AppSource

Apps

Consulting Services

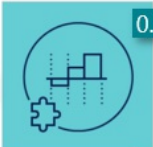
Co-Sell Solutions

List on AppSource

Blog

Apps > Sales and Inventory Forecast

14



0.C

Sales and Inventory Forecast

Microsoft

Overview

Reviews

5

GET IT NOW

SAVE FOR LATER

★★★★★ (4)

11

Products

Dynamics 365 Business Central

Publisher

Microsoft

Acquire Using

Work or school account

9

Version

2.0.22794.0

4

Categories

Operations + supply chain

Sales

3

Industries

Distribution

Professional services

20

Support

Support

Help

6

Legal

License Agreement

19

Privacy Policy

18

1

Use reliable forecasting to help ensure that you always have the items your customers want.

2.A

Do you have the right stock on your shelves? Are stock outs costing you customers? And are your procurement decisions relying on basic spreadsheets?

2.B

Managing inventory is a delicate balancing act. Carry too little and you lose orders (and customers). Carry too much and you tie up much needed working capital. Carry far too much and you end up discounting, or worse, writing off obsolete products.

2.C

Our app uses Cortana Intelligence to analyze historical data to predict future demand, so you can base procurement decisions on accurate and reliable forecasts, and help your company avoid lost revenue, optimize shipping costs, discover trends and boost your brand reputation by always delivering on orders.

2.C

Stop relying on basic spreadsheets that take hours of valuable time to complete. Turn anxiety into proactive control and manage this critical business process in minutes by using Microsoft Sales and Inventory Forecast extension.

2.D

Features and benefits of using this extension

- Free up cash
- Know exactly when to replenish stock
- Always have inventory on hand to satisfy every customer order

2.E

Click the **Get it now** button and start aligning your inventory replenishment with your customer demand. Your customers and sales team will love you for it.

2.F


Supported editions:
This extension supports both the Essential and Premium editions of Microsoft Dynamics 365 Business Central.

2.G

Supported countries:
Austria, Belgium, Canada, Finland, France, Germany, Italy, Netherlands, Spain, Sweden, Switzerland, United Kingdom, and United States

16

Learn more
[Dynamics 365 Business Central capabilities guide](#)



15

17

ITEM NUMBER	VALIDATION REQUIREMENTS	GUIDANCE/EXAMPLES
-------------	-------------------------	-------------------

ITEM NUMBER	VALIDATION REQUIREMENTS	GUIDANCE/EXAMPLES
0.A - 0.C	Your app can be in any language – if not in English, a document with English translation is required. Branding - Remember to use the product branding guidelines properly	General Best Practices including Language and Branding
0.D	Must be the same as the manifest. (Your offer name) for Microsoft Dynamics 365 Business Central	Offer Name
1	Max. 25 words or 100 characters, value proposition	Offer Summary
2 - 2.G	Use simple HTML tags, 3000 characters See link for best practices on formatting the content.	Description
2.F	Supported editions - Premium, Essential or both	Premium SKU Listing
2.G - 7	Mark only the Dynamics 365 Business Central supported countries that apply.	Supported Countries
3-4	Choose the industries the your offer is best aligned to. (max 2) Categories that your offer caters to (max 3)	Industries and Categories
5	Add-on App - free or trial, Connect App - Contact me	Package and App Type
6	Your Help Link cannot be the same as your Support Link. Your Help Link should be a landing page on your website where one can find help resources, such as documentation, FAQs, step-by-step guides, webinars, etc.	Help Link
7 - 8	Listed under Description and Language & Branding	Supported Countries and Languages
9-10-11	Enter the latest version number. Enter date (dd/mm/yyyy)when you expect your app to be released	App Version, App Release Date, Supported Products
12	Although it's not required, it is strongly recommended that you use keywords as to optimize the searchability of your app. Maximum of 3 words.	Keywords
13	Enter a secret key that you'll use to preview your offer on AppSource before going live	Hide Key

ITEM NUMBER	VALIDATION REQUIREMENTS	GUIDANCE/EXAMPLES
14	You are required to provide two logos on AppSource. 48x48 pixel resolution (for your app's search page) and 216x216 resolution (for your app's details page). Both in .png format.	Logos
15	We recommend that you add one or two videos that would act as a demo or a quick 3 minute sales pitch on your app.	Videos
16	Minimum of 1 document (max of 3) permitted.	Documents
17	Please double check the resolution needs to be 1280 x 720 and only in .png format. It's required to have a minimum of 3 but you can have up to 5.	Screenshots
18	Landing page where prospects can find information on how you handle their data.	Privacy Policy Link
19	You need to provide a URL for a distinct page with your license agreement, where prospects can find the information on your terms of use.	License Agreement
20	It is a requirement that you provide different (i.e. more than two) contact options on your app landing page.	Support Link
	Special Rules for Localization, Connect & Consulting Services Apps	
	Add-On Apps with some unique requirements.	Localization Apps
	Add-On Apps with some unique requirements.	Connect Apps
	Add-On Apps with some unique requirements.	Consulting Services

Marketing Checklist for Showcasing Your App

The storefront details on AppSource is the first impression that prospects get regarding your offer. First impressions last, so make sure to invest some time in developing the content on the storefront, so it gives a good impression from the beginning. Failing to do so will jeopardize the hard work you put in, when developing your offer, likely leaving the prospect confused or looking elsewhere. Accordingly, we recommend you put in the time, effort and due diligence when developing this content. Follow this marketing validation checklist and get your app passed on the first submission.

Helpful Videos

[Video - Best Practices for Submitting your App](#)

[Video - Successfully submit an app on AppSource](#)

Why marketing validation is mandatory

The marketing validation is in place to make sure that the customer journey on AppSource is a uniform experience, where customers quickly and easily can get an overview of your offer's functionality, why they can benefit from using it, while also enticing them to learn more and take the necessary actions to start using your offer.

Accordingly, to ensure that your listing establishes a good first impression, we carry out a marketing validation of all Dynamics 365 Business Central apps that are being published on AppSource. This also goes for apps that already are live, if some of the content in their storefront details needs to be edited – in this case they need to be resubmitted for marketing validation as well. Consequently, to be published on AppSource all listings need to pass the required elements within the marketing validation process.

Technical Validation Checklist

3/31/2019 • 2 minutes to read

The following is a checklist of all requirements that you **must meet before submitting** an extension for validation. If you do not meet these mandatory requirements, your extension will fail validation.

REQUIREMENT	EXAMPLE/GUIDANCE
Develop your extension in Visual Studio Code.	Developing AL Language extensions
The app.json file has mandatory settings that you must include. Here you can also read more about dependency syntax and multiple countries per a single app syntax.	Mandatory app.json settings
Coding of <code>Date</code> must follow a specific format (no longer region specific)	Use the format <code>yyyymmdd</code> . For example, <code>20170825D</code> .
Remote services (including all Web services calls) can use either HTTP or HTTPS. However, HTTP calls are only possible by using the HttpRequest AL type.	Guidance on HTTP use
Only JavaScript based Web client add-ins are supported. The zipping process is handled automatically by the compiler. Simply include the new AL <code>controladdin</code> type, JavaScript sources, and build the app.	Control Add-Ins
The .app file must be digitally signed.	Signing an APP Package File
The user scenario document must contain detailed steps for all setup and user validation testing.	User Scenario Documentation
Set the application areas that apply to your controls. Failure to do so will result in the control not appearing in Dynamics 365 Business Central.	Application Area guidance
Permission set(s) must be created by your extension and when marked, should give the user all setup and usage abilities. A user must not be required to have SUPER permissions for setup and usage of your extension.	Exporting Permission Sets Managing Users and Permissions
Before submitting for validation, ensure that you can publish/sync/install/uninstall/reinstall your extension. This must be done in a Dynamics 365 Business Central environment.	How to publish your app
Thoroughly test your extension in a Dynamics 365 Business Central environment.	Testing Your Extension
Do not use <code>OnBeforeCompanyOpen</code> or <code>OnAfterCompanyOpen</code>	Replacement Options
Include the proper upgrade code allowing your app to successfully upgrade from version to version.	Upgrading Extensions

REQUIREMENT	EXAMPLE/GUIDANCE
Pages and code units that are designed to be exposed as Web services must not generate any UI that would cause an exception in the calling code.	Web Services Usage
You must include all translations of countries your extension is supporting. The use of xlf is required.	Translating Your Extension, Countries and Translations Supported.
You are required to prefix or suffix the Name property of your fields. This eliminates collision between apps.	Prefix/Suffix Guidelines
You are required to include a Visual Studio Code test package with your extension. Ensure that you include as much code coverage as you can.	Testing the Advanced Sample Extension
DataClassification is required for fields of all tables/table extensions. Property must be set to other than <code>ToBeClassified</code> .	Classifying Data
You must use the Profile object to add profiles instead of inserting them into the Profiles table.	Profile Object
Use <code>addfirst</code> and <code>addlast</code> for placing your actions on Business Central pages. This eliminates breaking your app due to Business Central core changes.	Placing Actions and Controls

See Also

[Developing AL Language extensions](#)

How to Make Compelling Videos

5/3/2019 • 9 minutes to read

Why use video? It is well worth investing time and resources to create marketing videos for your app, it is taken seriously in a business environment.

Reasons why video is a superior medium

- Videos offers a very rich, stimulating communication medium that engages multiple senses.
- Video engages the mind and triggers emotions, which makes it more compelling than text-based content.
- Our brains have an easier time processing visual stories than bullet points or straight facts.

A recent Demand Gen survey indicated that 58% of B2B buyers consume video content, while Hyperfine media states that 59% of executives would rather watch video than read text. Also, 50% of executives look for more information after seeing a product/service in a video.

Speak to Specific Personas in your videos

You should create a video for each of the three core personas in the company:

- WHY persona: Owner/executive/leadership
- HOW: Business line manager
- WHAT: IT buyer, User

A horizontal generic message that attempts to speak to everyone will likely not reach anyone in an emotionally engaging way. Wasting a prospect's time by requiring him/her to listen to irrelevant data or information will only create frustration and lead him/her to form a negative bias towards your company.

Choose the video format that is relevant for the audience that you want to target

Video type 1: "Why" video

How to set up "Why" videos

- Recommended length: 60-90 seconds
- Purpose:
 - Your video should clearly communicate WHY prospects need to buy your solution now.
- Focus:
 - Make sure the prospect is the hero of the story, not you or your company. Prospects are not interested in hearing about your company at this stage. They are simply trying to determine if what you offer is of value to THEM.
 - Your video should speak to the principal challenges and goals of your core decision-maker persona.
 - Describe the desired end state they will achieve by using your app.
 - A client/customer speaking about the benefits they received from your app is far more credible and compelling than anyone from your organization.
 - Don't only rely on "features" to acquire new customers.

How to speak to a WHY persona in a video

- Target audience:
 - Owner/executive/leadership
 - They have limited time and financial resources as well as many competing priorities and resource requirements
 - You need to elevate the discussion to a strategic level, where you highlight market share, competitiveness, profitability, differentiation, revenue loss, and more.
- Message:
 - The question you must answer beyond a doubt is WHY should they invest the time and money to buy your app? What will they get out of it?
 - Why should they spend money on a new system now? Can't they put it off?
 - The WHY messaging teaches people something and it is industry specific and results oriented, as well as being memorable. It engages the emotional/limbic brain and leads to meaningful action.

Video type 2: "How & What" video

How to set up "How and What product videos"

- Recommended length: Up to 3 minutes.
- Purpose:
 - This video goes into greater depth communicating the main benefits of your app as well as HOW you solve your prospects' problems. You can include some WHAT content.
 - Focus:
 - Demonstrating the proof of your claims is critical during this video.
 - Show very specific dashboards or visually show how you address prospect challenges.
 - If possible, use contrast to create desire and a sense of urgency. For example, you could show a complex, ugly data-filled forecast spreadsheet next to a beautiful visual dashboard stating "your sales forecast before and after."

How to speak to a HOW persona: (Business line manager)

- Target audience: Business Line Manager
 - HOW focuses on the operational benefits your solution will provide and HOW your organization will support the implementation.
 - Speaking to the HOW persona starts to separate you from the pack.
 - Message:
 - HOW content is VISUAL in nature and ACTION oriented. It allows your prospects to identify with you at a FUNCTIONAL business level and to Add-on with you. It provides evidence that your organization has relevant industry experience. Tribal acceptance increases, while risk decreases.
 - HOW messaging begins to appeal to the limbic brain because it is focused primarily on emotional business pains and problems.

How to speak to a WHAT persona: (IT buyer, User)

- Target audience: IT-buyer, User
 - WHAT people are often tasked with finding a solution and are important influencers in the decision, but they are not the financial decision makers, and their opinions are easily overturned by HOW and WHY people in the organization.
 - Therefore, don't invest all of your marketing time, money, and effort into providing content just for them.
- Message:
 - You need to survive the WHAT inquisition and provide information about product-related features, functionality, and data so that prospects clearly understand your solution offering.
 - However, this will seldom trigger an emotional response and, therefore, it is likely there will be little or no emotional engagement with your content.

- WHAT content is binary. WHAT content is a commodity. WHAT content is boring. Logical WHAT content is a necessary evil because many prospects initially go looking for it, but stopping here means remaining relevant only to WHAT personas.

Video type 3: “Getting started” video

How to set up “Getting started videos”

- Recommended length: 2–3 minutes maximum.
- Purpose: This video should prove it is quick and easy to get up and running with your app.
- Target: What personas (Users, It buyers)

Video type 4: “Customer testimony” video

How to set up “Customer testimonial videos” - Recommended length: Up to 2 minutes

- Purpose:
 - Social reinforcement: Customer stories are the best proof of gain.
- Focus:
 - A story coming directly from your client in the form of a testimonial is stronger than having your prospects take your word for it. If prospects see that other similar people or companies have already purchased your solution, then their natural response will be to more readily accept it as a solution for themselves.

Video tips

How to structure your video and practical things to keep in mind when producing videos

How to structure the flow in your video?

- Gain immediate attention in the first 10 seconds of the video Stimulate curiosity by include a hook phrase/comment that will elude to solving a pain point. Ask questions about the prospects’ core business challenges or ask about something they would like to do but can’t accomplish today.
- Highlight the prospects’ problems: Use an empathetic approach when describing their current situation and demonstrate that you understand their current business challenges. They must relate to this if they are to continue watching.
- Give them new learning Teach them something they don’t know. Demonstrate you have expertise and knowledge about their business or industry that they might not. Show you can offer strategic value to them.
- Paint a picture of a desired outcome they would love to have or state they crave to experience Highlight the benefits, rewards, and value they will enjoy after they purchase from you. Include both what it looks like and how it will feel.
- Prove what you’re saying is true Prospects don’t trust us when we say our products are great. Include objective and credible proof in the form of data, charts, graphs, quotes, statistics, or testimonials as evidence of your claims.
- Ask them to take action Include a call to action at the end of all videos. When viewers watch your videos, they should feel inspired to take the next step towards purchasing. Tell them what to do next and include an interactive link to the next step in the buying cycle. Use scarcity to compel them to action. Provide a timelimited offer or, for example, say it is “only for the first 20 customers”.

Practical things to keep in mind when producing and distributing your video

Does and don’t when producing your video

- Don’t make the video too long As our attention span is 8 seconds the ideal length of video is 90 seconds

(minimum 30 seconds/maximum 2 minutes).

- Add interactivity where possible Overlay text, charts, animation, questions etc. Visually call out key messages.
- Make sure your audio is high quality.
- Make your video easily shareable
- Enable your video to be shared on multiple media. Track views and attention span. Observe and measure viewer patterns so that you can learn from prospects' actual behaviors and then improve future content.

How to make a good narrative that speak to the right persona in the right way?

- Your narrative should have a beginning, middle, and end.
 - Lead with a story, not with your app or the technology.
 - Don't turn your videos into a product pitch.
 - You'll build more brand affinity and trust by shedding light on a problem your prospects care about rather than by pitching your solutions to them directly.
- The brain is on alert at the beginning of the video and at the end.
 - Make sure the first and last ten seconds are compelling, memorable, and interesting.
- Speak directly to a particular persona in the second person.
 - Do not talk about them in the third person, and avoid using terms like "our clients" and "companies"; instead, use "you" language as often as possible.
 - Use a lot of industry specific vocabulary, terminology, and visuals. If possible, film onsite at a customer's location rather than in your office or in a studio.
- Speak to a particular persona:
 - Do not try to appeal to everyone at once, as you may not fully engage anyone with this approach.
 - Keep your delivery casual and authentic to instill trust. Speak directly to the prospect as if you were having a fireside chat
 - The prospect should be the hero of the story, i.e. do not speak about you and your company.
- Ask rhetorical questions that stimulate pain and anxiety in your prospects in order to demonstrate that you understand their business problems.
 - For example: Are your margins decreasing? Having cash flow problems because you can't collect payments sooner than 90 days? Had another large write off? Lost an important customer recently due to a late delivery?
- Use visual and auditory language to help the prospect imagine a new possible future.
 - For example: "imagine seeing" , "picture yourself", or " how would you like to hear your clients say..." and so on.
- Use contrast whenever possible.
 - Compare prospects' experience now versus what it could be after the implementation of your solution.
 - Call out your competitive differentiators while anchoring your solution in prospects' minds so that they can compare all others against the bar you set.

How to make a good narrative that speak to the right persona in the right way?

- Where possible, use tangible, concrete language.
 - Include quantifiable proof in the form of data or visual pictures.
 - No vague claims like "transform your business with the cloud". This is an emotionless statement.
- Providing customer references and testimonials is much more compelling and effective than selling your company or product yourself.
 - Let others speak for you. A customer testimonial video will always be more believable and compelling than a video of you saying the same thing.
- Surprise and delight them.
 - Use humor to make them smile. We take ourselves and our problems too seriously. Be warm,

memorable, and unique.

Guideline on Creating an Effective Sales Landing Page for Your App

5/3/2019 • 12 minutes to read

Building a landing page that drives a successful buying transaction

Microsoft will drive qualified traffic to AppSource. Though, once a prospect becomes aware of your app, it will be your job to guide them through to a successful buying transaction. Deliberately mapping and architecting the buying journey is critical to ensure a high level of engagement and conversion. Only presenting your app's features and functionality, or just providing a free trial, will not ensure prospects will become buyers. For this you need to have a good landing page that is built to help you capture attention, accelerate your customer acquisition process, and drive buying behavior. The recommendations on this page will help you do so.

Your app landing page should be built to move prospects effectively through the following stages:

Accommodating more languages than English

English is the de facto language that is used on AppSource to ease the validation process and create a uniform user experience.

For you, this means that both the storefront details of your app, and everything that is accessible through it must be in English too.

- This includes: your app's landing page, videos, documentation – such as "Learn more" documents, factsheets, set up instructions, privacy policies, SLAs etc. – as well as help, support- and contact options.

If your app caters to a local language that isn't English, you can improve the user experience by:

- Creating a website that has two landing pages (i.e. two language buttons – cf. one in English and one in the given local language). In so doing your customers can switch to the language they master and thus easily be able to find the right docs and contact info by a shift of a button.
- However, this set up implies that you need to *make two versions of all your docs, support options and landing pages*.
- Note, as mentioned earlier within the "Marketing Validation Checklist" everything that is accessible through the Cloud Partner Portal needs to be in English.

Below you can see how Deex Korea Co Ltd has set up their apps landing page to accommodate two languages, and everything that this entails. We recommend that you use it as inspiration on how to create a user-friendly landing page. If you click on the pictures you will be re-directed to their two respective landing pages.

Example of Deex Korea Co Ltd.'s user-friendly landing pages that accomodates two language options and is set up in accordance with our best practices:

Examples of how other partners have implemented our best practices

To inspire you in creating a good landing page for your app, two of our valued partners, LS Retail and Industry Built, have offered to provide a sample of what a best practice landing page for a Microsoft Dynamics 365 Business Central partner could look like.

Have a look at their app landing pages and use them as inspiration to build your own landing page:

[Industry Built's Build Food app](#)

In the following checklist, we have “broken down” the elements, on their landing pages in order to showcase best practices on design and messaging. More specifically, we are looking into layout and structure elements, content elements, visual elements, anxiety reducing elements and support elements.

Additionally, we have provided specific recommendations on how to apply these elements to help you increase conversion and maximize the effectiveness of your product’s sales landing page.

We urge you to review and implement these best practices on your landing page – in so doing you will contribute in providing the Microsoft community of customers with a consistent buying experience across publishers.

Layout and structure elements

ELEMENT	DESCRIPTION	EXAMPLE
Company	Include the company logo on the page	./media/image30.jpg
App name & app logo	Include a visual logo of your product name and a onesentence positioning statement.	./media/image31.jpg
		./media/image32.jpg
		./media/image33.jpg
Top menu choices	Use clean, straightforward and descriptive menu options.	/media/image34.jpg
Search box	Include a search box so visitors can quickly find what they are looking for.	./media/image37.jpg
Emotional tribal anchor photos	Visuals create an emotional Add-onion. The brain skims over non-emotional photos.	./media/image38.jpg
Visual	Make your page easy to scan, with lots of strong visual imagery.	./media/image39.jpg

Logo

- The upper-left corner of the landing page is the most valuable section of the entire landing page.
- Place your company logo in this location.

If you need help formulating a positioning statement, try the value proposition generator located at [here](#)

- There should ideally be 5 or fewer choices; do not include more than 7 options.
- The menu text should state what the prospect gains if they click on the menu item
- The text should be written from their perspective, not yours.

Recommended menu items:

- How to Buy, Benefits Gained, Why Us, and Contact.

The upper-right corner of the page is usually an ideal spot.

- Faces evoke more emotion than landscapes or machines, and so on.
- Include a happy customer that looks similar to your prospect in terms of age, demographic, and industry, and which shows them dealing with the issues that your prospect can relate to.
- Try not to use stock photos of people or objects.

Engagement

- Too much text forces the brain to skim, skip, and exit. Text engages the logical, analytical brain, but not the emotional brain.
- Keep it clean and straightforward in terms of design and layout. Use lots of pictures, graphs, and screen shots to enhance engagement.

Content elements: Text and messaging

| Element | Description | Example | |-----|-----|-----| | Include a headline question| Get your prospects' attention by asking them a compelling pain-based question that they can relate to. | "Struggling to manage your ingredient inventory and fretting over allergens?" | | You want the prospect to mentally say "YES" as often as possible and to peak their curiosity enough to read more. | | Your questions should be intriguing and customer-centric. | | In general, 8 out of 10 people will read headline copy, but only 2 out of 10 will read the rest. | | Microsoft Dynamics 365 product description| Somewhere on the landing page, make sure you include the standard Microsoft Dynamics 365 Business Central product description provided by Microsoft *This is a requirement because your product is adding value to and building on this foundational solution.* | Insert this paragraph: Microsoft Dynamics 365 Business Central is a comprehensive business management solution for small and medium-size businesses (SMBs) that have outgrown their basic accounting software. From day one, this new application makes ordering, selling, invoicing, and reporting easier and faster. Dynamics 365 Business Central is deeply integrated with Office 365 and includes built-in intelligence, so it is easy to use and helps users make better business decisions. | Messaging (Address their pains)| Pain is a strong motivator of action. | ./media/image40.jpg | | - Identify 1-3 key sources of the client's most prominent pain early on the page. | ./media/image41.jpg | | - Call out the fears that are likely to be holding them back. | - Your landing page text and messaging should predominantly focus on the pain the prospect is experiencing, and NOT the features of your product or service.

Clearly demonstrate to your prospects that you genuinely understand their industry and unique business problems.

- Describe the business challenges they are facing now and the ways their revenue growth, margins, productivity and so on, are being negatively impacted by not taking action now.

| Element | Description | Example | |-----|-----|-----| | **Messaging (Product benefits)** | Paint a clear, visual and desirable picture of what is possible. | ./media/image43.jpg | | ./media/image44.jpg | | ./media/image46.jpg |

- Describe the most significant benefits and rewards that your prospect will realize after purchase.
 - For example, "Save time and money (benefits) by having a system that does all the tracking and calculations for you (features)."
- Don't only list features and app functionality, start with the benefit first, then you can follow with the features.
- Paint a picture of a possible experience the prospect will immediately desire.

Clearly articulate a compelling desired outcome

- If possible, use industry-specific language and vocabulary to resonate with your prospect deeply.
- Choose a particular persona to speak to directly.
- Engage prospects by speaking directly to them using first person "you" language.

| Element | Description | Example | |-----|-----|-----| | **Messaging (Prove your claims)** | Include specific calls-to-action on your app page. | "Reduce how long it takes to set up your recipes

in the morning from 1 hour to 10 minutes.” ./media/image52.jpg|

Don’t make general and abstract claims.

- Use data as often as possible to support your statements.

If you make specific claims, support your claims with proof, while Quantifying impacts and gains.

- The more specific and concrete your promise of value is, the better.
- Abstract concepts such as “more efficiency, more productivity, transform your business” are not emotionally impactful or convincing and do not compel a prospect to act.

Target market If you support multiple countries or languages, this is a key selling feature. • Find a way to show this visually. ./media/image49.jpg

| Element | Description | Example | |-----|-----|-----| | **Messaging (Compelling call-to-action)** | Include specific calls-to-action on your app page|./media/image53.jpg|

- This can be your free trial; a time-limited special price; a scheduled walk-through demonstration; and so on.
- The words “free” and “save” are highly emotional words in the English language, so they should be used.
- Use bright colors, such as orange, yellow, or red, to call attention to your buttons.

Button text should use benefit language rather than descriptive language.

- For example, instead of “Download” write “Click here to start saving money now.”
- Try not to send prospects away from your page – always have an embedded next step in your call to action that brings them back to your landing page.

| Element | Description | Example | |-----|-----|-----| | **Messaging (Create a sense of urgency by teaching the prospects)** | Help your prospect gain a sense of urgency to buy by teaching them one thing about how they can be more efficient or profitable now. |./media/image55.jpg Your bakery profitability will decrease over the next five years due to an increase of 3% in the cost of key inputs, such as wheat and sugar. Want to know five key strategies that can help you mitigate this challenge? Click here to find out how to preserve your profit margin ./media/image56.jpg|

- Show them how their performance in one key business area is below that of their competitors.
- For an example you can provide a quick online self-assessment, a top-10 tips blog post, and much more. Visual elements

ELEMENT	DESCRIPTION	EXAMPLE
Pictures (Differentiation comparison images) Show them, don’t tell them	Show the before and after state.	./media/image57.jpg

- This is a visual image of how your prospects do things now versus how they will be able to do it in the future.
- You are not telling them but showing them using a visual.

ELEMENT	DESCRIPTION	EXAMPLE
Compelling proof screen shots	Visually demonstrate all the claims that you are making.	Quickly and easily view inventory items ./media/image61.jpg ./media/image62.jpg

- Graphic dashboards are the most effective method.
- Zoom in on the main benefit-related features.
- Make sure it is readable, and the benefit is obvious.

- Include a caption.
- Data should be industry specific so that it resonates with the viewer.

You want prospects to see how their data/process would look in your system.

ELEMENT	DESCRIPTION	EXAMPLE
Videos (Tell your story using videos not text)	Include as many videos as possible.	./media/image63.jpg ./media/image64.jpg

- Videos have a much higher level of engagement and viewing time and convey much more than you can ever say with words.

Include at least one customer testimonial video on your app landing page.

- Your client should speak specifically about the pains they had before and the benefits they gained after, not product features. It should be all about your customers, not you.

Include one product demonstration video.

- See the video best practices <https://aka.ms/ReadyToGo>

Elements that reduce anxiety and risk, while increasing trust

Element	Description	Example	Customer testimonials
	Don't sell your product; let your customers do that for you.	./media/image65.jpg ./media/image67.jpg	

- Social proof is more credible and trustworthy to prospects. The purpose of testimonials is to reduce the buyer's anxiety and fear.

Your testimonials should answer the following questions:

- "Will this work for my situation?"
- "What benefit will I really get if I buy this?"
- "Is this going to be too hard?"
- "How long is this going to take?"
- "Can I trust this company?"

ELEMENT	DESCRIPTION	EXAMPLE
Reduce risk	Prospects are afraid of being scammed and taken advantage of on the internet. They are naturally cautious and highly suspect.	Source: Microsoft.com

- You want to convert prospects to buyers.
- Make it easy for them to buy, while reducing their anxiety. Transparency is the key to building trust.
- Make sure that you include a link to a BUY NOW page, which includes full pricing details.
- Give them a compelling offer they cannot refuse. Offer a time-limited trial or special pricing discount if they buy in 30 days.
- Use scarcity to compel action. Offer a 100% money-back guarantee.

We recommend providing three offerings, optimized for three different customer segments. For more recommendations on pricing, see the pricing guide located at <https://mbspartner.microsoft.com/BFI/Topic/64>

ELEMENT	DESCRIPTION	EXAMPLE
Live chat	Include live chat, with a photo of one of your team members smiling at an appropriate time to increase conversion, such as when a prospect clicks the back button on your pricing page.	

- Include their name if possible to build trust.

Support elements: Interactivity and contact options

| Element | Description | Example | |-----|-----|-----| **|SHORT lead capture form|**Include a lead capture form on your page.| ./media/image68.jpg|

- Only ask for their name and email address, you can get the rest later.
- Your forms should not have more than 4 or 5 fields to fill out. You have not yet earned the right or enough trust to ask for too much information at this point.

Most lead capture forms are way too long, demanding, and intimidating, and have low completion rates.

- Note, nobody has the time or is willing to fill out an annoying form, which is of no value to them, especially if it is purely self-serving from your standpoint.

| Element | Description | Example | |-----|-----|-----| **|Contact|**Provide prospects with different contact options based on their readiness to interact with you.| ./media/image69.jpg|

- Ideally, include a phone number and an email address with an employee photo.
- This alone could double your conversion rate.

| Element | Description | Example | |-----|-----|-----| **|AppSource app page link & social share|**Include a link back to your listing on AppSource, so the prospect can return when ready.| Return to AppSource ./media/image70.jpg|

- Also, enable visitors to share and forward your app with others!

| Element | Description | Example | |-----|-----|-----| **|Close them! Add a get started button|**Include a very specific call-to-action button with the option to buy or try.|./media/image71.jpg ./media/image72.jpg|

Getting Started with C/SIDE and AL for On-Premises

3/31/2019 • 2 minutes to read

To get started with a mixed development environment of C/SIDE and AL, you must follow the steps below.

Steps to install Business Central on-premises with C/SIDE and AL development environment

1. Install Business Central on-premises and make sure to include the **AL Development Environment**.
2. Download [Visual Studio Code](#).
3. From Visual Studio Code, locate **Extensions** in the left navigation bar, and then choose **Install from vsix**.
4. Browse to the equivalent folder of `C:\Program Files (x86)\Microsoft Dynamics 365 Business Central\130\AL Development Environment` and then choose **Install**.
5. Now, press **Alt+A**, **Alt+L** to trigger the **AL Go!** command, choose a project, and then choose **Your own server**.
6. Authenticate with the credentials you use for signing into Business Central on-premises.
7. In the launch.json file, update the `"server": "http://localhost"` setting with the URL for server running Business Central on-premises and save the file.
8. In the app.json file, add the `"target": "Internal"` setting.
9. Now, use the Business Central Administration Console to ensure that the settings on the **Development** tab are set as follows:
 - **Allowed Extension Target Level** is set to **Internal**.
 - **Enable Developer Service Endpoint** checkbox is selected.
 - **Enable Loading Application Symbol References at Server Startup** checkbox is selected.
10. Make sure to read and ensure any additional settings here [Running C/SIDE and AL Side-by-Side](#).

TIP

For information about which sandboxes you can choose, see [Choosing Your Dynamics 365 Business Central Development Sandbox Environment](#).

NOTE

Build and get inspired by our sample library on [GitHub](#).

See Also

[AL Development Environment](#)

[FAQ for Developing in AL](#)

Running C/SIDE and AL Side-by-Side

3/31/2019 • 3 minutes to read

Business Central on-premises supports development using both C/SIDE and AL, as well as Designer side-by-side. When new objects are added or changed in C/SIDE these changes must be reflected in the symbol download in Visual Studio Code using the AL Language extension. To enable this reflection, a command and argument called `generatesymbolreference` has been added to `finsql.exe` and you can run it as illustrated below.

Get started generating symbols and compiling all objects

Use the `generatesymbolreference` command specified with the database and server name to add symbol references to the **Object Metadata** table for the specified database.

Given the `generatesymbolreference` command, C/SIDE will traverse all the objects in the database and generate symbols for them. This command should be run at least once to generate the initial set of symbols to which incremental updates can be applied.

Syntax example

```
finsql.exe Command=generatesymbolreference, Database="Demo Database NAV (11-0)", ServerName=.\NAVDEMO
```

This is a lengthy operation. When you run the command, the console returns to an empty command prompt, and does not display or provide any indication about the status of the run. However, the `finsql.exe` may still be running in the background. It can take several minutes for the run to complete, and the symbols will not be generated until such time. You can see whether the `finsql.exe` is still running by using Task Manager and looking on the **Details** tab for **finsql.exe**.

When the process ends, a file named **navcommandresult.txt** is saved to the Dynamics NAV Client connected to Business Central installation folder. If the command succeeded, the file will contain text like

```
[0] [06/12/17 14:36:17] The command completed successfully in '177' seconds.
```

If the command failed, another file named **naverrorlog.txt** will be generated. This file contains details about the error(s) that occurred.

NOTE

The symbol references are stored in the **Symbol Reference** column of the **Object Metadata** table of the database. For on-premises installations, if you experience problems with generating symbols, and the information in the `naverrorlog.txt` did not help, try changing the values in the **Symbol Reference** column to NULL by using an SQL query, then generate the symbols again

Continuously generate symbols each time you compile objects in C/SIDE

The `generatesymbolreference` flag enables incremental symbol generation through the UI or through the compile command passed on the command line. To update the symbols for a set of objects from the UI, start C/SIDE with the `generatesymbolreference` flag, make any desired modifications to your application objects, and compile them.

NOTE

Use `generatesymbolreference` set to yes as a command line argument each time you start finsql.exe to have all compilations add a symbol reference to the **Object Metadata** table. The default setting of the argument is no.

NOTE

If you make changes in C/SIDE and start the C/SIDE development environment without the `generatesymbolreference` flag set to `yes`, the symbols downloaded from Visual Studio Code will not reflect your changes.

Syntax example

```
finsql.exe generatesymbolreference=yes
```

This flag is also a part of the `Compile-NavApplicationObject` PowerShell command and you can use it to compile and generate symbols on a filtered set of application objects through PowerShell. This alternative should be considered if you do not work with the UI in C/SIDE. For more information about it, see [Compile-NavApplicationObject](#).

Business Central on-premises server setting

In addition to the symbol generation setting you have chosen above, you must enable the Business Central on-premises server setting.

1. Go to **Business Central Administration**.
2. Scroll to the **Development** tab and expand the tab.
3. Choose the **Edit** button, and then select the **Enable loading application symbols at server startup** checkbox.

IMPORTANT

This setting must be enabled to allow any symbol generation. If the setting is not enabled, the `generatesymbolreference` setting does not have any effect.

See Also

[Developing Extensions](#)

Creating Runtime Packages for Business Central On-Premises

5/21/2019 • 2 minutes to read

If you want to distribute extensions, you can generate runtime packages that do not contain AL code, but only the final artifacts used by the server at runtime. Runtime packages thereby allow you to protect the intellectual property represented by your AL source code.

When the runtime package is generated on the server, the developer license is checked for permissions to the used extension IDs. The extension in a runtime package can then be installed on servers that do not have a developer license; the server only needs permissions to run the objects, but not to modify or insert them.

Start using runtime packages

The first step in using runtime packages is to have an extension developed and published to an on-premise instance. Next, use the following PowerShell command to connect to the server, find the extension, and download the runtime package.

```
Get-NavAppRuntimePackage
```

For more information about this cmdlet, see [Get-NAVAppRuntimePackage cmdlet](#).

The following example gets the NAV App runtime package with the provided name and version.

```
Get-NAVAppRuntimePackage -ServerInstance DynamicsNAV -AppName 'Proseware SmartApp' -Version 2.3.4.500 -  
ExtensionPath 'Proseware SmartApp_2.3.4.500_runtime.app'
```

For publishing and installing the package, use the [Publish-NavApp](#) and the [Install-NAVApp](#) PowerShell cmdlets.

Limitations

The limitation of runtime packages is that they only work for on-premise installations and therefore cannot be submitted to AppSource. Moreover, the debugging experience is very limited since no source code is available.

See also

[Publish-NAVApp cmdlet](#)

[Install-NAVApp cmdlet](#)

JSON Files

6/17/2019 • 6 minutes to read

In an AL project there are two JSON files; the `app.json` file and the `launch.json` file. These files are generated automatically when you start a new project. The `app.json` file contains information about extension that you are building, such as publisher information and specifies the minimum version of base application objects that the extension is built on. Often the `app.json` file is referred to as the manifest. The `launch.json` file contains information about the server that the extension launches on.

App.json file

The following table describes the settings in the `app.json` file:

SETTING	MANDATORY	VALUE
id	Yes	The unique ID of the extension. When <code>app.json</code> file is automatically created, the ID is set to a new GUID value.
name	Yes	The unique extension name.
publisher	Yes	The name of your publisher, for example: NAV Partner, LLC
brief	No, but required for AppSource submission	Short description of the extension.
description	No, but required for AppSource submission	Longer description of the extension.
version	Yes	The version of the app package.
privacyStatement	No, but required for AppSource submission	URL to the privacy statement for the extension.
EULA	No, but required for AppSource submission	URL to the license terms for the extension.
help	No, but required for AppSource submission	URL to an online description of the extension. The link is used in AppSource and can be the same as the value of the <code>contextSensitiveHelpUrl</code> property or a different link, such as a link to your marketing page.
url	No	URL of the extension package.
logo	No, but required for AppSource submission	Relative path to the app package logo from the root of the package.
dependencies	No	List of dependencies for the extension package. For example: <pre>"dependencies": [{ "appId": "4805fd15-75a5-46a2-952f-39c1c4eab821", "name": "WeatherLibrary", "publisher": "Microsoft", "version": "1.0.0.0"}, {}]</pre>
screenshots	No	Relative paths to any screenshots that should be in the extension package.

SETTING	MANDATORY	VALUE
platform	Yes, if system tables are referenced in the extension	The minimum supported version of the platform symbol package file, for example: "11.0.0.0". See the Symbols for the list of object symbols contained in the platform symbol package file.
application	Yes, if base application objects are extended or referenced. The AL package will be compiled against the application that is present on the server that you connect to. This allows you to write a single AL Language extension for multiple country versions as long as you <i>do not</i> depend on country-specific code. If you <i>do</i> depend on country-specific code you should only try to compile your app against a server set up for that country.	The minimum supported version, for example: <code>"application": "11.0.0.0"</code>
idRange	Yes	For example: <code>"idRange": {"from": 50100, "to": 50149}</code> . A range for application object IDs. For all objects outside the range, a compilation error will be raised. When you create new objects, an ID is automatically suggested.
idRanges	Yes	For example: <code>"idRanges": [{"from": 50100, "to": 50200}, {"from": 50202, "to": 50300}]</code> . A list of ranges for application object IDs. For all objects outside the ranges, a compilation error will be raised. When you create new objects, an ID is automatically suggested. You must use <i>either</i> the <code>idRange</code> or the <code>idRanges</code> setting. Overlapping ranges are not allowed and will result in a compilation error.
showMyCode	No	This is by default set to <code>false</code> and not visible in the manifest. To enable viewing the source code when debugging into an extension, add the following setting: <code>"showMyCode": true</code>
target	No	By default this is <code>Extension</code> . For Dynamics NAV, you can set this to <code>Internal</code> to get access to otherwise restricted APIs and .NET Interop. The Dynamics NAV Server setting must then also be set to <code>Internal</code> .
contextSensitiveHelpUrl	No, but required for AppSource submission	The URL for the website that displays context-sensitive Help for the objects in the app, such as <code>https://mysite.com/documentation</code> .
helpBaseUrl	No	The URL for the website that overtakes all Help for the specified locales. This property is intended for localization apps specifically since the setting overwrites the default URL of <code>https://docs.microsoft.com/{0}/dynamics365/businesscentral</code> .

SETTING	MANDATORY	VALUE
supportedLocales	No	The list of locales that are supported for looking up Help. The value on the list is inserted into the URL defined in the <code>contextSensitiveHelpUrl</code> and <code>helpBaseUrl</code> properties. The first locale on the list is default. An example is <code>"supportedLocales": ["da-DK", "en-US"]</code> .
runtime	Yes	The version of the runtime that the project is targeting. The project can be published to the server with an earlier or the same runtime version. The available options are: <code>1.0</code> - Business Central April 2018 release, <code>2.2</code> - Business Central October 2018 release CU 2, and <code>3.0</code> - Business Central April 2019 release.

Launch.json file

The following table describes the settings in the `launch.json` file. The `launch.json` file has two configurations depending on whether the extension is published to a local server or to the cloud.

Publish to local server settings

SETTING	MANDATORY	VALUE
name	Yes	"Publish to your own server"
type	Yes	Must be set to <code>".a1"</code> . Required by Visual Studio Code.
request	Yes	Request type of the configuration. Must be set to <code>"launch"</code> . Required by Visual Studio Code.
server	Yes	The HTTP URL of your server, for example: <code>"http://localhost serverInstance"</code>
port	No	The port assigned to the development service.
serverInstance	Yes	The instance name of your server, for example: <code>"US"</code>
authentication	Yes	Specifies the server authentication method and can be set to <code>"UserPassword"</code> , <code>"Windows"</code> , or <code>"AAD"</code> . Currently, AAD authentication is supported only for Dynamics 365 Business Central sandboxes. AAD authentication cannot be used for on-premise servers.
startupObjectType	No	Specifies whether the object to open after publishing is a Page type (<code>"Page"</code>) or Table type (<code>"Table"</code>) object. The default is <code>"Page"</code> .

SETTING	MANDATORY	VALUE
startupObjectId	No	Specifies the ID of the object to open after publishing. Only objects of type Page and Table are currently supported.
schemaUpdateMode	No	Specifies the data synchronization mode when you publish an extension to the development server, for example: <code>"schemaUpdateMode": "Recreate"</code> The default value is Synchronize. For more information, see Retaining table data after publishing This feature is not supported in Dynamics NAV.
breakOnError	No	Specifies whether to break on errors when debugging. The default value is <code>true</code> .
breakOnRecordWrite	No	Specifies if the debugger breaks on record changes. The default value is <code>false</code> .
launchBrowser	No	Specifies whether to open a new tab page in the browser when publishing the AL extension (Ctrl+F5). The default value is <code>false</code> . If the value is not specified or set to <code>true</code> , the session is started. If the value is explicitly set to <code>false</code> , the session is not started unless you launch your extension in debugging mode.

Publish to cloud settings

SETTING	MANDATORY	VALUE
name	Yes	"Publish to Microsoft cloud sandbox"
type	Yes	Must be set to <code>".a1"</code> . Required by Visual Studio Code.
request	Yes	Request type of the configuration. Must be set to <code>"1a1"</code> . Required by Visual Studio Code.
startupObjectType	No	Specifies whether the object to open after publishing is a Page type (<code>"Page"</code>) or Table type (<code>"Table"</code>) object. The default is <code>"Page"</code> .
startupObjectId	No	Specifies the ID of the object to open after publishing. Only objects of type Page and Table are currently supported.
tenant	No	Specifies the tenant to which the package is deployed. If you specify multiple configurations, a drop-down of options will be available when you deploy. This parameter must contain a tenant AAD domain name, for example <code>mycustomer.onmicrosoft.com</code> .

SETTING	MANDATORY	VALUE
sandbox	No	Specifies which sandbox to use in cases where multiple sandboxes are owned by the same tenant.

The platform symbol file

The platform symbol file contains all of the base app objects that your extension builds on. If the AL Language extension in Visual Studio Code detects that the referenced symbols are not present on local disk, you will get a visual prompt in Visual Studio Code to download the symbols from one of the servers specified in the launch.json file. For more information about the platform symbol file, see [Symbols](#).

See Also

[AL Development Environment](#)
[Debugging in AL](#)
[Security Setting and IP Protection](#)
[Configure Context-Sensitive Help](#)

Security Setting and IP Protection

5/24/2019 • 2 minutes to read

When developing an extension, your code is by default protected against downloading or debugging. Read below about the security setting and adding Intellectual Property (IP) protection against downloading or debugging into an extension to see the source code in the extensions.

The extension development package provides a pre-configured setting for IP protection against viewing or downloading the code of the extensions. However, this setting can also be controlled in the manifest; the `app.json` file.

IP protection setting

When you start a new project, an `app.json` file is generated automatically, which contains the information about the extension that you are building on. The `app.json` file contains a setting called `showMyCode`, which controls whether it is possible to debug into the extension, when that extension is taken as a dependency. The default value of this property is set to **false**. This means that debugging into an extension to view the code is not allowed. For a more refined setting, you can specify the `NonDebuggable` attribute on methods and variables. For more information, see [NonDebuggable Attribute](#).

NOTE

The `showMyCode` setting is not visible in the `app.json` file when it is generated.

NOTE

Even though `showMyCode` is set to **false**, you will still be able to view that code if an extension is deployed through Visual Studio Code, as opposed to deploying using a cmdlet or via AppSource.

Changing the IP protection setting

If you want to allow debugging into an extension to view the source code, you can add the `showMyCode` property in the `app.json` file and set the property value to **true**. For example, if a developer develops extension A and he or someone else on the team develops extension B, and B depends on A, then debugging B will only step into the code for A if a method from A is called and if the `ShowMyCode` flag is set to **true** in the `app.json` for extension A as shown in the example below:

```
"showMyCode": true
```

By adding this setting, you *enable debugging* into an extension to view the source code when that extension is set as a dependency.

See Also

[JSON Files](#)

[AL Development Environment](#)

[NonDebuggable Attribute](#)

Developing for Multiple Platform Versions

3/31/2019 • 2 minutes to read

The AL language extension is compatible with multiple platform versions. You can install the AL Language extension from the Visual Studio Code marketplace and use it to develop solutions for Dynamics 365 Business Central.

Defining the platform version

To set the platform version, add the **runtime** property in the `app.json` file. This attribute defines the platform version that the extension is targeting. Depending on the platform version, some features become available, while some features are not supported. For example, OData-bound actions can only be used when the platform version is 2.0 or higher.

NOTE

AL Language extension is not compatible with Dynamics NAV 2018 version backwards. For Dynamics NAV 2018 development, the traditional method should be used. You must install the Visual Studio Code extension from the `ALLanguage.vsix` file shipped on the DVD.

Version compatibility

The following two elements are compared when you publish an extension.

1. The runtime version of the extension defined in the `app.json` file.
2. The runtime version of the platform that the extension is targeting.

In the `app.json` file, set the extension **runtime** version lower than the platform version. When you set the extension to a higher **runtime** version, the extension package may contain certain features that the platform may not support which would result in an error. Therefore, you must lower the extension runtime version than the one that platform supports in order to publish your extension.

Things to be aware of

1. An error will be thrown when you publish an extension with a higher runtime version than the one that platform supports. For example, if you set the runtime value to `2.0`, you get the following error message.

The runtime version of the extension package is currently set to '2.0'. The runtime version must be set to '1.0' or earlier in the `app.json` file in order to install the extension package on this platform.

2. When you lower the extension runtime version, you may get warnings about the newest features not supported by the earlier versions of the platform.
3. A best-effort compilation is made when you publish an extension compiled with a lower runtime version. This is allowed in order to avoid recompilation of the extension package every time you upgrade the platform.

See Also

[Debugging in AL](#)
[Developing Extensions](#)

Debugging

4/10/2019 • 6 minutes to read

The process of finding and correcting errors is called *debugging*. With Visual Studio Code and the AL Language extension you get an integrated debugger to help you inspect your code to verify that your application can run as expected. You start a debugging session by pressing F5. For more information about Debugging in Visual Studio Code, see [Debugging](#).

IMPORTANT

To enable debugging in versions before Business Central April 2019, the `NetFx40_LegacySecurityPolicy` setting in the `Microsoft.Dynamics.Nav.Server.exe.config` file must be set to **false**. This requires a server restart.

IMPORTANT

To use the development environment and debugger, you must make sure that port `7049` is available.

There are a number of limitations to be aware of:

- "External code" can only be debugged if the code has the `showMyCode` flag set. For more information, see [Security Setting and IP Protection](#).
- The debugger launches a new client instance each time you press F5. If you close the debugging session, and then start a new session, this new session will rely on a new client instance. We recommend that you close the Web client instances when you close a debugging session.

TIP

To control table data synchronization between each debugging session, see [Retaining table data after publishing](#).

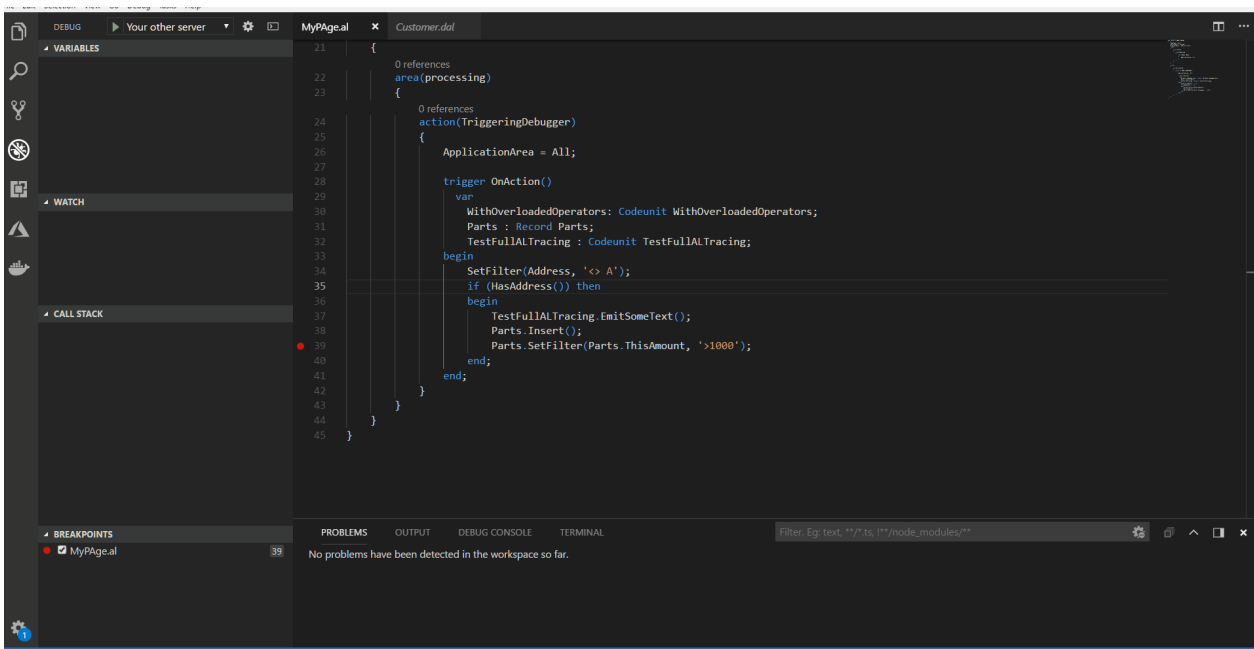
Breakpoints

The basic concept in debugging is the *breakpoint*, which is a mark that you set on a statement. When the program flow reaches the breakpoint, the debugger stops execution until you instruct it to continue. Without any breakpoints, the code runs without interruption when the debugger is active. You can set a breakpoint by using the **Debug Menu** in Visual Studio Code. For more information, see [Debugging Shortcuts](#).

Set breakpoints on the external code that is not part of your original project. You can step into the base application code by using the **Go To Definition** feature, and set breakpoints on the referenced code which is generally a `.dal` file. To set a breakpoint on the external code or base application code, you do the following:

- Use **Go To Definition** which opens the "external file" and then a breakpoint can be set.
- Using the debugger, step into the code, and then set a breakpoint.

In the following video illustration, the `Customer.dal` is an external file. A breakpoint is set in the `Customer.dal` file which is referenced from your AL project to stop execution at the marked point.



For more information about **Go To Definition**, see [AL Code Navigation](#).

Break on Errors

Specify if the debugger breaks on the next error by using the `breakOnError` property. If the debugger is set to `breakOnError`, then it stops execution both on errors that are handled in code and on unhandled errors.

The default value of the `breakOnError` property is **true**, which means the debugger stops execution that throws an error by default. To skip the error handling process, set the `breakOnError` property to **false** in the `launch.json` file.

TIP

If the debugging session takes longer, you can refresh the session by pressing the Ctrl+Shift+P keys, and select the Reload Window.

Break on Record changes

Specify if the debugger breaks on record changes by using the `breakOnRecordWrite` property. If the debugger is set to break on record changes, then it breaks before creating, modifying, or deleting a record. The following table shows each record change and the AL methods that cause each change.

RECORD CHANGE	AL METHODS
Create a new record	INSERT Method (Record)
Update an existing record	MODIFY Method (Record) , MODIFYALL Method (Record) , RENAME Method (Record)
Delete an existing record	DELETE Method (Record) , DELETEALL Method (Record)

The default value of the `breakOnRecordWrite` property is **false**, which means the debugger is not set to break on record changes by default. To break on record changes, you can set the `breakOnRecordWrite` property to **true** in the `launch.json` file. For more information, see [JSON Files](#).

Debugging shortcuts

KEYSTROKE	ACTION
F5	Start debugging
Ctrl+F5	Start without debugging
Shift+F5	Stop debugging
Ctrl+Shift+F5	Start debugging without publishing. Using this command on a changed, but not published code may trigger false existing breakpoints. For example, if you modify method "foo", add two lines and put a breakpoint on the second line and then start debugging without publishing, that breakpoint will not be hit, or if it is hit is not your new code that it breaks. If it breaks, it will break on the line that the server thinks the breakpoint is, based on the last published code.
F10	Step over
F11	Step into
Shift+F11	Step out
F12	Go To Definition

For more shortcuts, see [Debugging](#).

Debugging SQL behavior

Traditionally, debugging AL has been about examining behavior of the language runtime, for example, looking into the content of local variables at a breakpoint. As of Business Central April 2019, the AL debugger also offers the capability to examine the impact that your AL code has on the Business Central database.

View database statistics

In the **VARIABLES** pane in debugger, expand the node to get insights such as the current network latency between the Business Central Server and the Business Central database, the total number of SQL statements executed, and the total number of rows read, as well as insights into the most recent SQL statements executed by the server. The following insights are part of the database statistics:

Current SQL latency (ms)	When the debugger hits a breakpoint, the Business Central Server will send a short SQL statement to the database and measure how long time it takes. The value is in milliseconds.
Number of SQL Executes	This number shows the total number of SQL statements executed in the debugging session since the debugger was started.
Number of SQL Rows Reads	This number shows the total number of rows read from the Business Central database in the debugging session since the debugger was started.

View SQL statement statistics

The database insights also let you peek into the most recent and the latest long running SQL statements executed by the server. To view a list of these, expand either the **<Last Executed SQL Statements>** or **<Last Long Running SQL Statements>** node. The following insights are part of the SQL statement statistics:

Statement	The SQL statement that the AL server sent to the Business Central database. You can copy this into other database tools, such as SQL Server Management Studio, for further analysis.
Execution time (UTC)	The timestamp (in UTC) of when the SQL statement was executed. You can use this to infer whether the SQL statement was part of the AL code between current and last breakpoint (if set).
Duration (ms)	The duration in milliseconds of the total execution time of the SQL statement measured inside the Business Central Server. You can use this to analyze whether you are missing indexes (Business Central keys), or to experiment with performance of database partitioning and/or compression.
Approx. Rows Read	This number shows the approximate number of rows read from the Business Central database by the SQL statement. You can use this to analyze whether you are missing filters.

The number of SQL statements tracked by the debugger can be configured in the Business Central Server. The default value is 10.

NOTE

For Business Central on-premises, the Business Central Server instance has several configuration settings that control the SQL statistics that are gathered and then displayed in debugger, like whether long running SQL statements or SQL statements are shown. If you are not seeing the insights that you expect to see in debugger, check the server configuration. For more information, see [Configuring Business Central Server](#).

NonDebuggable Attribute

To restrict the ability to debug certain methods and/or variables, see [NonDebuggable Attribute](#).

See Also

[Developing Extensions](#)

[JSON Files](#)

[AL Code Navigation](#)

Working with Rapid Application Development

6/4/2019 • 2 minutes to read

Working with Visual Studio Code and Dynamics 365 Business Central you can benefit from Rapid Application Development (RAD) on large code projects. RAD allows faster development on projects with a large number of files by doing a delta compilation and publishing only on those application objects that have changed during development in Visual Studio Code. RAD publishing is an interim state and does not replace a full publish.

How RAD works

The files that have been changed by the application developer within Visual Studio Code are persisted in a special RAD (.rad) file during builds. This file is saved in the .vscode folder of the code project. RAD changes are the changes of application objects within a RAD session. Only application objects, page customization objects, and profile objects are handled for RAD. RAD changes will not be persisted during save, only during build, publish, and debug.

IMPORTANT

If you change many files and close Visual Studio Code without a build (Ctrl+Shift+B), publish (Ctrl+F5, Ctrl+Shift+F5) or debug (F5, Shift+F5) all the RAD changes will be lost. This means that if you, in the next Visual Studio Code session perform a RAD publishing, this is done on the latest changes and not the prior changes. This can lead to an incomplete published package if it succeeds. It is therefore a best practice to do a regular publish. You can always check the RAD file in the code project to see what application objects are going to be changed during publishing.

In scenarios when application IDs are renamed, or refactored it is also a best practice to first do a full publishing, and then a RAD publishing for the consecutive changes. RAD does not check for application ID changes and ID changes can occur in a wrongly published application.

A RAD published file will not contain the following files that are normally packaged during regular publishing:

- Translation files
- Permission files
- Custom word and report rdlc layout files
- Table data
- Web service definitions

These files will need to be re-generated with full publishing (Ctrl+F5).

A RAD file will be deleted as a result of a successful publishing.

NOTE

If RAD publishing fails, then you must do a full publishing before performing another RAD publishing. The final state of an application must be built using full publishing, and never with Rad publishing.

RAD shortcuts

There are two commands for starting a RAD-based action.

SHORTCUT	DESCRIPTION
Ctrl+Alt+F5	Start RAD debugging without publishing.
Alt+F5	Start RAD publishing.

See also

[Developing Extensions in AL](#)

[Debugging](#)

The Lifecycle of Apps and Extensions for Business Central

3/31/2019 • 6 minutes to read

When you build an app or extension to Business Central and get that published to AppSource, it becomes an app like so many others - the app itself can be updated, and the platform that it sits on, the Business Central service itself, will also get updated. But what happens after your app gets published?

When your app has passed all of our validations and has gone live to App Source, customers can install your extension and use it for their business. You might ask, what now happens with my app?

The following sections describe the different upgrade scenarios that we have seen play out as we update Business Central.

Scenario 1: Business Central service update

You don't need to make any bug fixes, feature adds, or app changes to your app. It continues to work fine without any interaction on your part.

Impact

The monthly service upgrades to Business Central do not impact your app. Your app just gets moved along and no upgrade code from your app needs to get used. Business Central itself gets upgraded on your tenant, and once complete, the customer sees no difference with your app.

Scenario 2: App update

You (our partner) add some features to your app and also some minor bug fixes. The app is submitted for validation. The app passes validation and gets checked into the service. This is now the active app for any new tenants and also for existing tenants that have never had your app installed before

Impact

Customers can either do an uninstall and then reinstall on their own, or they can ask their partner do it on their behalf from the Extension Management window within Business Central. Otherwise, they would have to wait until our every 6-month major release. That is the only time we do a force upgrade of extensions (except for critical bug hotfix extension updates)

Scenario 3: Reported bugs in your app

You (our partner) has various customers report some bugs that are impacting their usage of the app. The bugs aren't critical but they are important. The partner makes the fixes in the app and resubmits for validation. The app passes validation and gets checked into the service. This is now the active app for any new tenants and also for existing tenants that have never had your app installed before

Impact

We still do not force the upgrade of this app to this latest version on all of the tenants. Some tenants may not be using the functionality that includes this bug and continue to work fine on the current version of the app. Therefore, you should work directly with all of the impacted customer tenants to uninstall and reinstall to get the latest app version that contains fixes for the bug.

Scenario 4: Critical bug in your app

Critical bug within the app is found in tenants. These tenants cannot do their day to day work due to this bug. This fits into our hotfix scenario as it is critical. A support ticket is related to this case. The partner immediately provides a fixed app for validation. The validation team makes this top priority and does validation ASAP. Fixed app passes validation and gets checked into the service. All tenants with the app are refreshed automatically by the Microsoft team to this fixed version.

Scenario 5: Microsoft feature breaks your app

Microsoft has to break your app file for a needed Business Central core change. Some reasons for breaking could be security, bugs in the underlying code, high priority feature adds, and so on. Keep in mind, we do our very best to not break your app through our changes. We try and find proper ways of doing the changes without breaking your app. However, if we can't find a proper (non-breaking) way, then we could break your app. This won't be as likely in a minor update release (unless a security change is required on our part and that is the change that breaks you), but it can be more likely in our major (every 6-month) releases.

Impact

Here is our process when this takes place:

- First of all, Microsoft will not make a breaking change in the production environment at any point. Therefore, existing tenants are not expected to see this breaking change occur.
- When we make a breaking change, we do it in a build branch that is for a future release (monthly service minor or major release)
- We notify the partner in advance and give the partner ample time to fix their app, get it validated, and have it ready
- The fixed app will already be in our service and slotted as required for when your tenant is to be moved to the Business Central release that has the app breaking change
- As a result, the customer (tenant owner) should never see their Business Central break. Because the tenant gets moved from one monthly service update of Business Central to another, the tenant is being upgraded to the release of ours that breaks the specific app. However, our service detects that there is a new required version of that app (your fixed version). Therefore, we auto install the fixed version of the app for the tenant

Conclusions

You're responsible for your app. You own the process of updating the app and providing upgrade code if the schema changes between versions of the app.

If a customer uninstalls your app, and then installs it again later, then when they install the app the second time, they get the latest version from AppSource.

How Microsoft handles your app

When Microsoft upgrades a tenant with a service update, your app is tested against the new service version. If the app breaks, Microsoft rolls back to the previous healthy state. Your customer never learns that anything was about to break.

When a tenant uninstalls and reinstalls an extension via the Extension Management page or AppSource, there is platform logic that determines whether an *Install* or an *Upgrade* must take place. We detect which version of the extension the tenant previously had installed and perform the appropriate action. Therefore, the result of manually uninstalling/installing the extension is the exact same as an automated upgrade.

Additionally, there will not be any data loss during uninstall, install, or upgrade actions. Data for extensions is stored in its own tables in the tenant database. Before an extension gets installed, it first gets synchronized on the tenant database. This step is implicit and happens automatically when a tenant installs an extension. This synchronization process creates the database tables for the extension. Once the extension is installed and the

tenant is using it, extension-specific data will get stored in these tables.

When an extensions gets uninstalled, these tables do not get removed. Therefore, when the extension gets reinstalled (or upgraded), the data is still available. You do not need to worry about data loss for choosing the uninstall/install route. However, do keep in mind that if any actions are being performed on the tenant while the extension is uninstalled, the extension's events and such will not be firing, and your app may miss the creation of new data. Try to perform the uninstall/install while the tenant is not online.

See Also

[Retaining table data after publishing](#)

[Checklist for Submitting Your App](#)

[Upgrading Extensions](#)

Converting Extensions V1 to Extensions V2

3/31/2019 • 5 minutes to read

Extensions are a programming model where functionality is defined as an addition to existing objects and defines how they are different or modify the behavior of the solution. This article explains the steps involved in converting V1 extensions, written in C/SIDE, to V2 extensions; written using the AL Language extension for Visual Studio Code. The overall steps for the conversion are:

1. Convert the source code from C/AL to the AL syntax.
2. Complete the development of the extension in AL syntax.
3. Write upgrade code to restore and modify data from the V1 Extension tables.
4. Build the extension.
5. Uninstall the V1 extension, and publish and run upgrade on the V2 extension.

Convert the source code from V1 to V2

To convert the source code, you must use the Txt2Al conversion tool. The Txt2Al conversion tool allows you to take existing application objects that have been exported in .txt format and convert them into the new .al format. The .al format is used when developing extensions for Dynamics 365 Business Central. For more information about converting the source code, see [Txt2Al Conversion Tool](#).

Complete the development of the extension

When the source code has been converted using the Txt2Al conversion tool, open the project folder in Visual Studio Code, and then modify or add code to the new version as needed. For more information about getting started with Visual Studio Code and the AL Language extension, see [Getting Started with AL](#).

You might run into compilation errors, which can typically be caused by:

- Object IDs that have changed. The conversion tool tries to convert your code into the object ID range allowed for Extensions V2.
- Field or control names look different; the AL syntax requires names, this means that no empty or default names are allowed.
- Menu suites do not exist in Extensions V2.
- .NET references are not allowed; there is no support for .NET types. Instead you must use the classes that replace .NET calls. For more information, see [Reference](#).

IMPORTANT

In the app.json, keep the ID the same as in the V1 extension. Also, make sure to increase the version number.

The version number has the format `Major.Minor.Build.Revision`, for example `1.5.0.0`. To increase the version number, you must increase the value of `Major`, `Minor`, or `Build` by at least one, for example `1.5.1.0` or `1.6.0.0`.

To use `NAVAPP.RestoreArchiveData()` method for upgrading, you must not change the IDs of the tables that are being restored; this means that tables from your V1 extension must have the same IDs in the V2 extensions.

Write upgrade code to move data from V1 Extensions

Just like with V1 extensions, you have to write code to handle data in tables during upgrade. Writing code for the V1-to-V2 extension upgrade is very similar to the code that you have been writing for V1 Extensions. The differences are:

- Instead of adding code to normal codeunit, you write code in an upgrade codeunit, which is a codeunit whose [SubType property](#) is set to **Upgrade**.
- Instead of adding code to the user-defined methods `OnNavAppUpgradePerDatabase()` or `OnNavAppUpgradePerCompany()`, you add code to one or more of the following system triggers for data upgrade. These triggers are invoked when a data upgrade is started. The following table lists the upgrade triggers in the order in which they run.

TRIGGER	DESCRIPTION
OnCheckPreconditionsPerCompany() or OnCheckPreconditionsPerDatabase()	Used to check that certain requirements are met in order to run.
OnUpgradePerCompany() or OnUpgradePerDatabase()	Used to run the actual upgrade work
OnValidateUpgradePerCompany() or OnValidateUpgradePerDatabase()	Used to check that the upgrade was successful

However, for this one-time conversion, all of the same **NAVAPP** system methods you used in V1 extensions work with V2 extensions and can be called from any of the upgrade triggers.

METHOD	DESCRIPTION
<code>NAVAPP.DeleteArchiveData(70000000)</code>	Deletes the archived data from table 70000000.
<code>NAVAPP.GetArchiveRecordRef(70000000, archRef)</code>	Gets a record ref to the archived data from table 70000000.
<code>archVersion := NAVAPP.GetArchiveVersion()</code>	Gets the version of the archived data from the old extension.
<code>NAVAPP.RestoreArchiveData(70000000)</code>	Restores the data from the archive of table 70000000.

By using these methods, you can restore or move all your data from the old V1 extension into the new V2 by running an upgrade.

IMPORTANT

To use `NAVAPP.RestoreArchiveData()`, you must not change the IDs of the tables that are being restored; this means that tables from your V1 extension must have the same IDs in the V2 extensions.

Example

This code illustrates a simple upgrade codeunit for restoring the V1 extension data for extension table `70000000`.

```
codeunit 70000001 MyExtensionUpgrade
{
    Subtype=Upgrade;

    trigger OnUpgradePerDatabase();
    begin
        NAVAPP.RestoreArchiveData(70000000);
    end;
}
```

TIP

Typing the shortcut `ttrigger` in Visual Studio Code will create the basic structure for a trigger.

Build the extension package

Press Ctrl+Shift+B to compile and build the extension complete with the application objects and upgrade codeunit.

Run the upgrade

The final task of the conversion is to publish the V2 extension, and run the data upgrade. The following steps use an example that upgrades a V1 extension that is called 'ProsewareStuff' and has the version '1.5.0.0'. The V1 extension is published, installed, and populated with data. The V2 extension has the same name (and ID), but it has the version '1.5.1.0'. The Dynamics 365 Business Central service instance is called 'DynamicsNAV', and there is only one tenant.

The steps use the Dynamics NAV Server Administration tool.

1. Uninstall the V1 extension.

```
Uninstall-NAVApp -ServerInstance NAV -Name ProsewareStuff -Version 1.5.0.0
```

This removes the tables from the SQL Server database and archives extension data.

IMPORTANT

The V1 extension must be uninstalled before upgrading it to a V2 extension.

2. Publish the V2 extension. This example assumes the extension is not signed.

```
Publish-NAVApp -ServerInstance DynamicsNAV -Path .\ProsewareStuff_1.5.1.0.app -SkipVerification
```

This validates the extension syntax against server instance, and stages it for syncing.

3. Synchronize the V2 extension with the database.

```
Sync-NAVApp -ServerInstance NAV -Name ProsewareStuff -Version 1.5.1.0
```

This adds tables from V2 extension to SQL database.

4. Run the upgrade process to handle archived data from the V1 extension.

```
Start-NAVAppDataUpgrade -ServerInstance NAV -Name ProwareStuff -Version 1.5.1.0
```

This runs the upgrade logic defined by the upgrade codeunit in the extension, and installs the new V2 extension.

5. (optional) Unpublish the V1 extension.

```
Unpublish-NAVApp -ServerInstance NAV -Name ProwareStuff -Version 1.5.0.0
```

This removes the unused extension package from server.

Going forward

The upgrade code unit becomes an integral part of the extension. The **NAVAPP** methods were mainly be used for the conversion from V1 to V2. After converting the extension, you should begin to write upgrade code as described in [Upgrading Extensions](#).

See Also

[Getting Started with AL](#)

[Keyboard Shortcuts](#)

[AL Development Environment](#)

The Txt2Al Conversion Tool

3/31/2019 • 3 minutes to read

The Txt2Al conversion tool allows you to take existing Dynamics NAV objects that have been exported in .txt format and convert them into the new .al format. The .al format is used when developing extensions for Dynamics 365 Business Central. Converting the objects consists of following two steps:

1. Exporting the objects from C/SIDE in a cleaned format.
2. Converting the objects to the new syntax.

Using the Txt2Al conversion tool

To run the Txt2Al conversion tool, follow the steps outlined below.

1. Start with a clean Dynamics NAV database and compile the database.
It is **very** important that you compile the database to get the right result in the next step.
2. Make an export of **all the baseline objects** in the command line using the following syntax:

```
finsql.exe Command=ExportToNewSyntax, File=<filename.txt>, Database="<databasename>", ServerName=<servername>, Filter=Type=table;ID=<tableID>
```

The following example exports the table **225** from the Demo Database NAV (13-0) database:

```
finsql.exe Command=ExportToNewSyntax, File=exportedBaselineObjects.txt, Database="Demo Database NAV (13-0)", ServerName=. \NAVDEMO, Filter=Type=table;ID=225
```

3. Import your solution using the import option in C/SIDE and compile the database.
It is **very** important that you compile the database to get the right result in the next step.
4. Export all **new and/or modified** objects using the following syntax:

```
finsql.exe Command=ExportToNewSyntax, File=<filename.txt>, Database="<databasename>", ServerName=<servername>, Filter=Type=table;ID=<tableID>
```

The following example exports the table **231** from the Demo Database NAV (13-0) database:

```
finsql.exe Command=ExportToNewSyntax, File=exportedNewModifiedObjects.txt, Database="Demo Database NAV (13-0)", ServerName=. \NAVDEMO, Filter=Type=table;ID=231
```

5. Run the Set-ObjectPropertiesFromMenuSuite cmdlet which will convert MenuSuite information on pages and reports in the generated AL objects to enable them for search. For more information, see [Making Pages and Reports Searchable in the Web client](#)
6. Create .delta files using the Compare-NAVApplicationObject powershell script. For more information, see [Generating DELTA Files](#).
7. Go to the `\Program Files(x86)\Microsoft Dynamics 365 Business Central\130\RoleTailored Client` folder and locate the **txt2al.exe** converter tool.
8. Run the tool from the command line using the following syntax:

```
txt2al --source --target --rename --type --extensionStartId --injectDotNetAddIns --dotNetAddInsPackage --dotNetTypePrefix --translationFormat --addLegacyTranslationInfo
```

Parameters

PARAMETER	DESCRIPTION
--source=Path	Required. The path of the directory containing the .delta files.
--target=Path	Required. The path of the directory into which the converted AL files will be placed.

PARAMETER	DESCRIPTION
--rename	Rename the output files to prevent clashes with the source .txt files.
--type=ObjectType	The type of object to convert. Allowed values: Codeunit, Table, Page, Report, Query, XmlPort
--extensionStartId	The starting numeric ID of the extension objects (Default: 70000000). It will be incremented by 1 for each extension object.
--help	Show help screen.
--injectDotNetAddIns	Inject the definition of standard .NET add-ins in the resulting .NET package. The standard .NET add-ins are a set of add-ins that are embedded into the platform.
--dotNetAddInsPackage=Path	Specify the path to an AL file containing a definition for a .NET package containing .NET type declarations that should be included in the .NET package definition produced by the conversion. This should be used to inject a custom set of .NET control add-in declarations. The file should contain something similar to the example shown below.
--dotNetTypePrefix	Specify a prefix to be used for all .NET type aliases created during the conversion.
--translationFormat=ObjectType	Specify the format to use when generating translation files. The allowed values are: Xliff, Lcg.
--addLegacyTranslationInfo	Add information to the translation file that can be used to migrate existing translations/translated resources. During conversion, XLIFF files from all the ML properties in the app are extracted. If this switch is set, a comment is added in the generated XLIFF that specifies what the ID of the translation item would be in C/SIDE. This acts as a mapping that allows you to convert existing translation resources for your app.

NOTE

It is recommended to only use the conversion tool for export. Importing objects that have been exported can damage your application.

TIP

You can use the Dynamics NAV Development Shell cmdlet `Export-NAVApplicationObject` with the `-ExportToNewSyntax` flag set instead of using finsql. From the command prompt in the Dynamics NAV Development Shell, run `Get-Help Export-NAVApplicationObject -full` to see the full syntax.

See Also

[Developing Extensions](#)

[AL Development Environment](#)

[Page Extension Object](#)

Generating Delta files

5/24/2019 • 2 minutes to read

You can use the `Compare-NAVApplicationObject` powershell cmdlet to generate .delta files from two versions of a set of application objects.

The cmdlet has a `ExportToNewSyntax` switch that allows generating .delta files that can be used as a starting point for creating extensions. Setting the `ExportToNewSyntax` flag generates .delta files that contain additional information needed to generate the correct structure and layout of extension objects.

IMPORTANT

The Txt2AI conversion tool will reject .delta files that were generated without using the `-ExportToNewSyntax` flag.

Using the `ExportToNewSyntax` switch for the `Compare-NAVApplicationObject` cmdlet produces a .delta file that can be converted to an extension.

The ExportToNewSyntax flag

PARAMETER	DESCRIPTION
Type	SwitchParameter
Aliases	None
Position	Named
Default value	None
Accept pipeline input	False
Accept wildcard characters	False

Example

```
Compare-NAVApplicationObject -OriginalPath "C:\PageWith2Controls.txt" -ModifiedPath "C:\PageWith3Controls.txt"
-ExportToNewSyntax
```

See Also

[The Txt2AI Conversion Tool](#)

[Developing Extensions](#)

[Converting Extensions V1 to Extensions V2](#)

Exporting data for Extensions

3/31/2019 • 2 minutes to read

For your extension to run properly, configuration and starting data such as permission sets and table data may be needed. An extension can include the following types of data that can be imported for the tenant during the installation of the extension.

- Permission sets
- Web services
- Starting table data
- Custom report layouts

The data must be exported into files to be included in the extension. To use the export functions you must use a container sandbox environment for Dynamics 365 Business Central. For more information, see [Get started with the Container Sandbox Development Environment](#).

To export permission sets

1. Open the Microsoft Dynamics NAV Development Shell.
2. Export the relevant permission set using the `Export-NAVAppPermissionSet` cmdlet to export the permission set to a file. The following command exports the BASIC permission set.

```
Export-NAVAppPermissionSet -ServerInstance DynamicsNAV110 -Path '.\PermissionSet.xml' -PermissionSetId BASIC
```

NOTE

Export each permission set to a separate XML file.

3. Add the exported permission set files to the Visual Studio Code project that contains your extension.

WARNING

If you do not include a permission set with your extension, only users with the SUPER permission set will be able to use the extension.

To export web services

1. Open the Microsoft Dynamics NAV Development Shell.
2. Export the relevant web service using the `Export-NAVAppTenantWebService` cmdlet to export the web service to a file. The following command exports the Customer Card page.

```
Export-NAVAppTenantWebService -ServerInstance DynamicsNAV110 -Path TenantWebService.xml -ServiceName Customer -ObjectType Page -ObjectId 21
```

NOTE

Export each web service to a separate XML file.

3. Add the exported web services files to the Visual Studio Code project that contains your extension.

To export table data

1. Open the Microsoft Dynamics NAV Development Shell.
2. Export the relevant data using the `Export-NAVAppTableData` cmdlet to export the data to a file. This includes setting the path to a folder where you want the .navxdata file created. A data file in the format of TAB.navxdata will be created. (Example: TAB10000.navxdata).

```
Export-NAVAppTableData -ServerInstance DynamicsNAV110 -Path 'C:\NAVAppTableData' -TableId 10000
```

NOTE

Export the data for each table to a separate XML file.

3. Add the exported table data files to the Visual Studio Code project that contains your extension.
4. Call the procedure in a Codeunit with the Subtype property `Install` or `Upgrade` and specify the table ID in the `NavApp.LoadPackageData` procedure as shown in the following example.

```
codeunit 50100 MyExtensionUpgrade
{
    Subtype = Upgrade;
    trigger OnUpgradePerDatabase()
    begin
        NavApp.LoadPackageData(50100);
    end;
}
```

WARNING

An extension can only include table data for new tables that are added as part of the extension.

To export custom report layouts

1. Open the Microsoft Dynamics NAV Development Shell.
2. Export the relevant report layouts using the `Export-NAVAppReportLayout` cmdlet to export to a file:

```
Export-NAVAppReportLayout -ServerInstance DynamicsNAV110 -Path .\ReportLayout.xml -LayoutId 1
```

NOTE

Export each custom report layout to a separate XML file.

3. Add the exported custom report files to the Visual Studio Code project that contains your extension.

See Also

[Developing Extensions in AL](#)

[Converting Extensions V1 to Extensions V2](#)

[Writing Extension Install Code](#)

Writing Extension Install Code

5/28/2019 • 2 minutes to read

There might be certain operations outside of the extension code itself that you want performed when an extension is installed. These operations could include, for example, populating empty records with data, service callbacks and telemetry, version checks, and messages to users. To perform these types of operations, you write extension install code. Extension install code is run when:

- An extension is installed for the very first time.
- An uninstalled version is installed again.

This enables you to write different code for initial installation and reinstallation.

How to write install code

You write install logic in an *install* codeunit. This is a codeunit that has the [SubType property](#) is set to **Install**. An install codeunit supports two system triggers on which you can add the install code.

TRIGGER	DESCRIPTION
OnInstallAppPerCompany()	Includes code for company-related operations. Runs once for each company in the database.
OnInstallAppPerDatabase()	Includes code for database-related operations. Runs once in the entire install process.

The install codeunit becomes an integral part of the extension version. You can have more than one install codeunit. However, be aware that there is no guarantee on the order of execution of the different codeunits. If you do use multiple install units, make sure that they can run independent of each other.

Install codeunit syntax

The following code illustrates the basic syntax and structure of an install codeunit:

```
codeunit [ID] [NAME]
{
    Subtype=Install;

    trigger OnInstallAppPerCompany()
    begin
        // Code for company related operations
    end;

    trigger OnInstallAppPerDatabase()
    begin
        // Code for database related operations
    end;
}
```

TIP

Use the shortcuts `tcodunit` and `ttrigger` to create the basic structure for the codeunit and trigger.

Get information about an extension

Each extension version has a set of properties that contain information about the extension, including: AppVersion, DataVersion, Dependencies, Id, Name, and Publisher. This information can be useful when installing. For example, one of the more important properties is the `DataVersion` property, which tells you what version of data you are dealing with. These properties are encapsulated in a `ModuleInfo` data type. You can access these properties by through the `NAVApp.GetCurrentModuleInfo()` and `NAVAPP.GetModuleInfo()` methods.

Install codeunit example

This example uses the `OnInstallAppPerDatabase()` trigger to check whether the data version of the previous extension version is compatible for the upgrade.

```
codeunit 50100 MyInstallCodeunit
{
    Subtype=Install;

    trigger OnInstallAppPerDatabase();
    var
        myAppInfo : ModuleInfo;
    begin
        NavApp.GetCurrentModuleInfo(myAppInfo); // Get info about the currently executing module

        if myAppInfo.DataVersion = Version.Create(0,0,0,0) then // A 'DataVersion' of 0.0.0.0 indicates a
            'fresh/new' install
            HandleFreshInstall
        else
            HandleReinstall; // If not a fresh install, then we are Re-installing the same version of the
extension
        end;

        local procedure HandleFreshInstall();
        begin
            // Do work needed the first time this extension is ever installed for this tenant.
            // Some possible usages:
            // - Service callback/telemetry indicating that extension was installed
            // - Initial data setup for use
        end;

        local procedure HandleReinstall();
        begin
            // Do work needed when reinstalling the same version of this extension back on this tenant.
            // Some possible usages:
            // - Service callback/telemetry indicating that extension was reinstalled
            // - Data 'patchup' work, for example, detecting if new 'base' records have been changed while you
have been working 'offline'.
            // - Setup 'welcome back' messaging for next user access.
        end;
    }
}
```

See Also

[Developing Extensions](#)

[Getting Started with AL](#)

[How to: Publish and Install an Extension](#)

[Converting Extensions V1 to Extensions V2](#)

[Building Your First Sample Extension With Extension Objects, Install Code, and Upgrade Code](#)

Upgrading Extensions V2

5/28/2019 • 5 minutes to read

This article provides information about how to make a newer version of extension upgrade available on tenants. The first phase of this process is to develop the extension for upgrading, which means adding code to upgrade data from the previous extension version. Once you have the upgrade code in place, you can publish and synchronize the new version, and then run the data upgrade.

NOTE

An *upgrade* is defined as enabling an extension that has a greater version number, as defined in the app.json file, than the current installed extension version.

Developing an extension for upgrading

When developing a new extension version, you must consider the data from the previous version, and any modifications that must be applied to the data to make it compatible with the current version. For example, it could be that the new version adds a new field that needs default values set for existing records or the new version adds new tables that must be linked to existing records. To address this type of data handling, you must write upgrade code for the extension version.

If there are no data changes between the versions of your extension, then you do not need to write upgrade code. All data that is not modified by upgrade code will automatically be available when the process completes.

Writing upgrade code

You write upgrade logic in an upgrade codeunit, which is a codeunit whose [SubType property](#) is set to **Upgrade**. An upgrade codeunit supports several system triggers on which you can add data upgrade code. These triggers are invoked when you run the data upgrade process on the new extension.

The upgrade codeunit becomes an integral part of the extension and can be modified as needed for subsequent versions. You can have more than one upgrade codeunit. However, be aware that although there is a set order to the sequence of the upgrade triggers, there is no guarantee on the order of execution of the different codeunits. If you do use multiple upgrade units, make sure that they can run independent of each other.

Upgrade triggers

The following tables describes the upgrade triggers and lists them in the order in which they are invoked.

TRIGGER	DESCRIPTION	FAILS THE UPGRADE ON ERROR
OnCheckPreconditionsPerCompany() and OnCheckPreconditionsPerDatabase()	Used to check that certain requirements are met in order to run the upgrade.	Yes
OnUpgradePerCompany() and OnUpgradePerDatabase()	Used to perform the actual upgrade.	Yes
OnValidateUpgradePerCompany() and OnValidateUpgradePerDatabase()	Used to check that the upgrade was successful.	Yes

PerCompany triggers are run once for each company in the database, where each trigger is executed within its own system session for the company.

`PerDatabase` triggers are run once in the entire upgrade process, in a single system session that does not open any company.

NOTE

These triggers are also available in upgrade codeunits for the base application, not just for extensions.

Upgrade codeunit syntax

The following code illustrates the basic syntax and structure of an upgrade codeunit:

```
codeunit [ID] [NAME]
{
    Subtype=Upgrade;

    trigger OnCheckPreconditionsPerCompany()
    begin
        // Code to make sure company is OK to upgrade.
    end;

    trigger OnUpgradePerCompany()
    begin
        // Code to perform company related table upgrade tasks
    end;

    trigger OnValidateUpgradePerCompany()
    begin
        // Code to make sure that upgrade was successful for each company
    end;
}
```

TIP

Use the shortcuts `tcodeunit` and `ttrigger` to create the basic structure for the codeunit and trigger.

Get information about an extension

Each extension version has a set of properties that contain information about the extension, including:

`AppVersion`, `DataVersion`, `Dependencies`, `Id`, `Name`, and `Publisher`. This information can be useful when upgrading.

The `AppVersion` is one of the available properties and it's value differs depending on the context of the code being run:

- Normal operation: `AppVersion` represents the value of the currently installed extension.
- Installation code: `AppVersion` represents the version of the extension we are trying to install.
- Upgrade code: `AppVersion` represents the version of the extension that we are upgrading to (e.g. 'newer' version).

Another one of the more important properties is the `DataVersion` property, that represents the value of most recently installed/uninstalled/upgraded version of the extension, meaning that it reflects the most recent version of the data on the system, be that from the currently installed, or a previously uninstalled extension. The

`DataVersion` property value differs depending on the context of the code being run:

- Normal operation: `DataVersion` represents the version of the currently installed extension, in which case it is identical to the `AppVersion` property.
- Installation code:

- Reinstallation (applying the same version): `DataVersion` represents the version of the extension we are trying to install (identical to the `AppVersion` property).
- New installation: `DataVersion` represents the value of '0.0.0.0' which is used to indicate that there is no data.
- Upgrade code:
 - The version of the extension we are upgrading from. Either what was last uninstalled, or what is currently installed.

All of these properties are encapsulated in a `ModuleInfo` data type. You can access these properties through the `NAVApp.GetCurrentModuleInfo()` and `NAVApp.GetModuleInfo()` methods.

Upgrade codeunit example

This example uses the `OnCheckPreconditionsPerDatabase()` trigger to check whether the data version of the previous extension version is compatible for the upgrade before restoring the archived data of the old extension.

```
codeunit 70000001 MyUpgradeCodeunit
{
    Subtype=Upgrade;

    trigger OnCheckPreconditionsPerDatabase();
    var
        myInfo : ModuleInfo;
    begin
        if NavApp.GetCurrentModuleInfo(myInfo) then
            if myInfo.DataVersion = Version.Create(1, 0, 0, 1) then
                error('The upgrade is not compatible');
        end;

        trigger OnUpgradePerDatabase()
        begin
            NavApp.RestoreArchiveData(Database::"TableName");
        end;
    end;
}
```

Running the upgrade for the new extension version

To upgrade to the new extension version, you use the [Sync-NAVApp](#) and [Start-NAVAppDataUpgrade](#) cmdlets of the Dynamics NAV Administration Shell to synchronize table schema changes in the extension with the SQL database and run the data upgrade code.

1. Publish the new extension version. For simplicity, this example assumes the extension is not signed, which is not allowed with Dynamics 365 and is not recommended with an on-premise production environment.

```
Publish-NAVApp -ServerInstance DynamicsNAV -Path .\ProwareStuff_1.7.1.0.app -SkipVerification
```

This validates the extension syntax against server instance, and stages it for synchronizing.

2. Synchronize the new extension version with the database.

```
Sync-NAVApp -ServerInstance DynamicsNAV -Name ProwareStuff -Version 1.7.1.0
```

This synchronizes the database with any table schema changes in the extension; it adds the tables from the extension to the tenant.

3. Run a data upgrade.

```
Start-NAVAppDataUpgrade -ServerInstance DynamicsNAV -Name ProswareStuff -Version 1.7.1.0
```

This runs the upgrade logic that is defined by the upgrade codeunits in the extension. This will uninstall the current extension version, and enable the new version instead.

See Also

[Developing Extensions](#)

[Getting Started with AL](#)

[How to: Publish and Install an Extension](#)

[Converting Extensions V1 to Extensions V2](#)

[Sample Extension](#)

Publishing and Installing an Extension v2.0

5/24/2019 • 3 minutes to read

To make your extension available to tenant users requires three basic tasks: publish the extension package to the Dynamics 365 Business Central server instance, synchronize the extension with the tenant database, and install the extension on the tenant.

NOTE

This article describes how to publish and install the first version of an extension. If you want to publish an install newer version of an extension, see [Upgrading Extensions](#).

Publish and synchronize an extension

Publishing an extension to a Dynamics 365 Business Central server instance adds the extension to the application database that is mounted on the server instance, making it available for installation on tenants of the server instance. Publishing updates internal tables, compiles the components of the extension behind-the-scenes, and builds the necessary metadata objects that are used at runtime.

Synchronizing an extension updates the database schema of the tenant database with the database schema that is defined by the extension objects. For example, if a table or table extension is included in the extension, then the respective full or companion table is created in the tenant database.

To publish and synchronize an extension

1. Start the Dynamics NAV Administration Shell.
2. To publish the extension, run the [Publish-NAVApp cmdlet](#).

The cmdlet takes as parameters the Dynamics 365 Business Central service instance that you want to install to and the .app package file that contains the extension. The following example publishes the extension **MyExtension.app** to the **YourDynamicsNAVServer** instance.

```
Publish-NAVApp -ServerInstance YourDynamicsNAVServer -Path ".\MyExtension.app"
```

3. To synchronize the schema of a tenant database to the extension, run the [Sync-NavApp cmdlet](#).

The following example synchronizes the extension **MyExtension** with version number 1.0.0.0:

```
Sync-NavApp -ServerInstance YourDynamicsNAVServer -Name ExtensionName -Version 1.0.0.0 -Tenant  
TenantID
```

Replace `TenantID` with the tenant ID of the database. If you do not have a multitenant server instance, use `default` or omit this parameter.

The extension can now be installed on tenants.

Install an extension

After you publish and synchronize an extension, you can install it on tenants to enable the extension and make it available to users in the client. Installing an extension can be done from the Dynamics 365 client or Dynamics

NOTE

Installing an extension will run any installation code that is built-in to the extension. Installation code could, for example, perform operations like populating empty records with data, service callbacks and telemetry, version checks, and messages to users. For more information, see [Writing Extension Install Code](#).

To install an extension by using Dynamics NAV Administration Shell

1. Start the Dynamics NAV Administration Shell.
2. To install the extension on one or more tenants, use the `Install-NAVApp` cmdlet.

The following example installs the extension **My Extension** for Tenant1 and Tenant3. In single-tenant deployments, you either specify `default` as the tenant ID, or you omit the `-Tenant` parameter.

```
Install-NAVApp -ServerInstance YourDynamicsNAVServer -Name "My Extension" -Tenant Tenant1, Tenant3
```

To install an extension by using the client

1. In Dynamics 365 Business Central, use search to open the **Extension Management** page.

In the **Extension Management** window, you can view the extensions that are published to your server. For each extension, you can see the current installation status.

2. Choose an extension to see additional information and to install the extension.
3. Review and accept the license agreement.
4. Choose the **Install** button to install the extension.

See Also

[Unpublishing and Uninstalling Extensions](#)

[Developing Extensions](#)

Upgrading AppSource Apps in Production

5/21/2019 • 2 minutes to read

When an updated version of an AppSource app becomes the active version in the Dynamics 365 Business Central service, tenants do not automatically get this updated version. This upgrade must be done manually.

To upgrade an AppSource app

Follow the steps below to update a tenant to the latest version of any AppSource app.

1. Login to the Dynamics 365 Business Central Web client.
2. In the **Tell Me** box, enter **Extension Management**, and then choose the related link.
3. Locate the app that you want to update.
4. Under **Manage**, choose **Uninstall**.
5. When the uninstall is complete, choose **Install** to reinstall the app.
This will now update the app to the latest available version

For Dynamics 365 Business Central "Major" releases (April and October), every app installed on your tenant will be updated automatically to the latest available version in our service.

See Also

[Developing Extensions](#)

[Getting Started with AL](#)

[How to: Publish and Install an Extension](#)

Signing an APP Package File

5/21/2019 • 3 minutes to read

Code signing is a common practice for many applications. It is the process of digitally signing a file to verify the author and that the file has not been tampered with since it was signed. The signature of the APP package file is verified during the publishing of the extension using the `Publish-NAVApp` cmdlet. For more technical information on signing, see [Authenticode](#).

NOTE

If you want to publish an unsigned extension package in your on-premise environment, you need to explicitly state that by using the `-SkipVerification` parameter on the `Publish-NAVApp` cmdlet. An extension without a valid signature will not be published on AppSource.

The signing of an APP package file must be performed on a computer that has Dynamics 365 Business Central installed. If you use a Dynamics 365 Business Central Docker image for your development environment, that environment will meet this requirement. You must also have the certificate that will be used for signing on the computer. The certificate must include code signing as the intended purpose. It is recommended that you use a certificate purchased from a third-party certificate authority.

IMPORTANT

If you publish the extension as an app on AppSource, the APP package file must be signed using a certificate purchased from a Certification Authority that has its root certificates in Microsoft Windows. You can obtain a certificate from a range of certificate providers, including but not limited to GoDaddy, DigiCert, and Symantec, see the image below.

	Issued To	Issued By	Expiration Date	Intended Purposes	Friendly Name	Status
Console Root						
Certificates (Local Computer)						
Personal						
Trusted Root Certification Authorities						
Certificates	AddTrust External CA Root	AddTrust External CA Root	5/30/2020	Server Authenticati...	The USERTrust Net...	
	Baltimore CyberTrust Root	Baltimore CyberTrust Root	5/12/2025	<All>	<None>	
	Baltimore CyberTrust Root	Baltimore CyberTrust Root	5/12/2025	Server Authenticati...	DigiCert Baltimore ...	
	Certum CA	Certum CA	6/11/2027	Server Authenticati...	Certum	
	Certum Trusted Network CA	Certum Trusted Network CA	12/31/2029	Server Authenticati...	Certum Trusted Net...	
Enterprise Trust	Class 2 Primary CA	Class 2 Primary CA	7/6/2019	Secure Email, Serve...	CertPlus Class 2 Pri...	
Intermediate Certification Authorities	Class 3 Public Primary Certificat...	Class 3 Public Primary Certificatio...	8/1/2028	Server Authenticati...	VeriSign Class 3 Pu...	
Trusted Publishers	COMODO RSA Certification Au...	COMODO RSA Certification Auth...	1/18/2038	Server Authenticati...	COMODO SECURE™	
Untrusted Certificates	Copyright (c) 1997 Microsoft C...	Copyright (c) 1997 Microsoft Corp.	12/30/1999	Time Stamping	Microsoft Timesta...	
Third-Party Root Certification Authorities						
Trusted People						

Steps for signing your .app file

1. Prepare your computer for signing.
2. Make sure that you sign the .app file on a computer that has Dynamics 365 Business Central installed.
3. Copy the certificate that you purchased from a third-party certificate authority to a folder on the computer. The example uses a pfx version of the certificate. If the certificate you purchased is not in a pfx format, create a [PFX file](#). The file path for the sample command is `C:\Certificates\MyCert.pfx`. (Optionally, create your own certificate for local test or development purposes using the [Self-signed certificate](#) information).
4. Install a signing tool such as [SignTool](#) or [SignCode](#) to the computer. The sample command will use SignTool.
5. Copy your extensions .app file to the computer if it is not already on the computer. The file path for the sample command is `C:\NAV\Proseware.app`.
6. Run the command to sign the .app file.
7. The following example signs the Proseware.app file with a time stamp using the certificate in the password-protected MyCert.pfx file. The command is run on the computer that was prepared for the signing. Once the command has been run, the Proseware.app file has been modified with a signature. This file is then used when

publishing the extension.

```
SignTool sign /f C:\Certificates\MyCert.pfx /p MyPassword /t  
http://timestamp.verisign.com/scripts/timestamp.dll "C:\NAV\Proseware.app"
```

IMPORTANT

It is recommended to use a time stamp when signing the APP package file. A time stamp allows the signature to be verifiable even after the certificate used for the signature has expired. For more information, see [Time Stamping Authenticode Signatures](#). Depending on the certification authority, you may need to acquire a specific certificate in order to time stamp, an [Extended Validation](#) certificate from DigiCert for example.

Self-signed certificate

For testing purposes and on-premise deployments, it is acceptable to create your own self-signed certificate using the [New-SelfSignedCertificate](#) cmdlet in PowerShell on Windows 10 or [MakeCert](#).

The following example illustrates how to create a new self-signed certificate for code signing:

```
New-SelfSignedCertificate -Type CodeSigningCert -Subject "CN=ProsewareTest"
```

The following MakeCert command is used to create a new self-signed certificate for code signing:

```
Makecert -sk myNewKey -n "CN=Prosewaretest" -r -ss my
```

See Also

[Getting Started with AL](#)

[Keyboard Shortcuts](#)

[AL Development Environment](#)

Deploying a Tenant Customization

3/31/2019 • 2 minutes to read

When you have finished developing and testing your tenant customization, you must deploy the extension (.app file) containing the customization to your customer's production tenant. You must be able to log into the customer's tenant as a user with permissions to the **Extension Management** page to complete the deployment.

Use the **Upload Extension** action to deploy the extension. The extension can be deployed for the current version or for the next version of the service. In most cases it is sufficient to select the current version, unless you have developed the extension specifically for the next version.

NOTE

When you deploy an .app file for the next version, the extension will be queued up to be deployed as part of the customer's tenant being upgraded to the next version. You can typically use this in a situation where you have built an upgrade of the extension to work with the next version.

The extension you are deploying could be the initial release of the customization or an upgrade to a previous version. You must use the same steps for uploading a new extension or an extension upgrade. The service will determine if the extension needs to be upgraded based on the extension's app ID and version.

IMPORTANT

If you are developing and deploying an extension as an update to a previously deployed extension, you must keep the app ID the same and increase the version to successfully upgrade the extension to the new version.

The platform metadata requires that extensions be unique across all tenants, based on the package ID, app ID, name, publisher, and version. For example, you successfully deployed an extension on a tenant. You then recompiled the extension's source code so that a new extension package file was created with a different package ID than the original. If you try to upload this extension on a different tenant, you will get the error

An extension with same App ID and version has already been uploaded. Resolve and deploy again. When developing a per-tenant extension from the same source code as an extension that is already deployed on a tenant, we recommend that you adjust the App ID, Name, Publisher, and Version of the extension to maintain uniqueness.

These parameters are defined in the app.json file of the extension. For more information, see [JSON files](#).

Steps for deploying your .app file

1. Log into your customer's Dynamics 365 Business Central tenant.
2. Open the **Extension Management** page.
3. Choose the **Upload Extension** action.
4. Select the browse button to select the .app file to upload. Browse to and select the extension's .app file.
5. Select if you want to deploy for the current version (most common) or next version. Select the language for the deployment.
6. Choose the **Deploy** button.
7. The extension will be deployed in the background.

To check the status of the deployment, choose **Deployment Status** and then view the status of the extension deployment. Select the row to see additional details.

In the deployment status details there is a **Refresh** button in the actions you must press to retrieve the most recent status and details.

8. When the extension has been successfully deployed, choose the **Refresh** button to see the new extension in the list of installed extensions.

See Also

[Getting Started with AL](#)

[AL Development Environment](#)

[FAQ for Developing in AL](#)

[Using Designer](#)

Extending Application Areas

5/24/2019 • 5 minutes to read

Application areas represents a feature in the system that offers developers, administrators, and users the ability to define differentiated user experiences.

Application areas are mapped to controls to show or hide them on page objects to enable more or fewer business scenarios.

Extending application areas and the experience tier

In this example you will:

- Add a new application area in the **Application Area Setup** table.
- Enable the application area in the **OnInstallAppPerCompany** trigger.
- Extend the experience tier in the **OnGetExperienceAppArea** event.
- Modify the experience tier (optional).
- Validate the application area in the **OnValidateApplicationAreas**.

IMPORTANT

The code used in this example is still under active development and might be subject to change in the future.

The following example extends the **Customer List** page. The field **ExampleField** is added and it is followed by a series of properties. The **ApplicationArea** property sets the application areas that apply to the control and in this code, **ExampleAppArea** is assigned to it.

IMPORTANT

If your extension fails to use **ApplicationArea** in any controls or actions, they will not be visible when you use an experience tier.

The **OnOpenPage** trigger will display the message only if **ApplicationArea** is enabled.

```

pageextension 50100 CustomerListExt extends "Customer List"
{
    layout
    {
        addafter(Name)
        {
            field(ExampleField; "Name 2")
            {
                Caption = 'Example Field';
                ApplicationArea = ExampleAppArea;
                ToolTip = 'This is a field added by an example extension';
                Importance = Promoted;
            }
        }
    }

    trigger OnOpenPage()
    var
        EnableExampleExtension : Codeunit "Enable Example Extension";
    begin
        if EnableExampleExtension.IsExampleApplicationAreaEnabled() then
            Message('App published: Example Extension');
    end;
}

```

Adding an application area

To add an application area, the **Application Area Setup** table must be extended. A new boolean field is added and the name of this field will be used in the attribute that you want to be tagged with this application area. This particular case, in the code below, is an exception, because space is used inside it. Usually, spaces are omitted in the application area attribute. At this point, the extension has an application area but it still needs to be enabled.

```

tableextension 50100 "Application Area Setup" extends "Application Area Setup"
{
    fields
    {
        // Spaces in field name are omitted in the ApplicationArea attribute
        // e.g. ApplicationArea = ExampleAppArea;
        field(50100;"Example App Area";Boolean)
        {
        }
    }
}

```

The codeunit **Install Example Extension** is of the subtype **Install** and it enables the application area inside the **OnInstallAppPerCompany** trigger.

```

codeunit 50101 "Install Example Extension"
{
    Subtype = Install;

    trigger OnInstallAppPerCompany()
    var
        EnableApplicationArea : Codeunit "Enable Example Extension";
    begin
        if(EnableApplicationArea.IsExampleApplicationAreaEnabled()) then
            exit;

        EnableApplicationArea.EnableExampleExtension();

        // Add your code here
    end;
}

```

The registration of the application area inside an experience tier is made inside the **OnGetEssentialExperienceAppArea**. There are different versions of this event, one for each experience tier and in this case, Essential is chosen. This will make the extension visible inside the Essential experience and the event exposes an **Application Area Setup** temporary record; **TempApplicationAreaSetup**, to the **Application Area Setup** table. At this point, to enable the application area, this must be set to true.

NOTE

This event is important because it is called every single time an experience tier is reset, which can happen because of many reasons.

Another thing that is possible inside these methods is to modify the experience tier. You can also modify other application areas, such as creating an extension that extends the Fixed Assets functionality. By subscribing to **OnValidateApplicationAreas**, the application area inside an experience tier is validated.

OnValidateApplicationAreas is guaranteed to be executed after the events in the **OnGet*ExperienceAppArea** family. The validation is necessary in the presence of extensions concurrently manipulating the same application areas.

In case a needed application area is not enabled, the suggested action is to show an error and turn off the extension to avoid unintended behavior. However, if the functionality controlled by this application area is of secondary importance and its loss does not affect the rest of the extension, it is also appropriate to keep the extension enabled.

```

codeunit 50100 "Enable Example Extension"
{
    // Extend and modify Essential experience tier with "Example App Area"
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"Application Area Mgmt.",
    'OnGetEssentialExperienceAppAreas', '', false, false)]
    local procedure RegisterExampleExtensionOnGetEssentialExperienceAppAreas(var TempApplicationAreaSetup:
    Record "Application Area Setup" temporary)
    begin
        TempApplicationAreaSetup."Example App Area" := true;
        // Modify other application areas here
    end;

    // Validate that application areas needed for the extension are enabled
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"Application Area Mgmt.", 'OnValidateApplicationAreas',
    '', false, false)]
    local procedure VerifyApplicationAreasOnValidateApplicationAreas(ExperienceTierSetup: Record "Experience
    Tier Setup"; TempApplicationAreaSetup: Record "Application Area Setup" temporary)
    begin
        if ExperienceTierSetup.Essential then
            if not TempApplicationAreaSetup."Example App Area" then
                Error('Example App Area should be part of Essential in order for the Example Extension to
    work.');
```

Adding Advanced application area to the Essentials and Premium experiences using an extension

If you are familiar with Dynamics NAV you will have noticed that Dynamics 365 Business Central is not exposing all the controls/actions that you find in Dynamics NAV. These controls have been hidden so far by using the application area **Advanced**, which is not assigned to any experiences. For more information, see [Frequently Asked Questions](#).

Most of these fields will become available/visible soon, but until then you will have to create an extension to get (almost) the same experience as you have in Dynamics NAV. See the [example](#) below.

IMPORTANT

Adding the application area **Advanced** to the experience will mean that you lose some of the simplification made to pages. For example, you will see more actions duplicated on many pages, compared to Business Central where the experience is intended to be simpler than in Dynamics NAV. You must also consider that we plan to re-tag the **Advanced** actions/controls and add them to the **Essentials** and/or **Premium** experiences in a future release.

To enable Advanced in an extension

Depending on which experience you want to enable **Advanced** for you can subscribe to

`OnGetEssentialExperienceAppAreas` Or `OnGetPremiumExperienceAppAreas` . If you have defined your own experience you must subscribe to `OnSetExperienceTier` .

The experiences are additive so you only need to subscribe to one of the events. For example, to enable **Essentials** and **Premium** experiences you only need to subscribe to `OnGetEssentialExperienceAppAreas` .

```
codeunit 50102 EnableAdvancedApplicationArea
{
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"Application Area Mgmt. Facade",
    'OnGetEssentialExperienceAppAreas','', false, false)]
    local procedure EnableAdvancedApplicationAreaOnGetEssentialExperienceAppAreas(var TempApplicationAreaSetup
    : record 9178 temporary)
    begin
        TempApplicationAreaSetup.Advanced := true
    end;

    [EventSubscriber(ObjectType::Codeunit, Codeunit::"Application Area Mgmt. Facade",
    'OnGetPremiumExperienceAppAreas','', false, false)]
    local procedure EnableAdvancedApplicationAreaOnGetPremiumExperienceAppAreas(var TempApplicationAreaSetup :
    record 9178 temporary)
    begin
        TempApplicationAreaSetup.Advanced := true
    end;

    [EventSubscriber(ObjectType::Codeunit, Codeunit::"Application Area Mgmt. Facade", 'OnSetExperienceTier','',
    false, false)]
    local procedure EnableAdvancedApplicationAreaOnSetExperienceTier(ExperienceTierSetup : record 9176;var
    TempApplicationAreaSetup : record 9178 temporary;var ApplicationAreasSet : boolean)
    begin
        TempApplicationAreaSetup.Advanced := true
    end;
}
```

Application areas advantages and disadvantages

If you decide to code application areas as an extension, there are some aspects that must be considered.

Application areas enable hiding entire business scenarios and you can have the same code base, which makes it possible to quickly modify the UI for different business scenarios or audiences. However, tagging errors as missing tags or incorrect tags will occur and every single control will need to be tagged.

See Also

[ApplicationArea Property](#)

[ApplicationArea Method](#)

[AccessByPermission Property](#)

[Properties](#)

Extending Item Charge Distribution Methods

3/31/2019 • 5 minutes to read

To ensure correct valuation, your inventory items must carry any added costs, such as freight, physical handling, insurance, and transportation that you incur when purchasing or selling the items.

Users can add these costs by adding a Charge (Item) line to the involved purchase or sales document. For more information, see [Use Item Charges to Account for Additional Trade Costs](#) in application help.

Item charges are distributed over other item lines in the document according to a distribution method. Dynamics 365 Business Central offers four distribution methods out of the box: **Equally**, **By Amount**, **By Weight**, and **By Volume**. This article explains how to remove or add item charge distribution methods. The article describes the method for purchases. The steps are similar for sales, except the events are located in codeunit 5807, **Item Charge Assgnt. (Sales)**.

To enable extension of item charges distribution methods, two events have been added to codeunit 5805, **Item Charge Assgnt. (Purch.)**. The work consists of the following two tasks:

1. In the **OnBeforeShowSuggestItemChargeAssignStrMenu** event, you manipulate the options that are presented to users. You can remove, add, and change the order of the options.
 - Keep in mind that other extensions may also manipulate the options.
 - You should not assume that an option will exist, nor should you write code that may remove an option added by another extension.
2. In the **OnAssignItemCharges** event, you distribute the item charge amount over the item lines according to your new distribution method.
 - You must verify that the option selected by the user is your new option. If it is not, then exit without taking action.
 - When you have distributed the amount over the lines, you must set the `ItemChargesAssigned` boolean to true. If you do not set this boolean to true, an error will occur.

The following procedures show how to extend the item charges distribution methods:

1. Add a new option to the item charges distribution methods in the **OnBeforeShowSuggestItemChargeAssignStrMenu** event.
2. Add a new distribution method for item charges.
3. Call the new distribution method in the **OnAssignItemCharges** event.

The procedures are based on an example where the **By Fairy Dust** option is added to the string menu (STRMENU) and added to the CASE statement.

NOTE

To complete this example you will have to add a new field **Fairy Dust** to the **Purchase Line** table and other relevant tables and pages.

To add a new option to the item charges distribution methods

Create a new codeunit and add an event subscriber to the **OnBeforeShowSuggestItemChargeAssignStrMenu** event.

```

codeunit 50100 "Item Ch. Assign by Fairy Dust"
{
    var
        ByFairyDustTok: Label 'By Fairy Dust';

    local procedure AssignByFairyDustMenuText(): Text
    begin
        exit(ByFairyDustTok)
    end;

    [EventSubscriber(ObjectType::Codeunit, Codeunit::"Item Charge Assgnt. (Purch.)",
    'OnBeforeShowSuggestItemChargeAssignStrMenu', '', false, false)]
    local procedure AddByFairyDustOnBeforeShowSuggestItemChargeAssignStrMenu(PurchLine: Record "Purchase Line";
    var SuggestItemChargeMenuTxt: Text; var SuggestItemChargeMessageTxt: Text; var Selection: Integer)
    begin
        // if 'By Fairy Dust' is not in the menu options, add it at the end
        if StrPos(SuggestItemChargeMenuTxt, AssignByFairyDustMenuText) = 0 then begin
            SuggestItemChargeMenuTxt += ',' + AssignByFairyDustMenuText;
            // make the last option ('By Fairy Dust') the default selection
            Selection := StrLen(Delete(SuggestItemChargeMenuTxt, '=', Delete(SuggestItemChargeMenuTxt, '=',
            ', '))) + 1;
        end;
    end;
}

```

To add a new distribution method for item charges

In the new codeunit, add functions to distribute the charges over the item lines.

```

local procedure AssignByFairyDust(var ItemChargeAssignmentPurch: Record "Item Charge Assignment (Purch)";
Currency: Record Currency; TotalQtyToAssign: Decimal; TotalAmtToAssign: Decimal);
var
    TempItemChargeAssgntPurch: Record "Item Charge Assignment (Purch)" temporary;
    LineArray: array[2] OF Decimal;
    TotalFairyDust: Decimal;
    QtyRemaining: Decimal;
    AmountRemaining: Decimal;
begin
    // copy lines to temp variable and calculate total Fairy Dust
    repeat
        if not ItemChargeAssignmentPurch.PurchLineInvoiced then begin
            TempItemChargeAssgntPurch.Init();
            TempItemChargeAssgntPurch := ItemChargeAssignmentPurch;
            TempItemChargeAssgntPurch.Insert(false);
            GetItemValues(TempItemChargeAssgntPurch, LineArray);
            TotalFairyDust := TotalFairyDust + (LineArray[2] * LineArray[1]);
        end;
    until ItemChargeAssignmentPurch.Next = 0;

    if TempItemChargeAssgntPurch.Findset(true) then
        repeat
            // Calculate Fairy Dust to assign to the line
            GetItemValues(TempItemChargeAssgntPurch, LineArray);
            if TotalFairyDust <> 0 then
                TempItemChargeAssgntPurch."Qty. to Assign" :=
                    (TotalQtyToAssign * LineArray[2] * LineArray[1]) / TotalFairyDust + QtyRemaining
            else
                TempItemChargeAssgntPurch."Qty. to Assign" := 0;

            // Assign Fairy Dust to the line and calculate the remaining Fairy Dust to assign
            ItemChargeAssignmentPurch.Get(
                TempItemChargeAssgntPurch."Document Type",
                TempItemChargeAssgntPurch."Document No.",
                TempItemChargeAssgntPurch."Document Line No.",
                TempItemChargeAssgntPurch."Line No.");
        repeat

```

```

        ItemChargeAssignmentPurch."Qty. to Assign" := Round(TempItemChargeAssgntPurch."Qty. to Assign",
0.00001);

        ItemChargeAssignmentPurch."Amount to Assign" :=
            ItemChargeAssignmentPurch."Qty. to Assign" * ItemChargeAssignmentPurch."Unit Cost" +
AmountRemaining;
        AmountRemaining := ItemChargeAssignmentPurch."Amount to Assign" -
            Round(ItemChargeAssignmentPurch."Amount to Assign", Currency."Amount Rounding Precision");
        QtyRemaining := TempItemChargeAssgntPurch."Qty. to Assign" - ItemChargeAssignmentPurch."Qty. to
Assign";

        ItemChargeAssignmentPurch."Amount to Assign" :=
            Round(ItemChargeAssignmentPurch."Amount to Assign", Currency."Amount Rounding Precision");
        ItemChargeAssignmentPurch.Modify(false);

        until TempItemChargeAssgntPurch.Next = 0;
        TempItemChargeAssgntPurch.DeleteAll(false);
    end;

    procedure GetItemValues(TempItemChargeAssgntPurch: Record "Item Charge Assignment (Purch)" temporary; var
DecimalArray: Array[2] OF Decimal);
    var
        PurchaseLine: Record "Purchase Line";
        PurchRcptLine: Record "Purch. Rcpt. Line";
        ReturnShptLine: Record "Return Shipment Line";
        TransferRcptLine: Record "Transfer Receipt Line";
        SalesShptLine: Record "Sales Shipment Line";
        ReturnRcptLine: Record "Return Receipt Line";
    begin
        // Get the Fairy Dust for the line
        Clear(DecimalArray);
        with TempItemChargeAssgntPurch do
            case "Applies-to Doc. Type" of
                "Applies-to Doc. Type"::Order,
                "Applies-to Doc. Type"::Invoice,
                "Applies-to Doc. Type"::"Return Order",
                "Applies-to Doc. Type"::"Credit Memo":
                    begin
                        PurchaseLine.Get("Applies-to Doc. Type", "Applies-to Doc. No.", "Applies-to Doc. Line
No.");

                        DecimalArray[1] := PurchaseLine.Quantity;
                        DecimalArray[2] := PurchaseLine."Fairy Dust";
                    end;
                "Applies-to Doc. Type"::Receipt:
                    begin
                        PurchRcptLine.Get("Applies-to Doc. No.", "Applies-to Doc. Line No.");
                        DecimalArray[1] := PurchRcptLine.Quantity;
                        DecimalArray[2] := PurchRcptLine."Fairy Dust";
                    end;
                "Applies-to Doc. Type"::"Return Receipt":
                    begin
                        ReturnRcptLine.Get("Applies-to Doc. No.", "Applies-to Doc. Line No.");
                        DecimalArray[1] := ReturnRcptLine.Quantity;
                        DecimalArray[2] := ReturnRcptLine."Fairy Dust";
                    end;
                "Applies-to Doc. Type"::"Return Shipment":
                    begin
                        ReturnShptLine.Get("Applies-to Doc. No.", "Applies-to Doc. Line No.");
                        DecimalArray[1] := ReturnShptLine.Quantity;
                        DecimalArray[2] := ReturnShptLine."Fairy Dust";
                    end;
                "Applies-to Doc. Type"::"Transfer Receipt":
                    begin
                        TransferRcptLine.Get("Applies-to Doc. No.", "Applies-to Doc. Line No.");
                        DecimalArray[1] := TransferRcptLine.Quantity;
                        DecimalArray[2] := TransferRcptLine."Fairy Dust";
                    end;
                "Applies-to Doc. Type"::"Sales Shipment":
                    begin
                        SalesShptLine.Get("Applies-to Doc. No.", "Applies-to Doc. Line No.");

```

```

        DecimalArray[1] := SalesShptLine.Quantity;
        DecimalArray[2] := SalesShptLine."Fairy Dust";
    end;
end;
end;

```

To call the new distribution method

In the new codeunit, add a subscriber to the **OnAssignItemCharges** event.

```

[EventSubscriber(ObjectType::Codeunit, Codeunit::"Item Charge Assgnt. (Purch.)", 'OnAssignItemCharges', '',
false, false)]
local procedure AssignByFairyDustOnAssignItemCharges(SelectionTxt: Text; var ItemChargeAssignmentPurch:
Record "Item Charge Assignment (Purch)"; Currency: Record Currency; PurchaseHeader: Record "Purchase Header";
TotalQtyToAssign: Decimal; TotalAmtToAssign: Decimal; VAR ItemChargesAssigned: Boolean);
begin
    // if item charges are already assigned, exit
    if ItemChargesAssigned then
        exit;
    // if the user did not choose 'By Fairy Dust', exit
    if not (SelectionTxt = AssignByFairyDustMenuText) then
        exit;
    // assign item charges by fairy dust
    AssignByFairyDust(ItemChargeAssignmentPurch, Currency, TotalQtyToAssign, TotalAmtToAssign);
    // charges have been assigned
    ItemChargesAssigned := true;
end;

```

See Also

[Extending Application Areas](#)

Events in AL

3/31/2019 • 3 minutes to read

The use of events is a proven and established programming concept that can ease application upgrade and limit or even eliminate the need for code modifications in customized applications because of application platform changes.

You can use events to design the application to react to specific actions or behavior that occur. Events enable you to separate customized functionality from the application business logic. By using events in the application where customizations are typically made, you can lower the cost of code modifications and upgrades to the original application.

- Code modifications to customized functionality can be made without having to modify the original application.
- Changes to the original application code can be made with minimal impact on the customizations.

Events can be used for different purposes, such as generating notifications when certain behavior occurs or the state of an entity changes, distributing information, and integrating with external systems and applications. For example, in the CRONUS International Ltd. demonstration database, events are used extensively for workflow and Dynamics 365 for Sales integration.

The following table describes all the different event types:

EVENT TYPES	DESCRIPTION
BusinessEvent	Specifies the method to be business type event publisher.
IntegrationEvent	Specifies the method to be integration type event publisher.
Global	Global events are predefined system events.
Trigger	Trigger events are published by the runtime.

The process for implementing these events is slightly different. To learn about the different types, see [Event Types](#).

How events work

The basic principle is that you program events in the application to run customized behavior when they occur. Events in AL are modeled after Microsoft .NET Framework. There are three major participants involved in events: the *event*, a *publisher*, and a *subscriber*.

- An *event* is the declaration of the occurrence or change in the application. An event is declared by an AL method, which is referred to as an *event publisher function*. An event publisher method is comprised of a signature only and does not execute any code.
- A *publisher* is the object that contains event publisher method that declares the event. The publisher exposes an event in the application to subscribers, essentially providing them with a hook-up point in the application.

Publishing an event does not actually do anything in the application apart from making the event available for subscription. The event must be raised for subscribers to respond. An event is raised by adding logic to

the application that calls into the publisher to invoke the event (the event publisher method).

Partners or subsystems can then take advantage of the published event in their solutions. An ISV that delivers vertical solutions, and Microsoft itself, are the typical providers of published events.

Business and integration type events must be explicitly published and raised, which means that you must create event publisher functions and add them to objects manually. On the other hand, trigger events, which occur on table and page operations, are published and raised implicitly by the system at runtime. Therefore, no coding is required to publish them.

- A *subscriber* listens for and handles a published event. A subscriber is an AL method that subscribes to a specific event publisher method and includes the logic for handling the event. When an event is raised, the subscriber method is called and its code is run. A subscriber enables partners to hook into the core application functionality without having to do traditional code modifications. Any Dynamics 365 solution provider, which also includes Microsoft, can use event subscribers.

There can be multiple subscribers to a single event publisher method. However, a publisher has no knowledge of subscribers, if any. Subscribers can reside in different parts of the application than publishers.

How to implement events

Implementing events consists of the following tasks:

1. Publish the event.

For business and integration events, create and configure a method in an application object to be an event publisher method. For more information, see [Publishing Events](#). This is not required for trigger events because these are automatically published by the system.

2. Raise the event.

Add code that calls the event publisher method. For more information, see [Raising Events](#). This is not required for trigger events because these are raised automatically by the system.

3. Subscribe to the event.

At the consumer end, add one or more subscriber methods that subscribe to published events when they are raised. For more information, see [Subscribing to Events](#).

See Also

[Publishing Events](#)

[Raising Events](#)

[Subscribing to Events](#)

[Developing Extensions Using the New Development Environment](#)

Event Types

3/31/2019 • 7 minutes to read

Dynamics 365 Business Central supports different types of events for different purposes.

Business events

A business event is a custom event that is raised by AL code. It defines a formal contract that carries an implicit promise not to change in future releases. It is the expectation that business events are published by solution ISVs, including Microsoft.

Business events can be compared with publicly released APIs on which 3rd party solution provider develop integrations and additions. Therefore, the downstream cost of making changes to a business event implementation can be considerable for those who use the event in their applications. There may be some cases where changes are required; however, you should keep these to an absolute minimum.

Development considerations

A typical business event reflects changes in "state" with regards to a process. This makes them very well suited for workflow. An example of a business event could be when a sales order has been posted. It is important to note that business events should not be tied to the implementation-details, such as the tables or fields in which the data is stored. Preferably, the event publisher developer should be free to change the implementation, while still keeping the business event intact. To learn about the syntax and example on how to use the BusinessEvent type, see [BusinessEvent Attribute](#).

Business events should be documented with the solution, including the before-state and after-state of the events.

Integration events

An integration event is also a custom event that is raised by AL code, like a business event, except that it does not carry the same promise of not changing, nor does it have the restriction not to expose implementation details.

The main purpose of integration events is to enable the integration of other solutions with Dynamics 365 Business Central without having to perform traditional code modifications.

Development considerations

An integration event can be changed to a business event later. At which time, it must adhere to the same implied contract and commitment as any business event. It can also simply be designed-in hook points for external additions. To learn about the syntax and example on how to use the IntegrationEvent type, see [IntegrationEvent Attribute](#).

Global events

Global events are predefined system events that are automatically raised by various base application codeunits. For example, codeunit 40 **LoginManagement** includes several global method triggers, such as CompanyOpen, CompanyClose, and GetSystemIndicator. For most of these global method triggers, there are one or two global events: a before and after event. For example, there is an OnBeforeCompanyOpen event and an OnAfterCompanyOpen event. The global events are defined as integration event publishers by local methods in the following codeunits.

CODEUNIT ID	CODEUNIT NAME	EVENT
40	LoginInManagement	OnRoleCenterOpen
		OnAfterLogInEnd
		OnBeforeLogInStart
		OnBeforeCompanyOpen
		OnAfterCompanyOpen
		OnBeforeCompanyClose
		OnAfterCompanyClose
42	TextManagement	OnBeforeMakeTextFilter
		OnAfterMakeDateTimeFilter
		OnAfterMakeDateFilter
		OnAfterMakeTextFilter
		OnAfterMakeTimeFilter
42	CaptionManagement	OnAfterCaptionClassTranslate
44	ReportManagement	OnAfterGetPrinterName
		OnAfterHasCustomLayout
45	AutoFormatManagement	OnAfterAutoFormatTranslate
49	GlobalTriggerManagement	OnAfterGetGlobalTableTriggerMask
		OnAfterOnGlobalInsert
		OnAfterOnGlobalModify
		OnAfterOnGlobalDelete
		OnAfterOnGlobalRename
		OnAfterGetDatabaseTableTriggerSetup
		OnAfterOnDatabaseInsert
		OnAfterOnDatabaseModify
		OnAfterOnDatabaseDelete

CODEUNIT ID	CODEUNIT NAME	EVENT
		OnAfterOnDatabaseRename
		OnBeforeOnDatabaseInsert
		OnBeforeOnDatabaseModify
		OnBeforeOnDatabaseDelete
		OnBeforeOnDatabaseRename

Trigger events

Unlike business and integration events which must be programmed, trigger events are predefined events. Trigger events are published by the runtime and they cannot be raised programmatically. There are two types of trigger events: database trigger events and page trigger events.

NOTE

Trigger events do not appear as methods in AL for a table or page object.

Database trigger events

Trigger events are automatically raised by the system when it performs database operations on a table object, such as deleting, inserting, modifying, and renaming a record, as defined in a table. Trigger events are closely associated with the table triggers for database operations: OnDelete, OnInsert, OnModify, OnRename, and OnValidate (for fields). For each database operation, there is a "before" and "after" trigger event with a fixed signature.

Available Database Trigger Events

The following table describes the available database trigger events:

DATABASE TRIGGER EVENT WITH SIGNATURE	DESCRIPTION
<code>OnBeforeDeleteEvent(VAR Rec: Record, RunTrigger: Boolean)</code>	Executed before a record is deleted from a table.
<code>OnAfterDeleteEvent(VAR Rec: Record, RunTrigger: Boolean)</code>	Executed after a record is deleted from a table.
<code>OnBeforeInsertEvent(VAR Rec: Record, RunTrigger: Boolean)</code>	Executed before a record is inserted in a table.
<code>OnAfterInsertEvent(VAR Rec : Record, RunTrigger : Boolean)</code>	Executed after a record is inserted in a table.
<code>OnBeforeModifyEvent(VAR Rec : Record, VAR xRec : Record, RunTrigger : Boolean)</code>	Executed before a record is modified in a table.
<code>OnAfterModifyEvent(VAR Rec : Record, VAR xRec : Record, RunTrigger : Boolean)</code>	Executed after a record is modified in a table.
<code>OnBeforeRenameEvent(VAR Rec : Record, VAR xRec : Record, RunTrigger : Boolean)</code>	Executed before a record is renamed in a table.

DATABASE TRIGGER EVENT WITH SIGNATURE	DESCRIPTION
OnAfterRenameEvent(VAR Rec : Record, VAR xRec : Record, RunTrigger : Boolean)	Executed after a record is renamed in a table.
OnBeforeValidateEvent(VAR Rec : Record, VAR xRec : Record, RunTrigger : Boolean; CurrentFieldNo : Integer)	Executed before a field is validated when its value has been changed.
OnAfterValidateEvent(VAR Rec : Record, VAR xRec : Record, RunTrigger : Boolean, CurrentFieldNo : Integer)	Executed after a field is validated when its value has been changed.

The following table describes the parameters of the trigger events:

PARAMETER	TYPE	DESCRIPTION
<i>Rec</i>	Record	The table that raises the event.
<i>xRec</i>	Record	The table that raises the event.
<i>RunTrigger</i>	Boolean	Specifies whether to execute the code in the event trigger when it is invoked. If this parameter is true, the code will be executed. If this parameter is false, then the code is not executed.
<i>CurrentFieldNo</i>	Integer	The number of the field that raises the event.

Order of Event Execution

The relative order of execution of database trigger events, table triggers, and database operations is as follows:

ORDER	ITEM	EXAMPLE
1	Trigger event (before)	OnBeforeDeleteEvent
2	Table trigger	OnDelete
3	Global table trigger in codeunit	OnDatabaseDelete
4	Database operations	Delete the record
5	Trigger event (after)	OnAfterDeleteEvent

Page trigger events

Page Trigger events are raised automatically by the system when it performs certain operations in a page object. Page trigger events are closely associated with the standard page triggers, such as OnOpenPage, OnClosePage, and OnAction.

Available Page Trigger Events

The following table describes the available page trigger events:

TRIGGER EVENT WITH SIGNATURE	DESCRIPTION
<code>OnBeforeActionEvent(VAR Rec : Record)</code>	Executed before the OnAction Trigger , which is called when a user selects an action on the page.
<code>OnAfterActionEvent(VAR Rec : Record)</code>	Executed after the OnAction Trigger , which is called when a user selects an action on the page.
<code>OnAfterGetCurrRecordEvent(VAR Rec : Record)</code>	Executed after the OnAfterGetCurrRecord Trigger , which is called after the current record is retrieved from the table.
<code>OnAfterGetRecordEvent(VAR Rec : Record)</code>	Executed after the OnAfterGetRecord Trigger , which is called after the record is retrieved from the table but before it is displayed to the user.
<code>OnBeforeValidateEvent(VAR Rec : Record, VAR xRec : Record)</code>	Executed before the OnValidate (Page fields) Trigger , which is called when a field loses focus after its value has been changed.
<code>OnAfterValidateEvent(VAR Rec : Record, VAR xRec : Record)</code>	Executed after the OnValidate (Page fields) Trigger , which is called when a field loses focus after its value has been changed.
<code>OnClosePageEvent(VAR Rec : Record)</code>	Executed after the OnClosePage Trigger , which is called when page closes after the OnQueryClosePage trigger is executed.
<code>OnDeleteRecordEvent(VAR Rec : Record, VAR AllowDelete : Boolean)</code>	Executed after the OnDeleteRecord Trigger , which is called before a record is deleted from a table.
<code>OnInsertRecordEvent(VAR Rec : Record, BelowxRec : Boolean, VAR xRec : Record, VAR AllowInsert : Boolean)</code>	Executed after the OnInsertRecord Trigger , which is called before a record is inserted in a table.
<code>OnModifyRecordEvent(VAR Rec : Record, VAR xRec : Record, VAR AllowModify : Boolean)</code>	Executed after the OnModifyRecord Trigger , which is called before a record is modified in a table.
<code>OnNewRecordEvent(VAR Rec: Record, BelowxRec : Boolean, VAR xRec : Record)</code>	Executed after the OnNewRecord Trigger , which is called before a new record is initialized.
<code>OnOpenPageEvent(VAR Rec : Record)</code>	Executed after the OnOpenPage Trigger , which is called after a page is initialized and run.
<code>OnQueryClosePageEvent(VAR Rec : Record, VAR AllowClose : Boolean)</code>	Executed after the OnQueryClosePage Trigger , which is called as a page closes and before the OnClosePage Trigger executes.

The following table describes the parameters of the trigger events:

PARAMETER	TYPE	DESCRIPTION
<i>Rec</i>	Record	The table that used page that raises the event.
<i>xRec</i>	Record	The table that used page that raises the event.

PARAMETER	TYPE	DESCRIPTION
<i>AllowDelete</i>	Boolean	Specifies whether the OnDeleteRecord trigger call was successful and the record can be deleted. If this parameter is true, the code will be executed. If this parameter is false, then the code is not executed.
<i>AllowModify</i>	Boolean	Specifies whether the OnModifyRecord trigger call was successful and the record can be modified. If this parameter is true, the code will be executed. If this parameter is false, then the code is not executed.
<i>BelowxRec</i>	Boolean	Specifies whether the new record was inserted after the last record in the table (xRec).
<i>AllowClose</i>	Boolean	Specifies whether to the page can close. If this parameter is true, the code will be executed. If this parameter is false, then the code is not executed.

See Also

[Events in AL](#)

[Publishing Events](#)

[Raising Events](#)

[Subscribing to Events](#)

Publishing Events

5/3/2019 • 3 minutes to read

The first phase of implementing an event is publishing the event. Publishing an event exposes it in the application. This provides hook up points for subscribers to register to the event, and eventually handle the event if it is raised. An event is published by adding an AL method that is specifically set up as an event publisher.

- Business and integration events require that you manually create an event publisher method for each event that you want to publish. An event publisher method declares the event in the application and makes it available for subscription; however, it does not raise the event. After an event is published, you can raise it in your application, as needed, from where event subscribers can react and handle the event.
- Trigger events, on the other hand, do not require that you create publisher methods. Trigger events are predefined event publisher methods that are called automatically at runtime. This means that trigger events are readily available to subscribers by default.

Creating an event publisher method to publish business and integration events

You create an event publisher method the same way you create any method in AL, except that there are specific attributes that you set to make it an event publisher. Additionally, an event publisher method has the following requirements and restrictions that you must follow, otherwise you will not be able to compile your code changes:

- An event publisher method cannot include any code except comments.
- An event publisher method cannot have a return value, variables, or text constants.

The following procedure provides an outline of the tasks that are involved in creating an event publisher method for declaring an event.

To create an event publisher method

1. Decide where you want to include the event publisher method.

You can include an event publisher method in the AL code of any object type, such as codeunit, page, or table. You can create a new object or use an existing object.

IMPORTANT

If you include the event publisher method in a page object, the page must have a source table. Otherwise, you cannot successfully create an event subscriber method to subscribe to the event.

2. Add an AL method to the object.

If you do not want the event publisher to be raised from other objects than the one defining it, make it a local method by affixing it with `local`. The event still remains available to event subscribers from other objects.

You should give the method a name that has the format *On[Event]*, where *[Event]* is text that indicates what occurred, such as `OnAddressLineChanged`.

3. Decorate the method with either the [Integration attribute](#) or [Business attribute](#) as follows:

```
[IntegrationEvent(IncludeSender : Boolean, GlobalVarAccess : Boolean)]
```

or

```
[BusinessEvent(IncludeSender : Boolean)]
```

TIP

Use the `teventint` snippet for an integration event or the `teventbus` snippet for a business event to get started.

For more information about integration and business events, see [Event Types](#).

4. Add parameters to the method as needed.

You can include as many parameters of any type as necessary.

Make sure to expose enough information as parameters to enable subscriber methods to add value to the application. On the other hand, especially with business events, do not expose unnecessary parameters that may constrain you from changing or extending methodically in the future.

You can now add code to the application that raises the event by calling the event publisher method. You can also create subscriber methods that handle the event when it is raised.

Example

This example creates the codeunit **7000001 MyPublisher** to publish an integration event. The event is published by adding the global method called `OnAddressLineChanged`. The event takes a single text data type parameter.

NOTE

This example is part of a larger, simple scenario where when users change the address of a customer on the page **21 Customer Card**, you want to check that the address does not include a plus sign (+). If it does, you want to display a message. To accomplish this, you will publish an event that is raised when the **Address** field on **Customer Card** is changed, and add an event subscriber method to that includes logic that checks the address value and returns a message to the user if it contains a plus sign.

```
codeunit 7000001 MyPublishers
{
    [IntegrationEvent(false, false)]
    procedure OnAddressLineChanged(line : Text[100]);
    begin
        end;
}
```

The next step would be to raise this event in the application. To see an example for how this event is raised, go to [Raising Event Example](#).

See Also

[Raising Events](#)
[Subscribing to Events](#)
[Events Dynamics 365](#)

Raising Events

3/31/2019 • 2 minutes to read

After an event has been published by an event publisher method, you can modify the application to raise the event where it is needed. Subscribers of an event will not react on the event until it is raised in the application.

To raise an event, you add logic in AL code of the application to call the event publisher method that declares the event. The procedure for calling the event publisher method is the same as calling any other method in AL.

When the code that calls the event publisher method is run, all event subscriber methods that subscribe to the event are run. If there are multiple subscribers, the subscriber methods are run one at a time in no particular order. You cannot specify the order in which the subscriber methods are called.

If there are no subscribers to the published event, then the line of code that calls the event publisher method is ignored and not executed.

Snippet support

Typing the shortcut `teventsub` will create the basic event subscriber syntax when using the AL Language extension in Visual Studio Code.

TIP

Typing the keyboard shortcut `Ctrl + space` displays IntelliSense to help you fill in the attribute arguments and to discover which events are available to use.

Example

This example uses a page extension object **70000002 MyCustomerExt** to modify the page **21 Customer Card** so that an event is raised when a user changes the **Address** field. This example assumes that the event has already been published by the event publisher method `OnAddressLineChanged` in a separate codeunit called **70000001 MyPublishers**.

NOTE

This example is part of a larger, simple scenario where when users change the address of a customer on the page **21 Customer Card**, you want to check that the address does not include a plus sign (+). If it does, you want to display a message. To accomplish this, you will publish an event that is raised when the **Address** field on **Customer Card** is changed, and add an event subscriber method to that includes logic that checks the address value and returns a message to the user if it contains a plus sign.

In the code that follows, the page extension object modifies the `OnBeforeValidate` trigger of the **Customer Card** page to raise the event `OnAddressLineChanged` which includes the new value of the **Address** field.

```
pageextension 70000002 MyCustomerExt extends "Customer Card"
{
    layout
    {
        modify(Address)
        {
            trigger OnBeforeValidate();
            var
                Publisher: Codeunit 70000001;
            begin
                Publisher.OnAddressLineChanged(Address);
            end;
        }
    }
}
```

To learn about how the event used in this example is published, see [Publishing Events Example](#).

The next step would be to subscribe to the event to handle to condition. To see an example of how to subscribe to this event, see [Subscribing to Events Example](#).

See Also

[Publishing Events](#)

[Subscribing to Events](#)

[Events Dynamics 365](#)

Subscribing to Events

6/25/2019 • 4 minutes to read

To handle events, you design event subscribers. Event subscribers determine what actions to take in response to an event that has been raised. An event subscriber is an AL method that subscribes to, or listens for, a specific event that is declared by an event publisher method. The event subscriber includes code that defines the business logic to handle the event. When the published event is raised, the event subscriber is called and its code is run.

Subscribing to an event tells the runtime that the subscriber method must be called whenever the publisher method is run, either by code (as with business and integration events) or by the system (as with trigger events). The runtime establishes the link between an event raised by the publisher and its subscribers by looking for event subscriber methods.

There can be multiple subscribers to the same event from various locations in the application code. When an event is raised, the subscriber methods are run one at a time in no particular order. You cannot specify the order in which the subscriber methods are called.

Creating an event subscriber method

You create an event subscriber method just like other methods except that you specify properties that set up the subscription to an event publisher. The procedure is slightly different for database and page trigger events than business and integration events because business and integration events are raised by event publisher methods in application code. Trigger events are predefined system events that are raised automatically on tables and pages.

For an explanation about the different types, see [Event Types](#).

To create an event subscriber method

1. Decide which codeunit to use for the event subscriber method.

You can create a new codeunit or use an existing one.

2. Add an AL method to the codeunit.

We recommend that you give the method a name that indicates what the subscriber does, and has the format *[Action][Event]*. *[Action]* is text that describes what the method does and *[Event]* is the name of the event publisher method to which it subscribes.

3. Add code to the method for handling the event.
4. Decorate the event subscriber method with the [EventSubscriber attribute](#), and change accordingly.

```
[EventSubscriber(ObjectType::<Event Publisher Object Type>, <Event Publisher Object>, '<Published Event Name>', '<Published Event Element Name>', <SkipOnMissingLicense>, <SkipOnMissingPermission>)]
```

Set the arguments according to the following table. For optional arguments, if you do not want to set a value, use an empty value (`' '`). In this, the default value, if any, is used.

ARGUMENT	DESCRIPTION	OPTIONAL
----------	-------------	----------

ARGUMENT	DESCRIPTION	OPTIONAL
<code><Event Publisher Object Type></code>	Specify the type of object that publishes the event. This can be <code>Codeunit</code> , <code>Page</code> , <code>Report</code> , <code>Table</code> , Or <code>XMLPort</code> .	no
<code><Event Publisher Object></code>	Specify the object that publishes the event. You can set this to the ID, such as <code>50100</code> , or the recommended way is to use the object name by using the syntax <code><Object Type>::"<Object Name>"</code> , such as <code>Codeunit::"MyPublishers"</code> , or for database triggers <code>Database::"Customer"</code> .	no
<code><Published Event Name></code>	Specify the name of method that publishes the event in the object that is specified by the <code><Event Publisher Object></code> parameter.	no
<code><Published Event Element Name></code>	Specifies the table field that the trigger event pertains to. This argument only requires a value for database trigger events, that is, when the <code><Event Publisher Object Type></code> is set to <code>Table</code> and the <code><Published Event Name></code> argument is a validate trigger event, such as <code>OnAfterValidateEvent</code> .	no
<code><SkipOnMissingLicense></code>	Set to <code>true</code> to skip the event subscriber method call if the user's license does not cover the event subscriber codeunit. If <code>false</code> , an error is thrown and the code execution stops. <code>false</code> is the default.	yes
<code><SkipOnMissingPermission></code>	Set to <code>true</code> to skip the event subscriber method call if the user does not have the correct permissions the event subscriber codeunit. If <code>false</code> , an error is thrown and the code execution stops. <code>false</code> is the default.	yes

TIP

Use the `teventsub` snippet to get started and typing the keyboard shortcut `Ctrl + space` displays IntelliSense to help you fill the attribute arguments and to discover which events are available to use.

5. Optionally, set the codeunit's **EventSubscriberInstance** property to specify how the event subscriber

method will be bound to the instance of this codeunit.

For more information, see [EventSubscriberInstance Property](#).

Example 1

This example creates the codeunit **7000002 MySubscriber** to subscribe to an event that has been published by the event publisher method called `OnAddressLineChanged` in the codeunit **7000001 MyPublishers**. The event is raised by a change to the **Address** field on page **21 Customer Card**. This example assumes the following:

- The codeunit **7000001 MyPublishers** with the event publisher method `OnAddressLineChanged` already exists. To see how to do this, see [Publishing Event Example](#).
- The code for raising the `OnAddressLineChanged` event has been added to the **Customer Card** page. To see how to do this, see [Raising Event Example](#).

The following code creates a codeunit called **7000002 MySubscriber** that includes an event subscriber method, called `CheckAddressLine`. The method includes code for handling the published event.

```
codeunit 7000002 MySubscriber
{
    EventSubscriberInstance = StaticAutomatic;

    [EventSubscriber(ObjectType::Codeunit, Codeunit::"MyPublishers", 'OnAddressLineChanged', '', true, true)]
    procedure CheckAddressLine(line : Text[100]);
    begin
        if (STRPOS(line, '+') > 0) then begin
            MESSAGE('Cannot use a plus sign (+) in the address [' + line + ']');
        end;
    end;
}
```

Example 2

This example achieves the same as example 1, except it subscribes to the page trigger event `OnBeforeValidateEvent` on the `Address` field instead. By using the page trigger, you avoid creating an event publisher and adding code to raise the event because this is done automatically by the system.

```
codeunit 7000002 MySubscriber
{
    EventSubscriberInstance = StaticAutomatic;

    [EventSubscriber(ObjectType::Page, Page::"Customer Card", 'OnBeforeValidateEvent', 'Address', true, true)]
    local procedure CheckAddressLine(var Rec : Record Customer)
    begin
        if (STRPOS('Rec.Address', '+') > 0) then begin
            MESSAGE('Cannot use a plus sign (+) in the address [' + 'Address' + ']');
        end;
    end;
}
```

See Also

[Publishing Events](#)

[Raising Events](#)

[Event Types](#)

[Events in AL](#) [EventSubscriberInstance Property](#) [EventSubscriber Attribute](#)

Discoverability of Events

5/24/2019 • 2 minutes to read

You subscribe to events to extend application and interact with the base application and other extensions. This topic describes how to discover events that you can subscribe to without writing the code manually. Using the Event Recorder, you can record the events that are published and raised while performing the actions of your scenario. For example, record the events raised when you post a purchase order and identify the events that you need for your extension. You can retrieve the events in the form of AL snippet code and use them in Visual Studio Code directly.

Using the Event Recorder

You can launch the Event Recorder session from Dynamics 365 Business Central. It can be accessed in the following two ways:

- In Dynamics 365 Business Central, search for **Event Recorder**.
- In Visual Studio Code, use the `Ctrl+Shift+P` keys and select the **Open Event Recorder** command to open the Event Recorder page in the Dynamics 365 Business Central web client.

How to record Events

The following steps describe how to record events when you are on the Event Recorder page.

1. To record the current session, click the **Start** button located on the actions ribbon.
2. Perform all the actions that you want to record while the Event Recorder session is on. For example, post a purchase order.

TIP

When the Event Recorder session is started, all the actions are recorded including the search activities. Therefore, before starting the recorder, you can open two separate windows; one, to perform the actions of your scenario; and second, to start and stop the Event Recorder session.

3. After you have performed the actions of your scenario, navigate back to the Event Recorder page and click the **Stop** button to finish recording.
All the events raised while performing the actions of your scenario are recorded and can be viewed on the Event Recorder page as shown below.

Dynamics 365 Business Central Event Recorder										
HOME										
Start Stop										
Record Events										
VIEW - EVENT RECORDER										
CALL ORDER	EVENT TYPE	HIT COUNT	OBJECT TYPE	OBJECT NAME	EVENT NAME	ELEMENT NAME	CALLING OBJECT TYPE	CALLING OBJECT NAME	CALLING METHOD	GET AL Snippet
1	Trigger Event	1	Page	Purchase Order List	OnAfterGetRecordEvent					Get AL Snippet.
2	Trigger Event	1	Page	Purchase Order List	OnBeforeActionEvent	Post				Get AL Snippet.
3	Custom Event	1	Codeunit	Purch.-Post (Yes/No)	OnBeforeConfirmPost		Codeunit	Purch.-Post (Yes/No)	Code	Get AL Snippet.
4	Custom Event	1	Codeunit	Purch.-Post (Yes/No)	OnAfterConfirmPost		Codeunit	Purch.-Post (Yes/No)	Code	Get AL Snippet.
5	Custom Event	1	Codeunit	Purch.-Post	OnBeforePostPurchaseDoc		Codeunit	Purch.-Post	OnRun	Get AL Snippet.
6	Custom Event	1	Codeunit	Purch.-Post	OnBeforeValidatePostingAndDocument...		Codeunit	Purch.-Post	ValidatePostingAndDocumentDate	Get AL Snippet.
7	Custom Event	1	Codeunit	Purch.-Post	OnAfterCheckMandatoryFields		Codeunit	Purch.-Post	CheckMandatoryHeaderFields	Get AL Snippet.
8	Custom Event	1	Codeunit	Gen. Jnl.-Check Line	OnAfterDateNotAllowed		Codeunit	Gen. Jnl.-Check Line	DateNotAllowed	Get AL Snippet.
9	Custom Event	8	Codeunit	DimensionManagem...	OnCheckDimValueAllowed		Codeunit	DimensionManagem...	CheckDimValueAllowed	Get AL Snippet.
10	Custom Event	1	Table	Purchase Header	OnCheckPurchasePostRestrictions		Codeunit	Purch.-Post	CheckPostRestrictions	Get AL Snippet.
11	Custom Event	1	Codeunit	Purch.-Post	OnAfterCheckTrackingAndWarehouseFo...		Codeunit	Purch.-Post	CheckTrackingAndWarehouseForReceive	Get AL Snippet.
12	Custom Event	1	Codeunit	Purch.-Post	OnAfterCheckPurchDoc		Codeunit	Purch.-Post	CheckAndUpdate	Get AL Snippet.
13	Custom Event	2	Codeunit	NoSeriesManagement	OnAfterGetNextNo3		Codeunit	NoSeriesManagement	GetNextNo3	Get AL Snippet.
14	Custom Event	1	Codeunit	Purch.-Post	OnAfterUpdatePostingNos		Codeunit	Purch.-Post	UpdatePostingNos	Get AL Snippet.
15	Custom Event	1	Codeunit	Purch.-Post	OnBeforePostCommitPurchaseDoc		Codeunit	Purch.-Post	CheckAndUpdate	Get AL Snippet.
16	Custom Event	1	Codeunit	ApplicationManagem...	OnBeforeOnDatabaseModify		Codeunit	ApplicationManagem...	OnDatabaseModify	Get AL Snippet.
17	Custom Event	1	Codeunit	Integration Managem...	OnGetIntegrationDisabled		Codeunit	Integration Managem...	IsIntegrationDisabled	Get AL Snippet.
18	Custom Event	1	Codeunit	Integration Managem...	OnIsIntegrationRecord		Codeunit	Integration Managem...	IsIntegrationRecord	Get AL Snippet.
19	Custom Event	1	Codeunit	Integration Managem...	OnUpdateReferencedIdField		Codeunit	Integration Managem...	InsertUpdateIntegrationRecord	Get AL Snippet.
20	Custom Event	1	Codeunit	Integration Managem...	OnUpdateRelatedRecordIdFields		Codeunit	Integration Managem...	InsertUpdateIntegrationRecord	Get AL Snippet.
21	Custom Event	1	Codeunit	Data Upgrade Mgt.	OnIsUpgradeInProgress		Codeunit	Data Upgrade Mgt.	IsUpgradeInProgress	Get AL Snippet.
22	Custom Event	1	Codeunit	ApplicationManagem...	OnAfterOnDatabaseModify		Codeunit	ApplicationManagem...	OnDatabaseModify	Get AL Snippet.

- Click **Get AL Snippet** to get the event subscription code in AL. You can use the AL snippet code in Codeunits to subscribe to those events.

NOTE

The recorded events are not saved. When you refresh the page, the recorded events disappear.

For more information on how to subscribe to events, see [Subscribing to Events](#).

Recorded Events

All the recorded events display in the order they were called. The Event Recorder page provides information on the events that were raised including the details whether the raised events were trigger events or custom events. The custom events are either Business Events or Integration Events. For more information, see [Event Types](#).

You can identify the Event types, additionally, you can discover which object types and methods raised the events with the details like calling methods, object types, and object names. For more information about Events, see [Events in AL](#).

See Also

[Events in AL](#)
[Publishing Events](#)
[Raising Events](#)
[Subscribing to Events](#)
[Debugging in AL](#)
[Developing Extensions](#)

Notifications

3/31/2019 • 6 minutes to read

Notifications provide a programmatic way to send non-intrusive information to the User Interface (UI) in the Web client. Notifications differ from messages initiated by the **MESSAGE** method. Messages are modal, which means users are typically required to address the message and take some form of corrective action before they continue working. On the other hand, notifications are non-modal. Their purpose is to give users information about a current situation, but do not require any immediate action or block users from continuing with their current task. For example, you could have a notification that a customer's credit limit is exceeded.

Notifications in the UI

In the UI, notifications appear in the **Notification** bar (similar to validation errors) at the top of the page on which a user is currently working. The user can then choose to dismiss the notification, which clears it. Or, if actions are defined on notification, the user can choose one of the actions.

- There can be multiple notifications. The notifications appear in chronological order from top to bottom.
- Notifications remain for the duration of the page instance or until the user dismisses them or takes action on them.
- Notifications that are defined on sub-pages, for example in parts and FactBoxes, appear in the same **Notification** bar.
- Validation errors on the page will be shown first.

Notifications in the development environment

By using the **Notification** and **NotificationScope** data types and methods in AL, you can add code to send notifications to users. The following table provides an overview of the available methods. The sections that follow provide additional information about how to create notifications.

METHOD	DESCRIPTION
MESSAGE	Specifies the content of the notification that appears in the UI.
SCOPE	Specifies the scope in which the notification appears.
SEND	Sends the notification to be displayed by the client.
ADDACTION	Adds an action on the notification.
SETDATA	Sets a data property value for the notification
GETDATA	Gets a data property value from the notification.
RECALL	Recalls a sent notification.

Creating and sending a notification

You create a notification by using the **MESSAGE** and **SEND** methods. The **MESSAGE** method defines the message part of the notification. When the **SEND** method is called, the notification is sent to the client and content

of the message is displayed.

```
MyNotification.MESSAGE := 'This is a notification';  
MyNotification.SEND;
```

The **SEND** method call should be the last statement in the notification code, after any **ADDACTION** or **SETDATA** method calls for the notification instance.

Defining the notification scope

The scope determines where the notification is broadcast in the client. There are two different scopes: *LocalScope* and *GlobalScope*.

- A *LocalScope* notification appears in context of the user's current task, that is, on the page the user is currently working on. *LocalScope* is the default.
- A *GlobalScope* notification is not directly related to the current task, and will appear regardless of which the page the user is viewing.

NOTE

GlobalScope is currently not supported. This will be implemented in a future release.

The following code creates a notification in the *LocalScope*:

```
MyNotification.MESSAGE := 'This is a notification';  
MyNotification.SCOPE := NOTIFICATIONSCOPE::LocalScope;  
MyNotification.SEND;
```

Adding actions on a notification

You add actions on notifications by using the **ADDACTION** method. This method provides a way for you to create interactive notifications. By default, users have the option to dismiss the notifications. However, there might be cases where you want to provide users with different actions that they can take to address the notification, like opening an associated page for modifying data.

Conceptually, a notification action calls a method in a specified codeunit, passing the notification object in the call. The method includes the business logic for handling the action.

```
MyNotification.MESSAGE := 'This is a notification';  
MyNotification.SCOPE := NOTIFICATIONSCOPE::LocalScope;  
MyNotification.ADDACTION('Action 1', CODEUNIT::"Action Handler", 'RunAction1');  
MyNotification.ADDACTION('Action 2', CODEUNIT::"Action Handler", 'RunAction2');  
MyNotification.SEND;
```

The basic steps for adding an action are as follows:

1. Create a global method in a new or existing codeunit. The method must have a **Notification** data type parameter for receiving the notification object.
2. Add AL code to the method for handling the action.
3. Specify the codeunit and method in the **ADDACTION** method call.

IMPORTANT

You can have more than one action on a notification. A LocalScope notification can have up to 3 actions. A GlobalScope notification can have up to 2 actions.

Sending data with a notification

You use the **SETDATA** and **GETDATA** methods to add data to a notification, which is typically needed when actions are invoked. The **SETDATA** method sets, or adds, data to the notification. The data is defined as text in a key-value pair. With the **GETDATA** method, you can then retrieve the data again.

The following code sets data for a notification:

```
MyNotification.MESSAGE := 'This is a notification';
MyNotification.SCOPE := NOTIFICATIONSCOPE::LocalScope;
MyNotification.SETDATA('Created', FORMAT(CURRENTDATETIME,0,9));
MyNotification.SETDATA('ID', FORMAT(CREATEGUID,0,9));
MyNotification.ADDACTION('Action 1', CODEUNIT::"Action Handler", 'RunAction1');
MyNotification.ADDACTION('Action 2', CODEUNIT::"Action Handler", 'RunAction2');
MyNotification.SEND;
```

The following code gets the data for a notification:

```
DataValue := MyNotification.GETDATA('Created');
DataValue := MyNotification.GETDATA('ID');
```

Example

This simple example illustrates how notifications work and provides some insight into how you can use them. This example extends page **42 Sales Order** of the CRONUS International Ltd. demonstration database according to the following:

- The code compares a customer's balance with their credit limit. If the balance exceeds the credit limit, a notification is sent to the client.
- The notification includes an action, which has the caption **Change credit limit**, that opens page **21 Customer Card**. This enables the user to increase the credit limit.

To complete the example, follow these steps:

1. Create a page extension object that extends page **42 Sales Order**, and add the notification code on the **OnOpenPage** trigger.

```

pageextension 50100 CreditBalanceNotification extends "Sales Order"
{
    trigger OnOpenPage()
    var
        Customer: Record Customer;
        CreditBalanceNotification: Notification;
        OpenCustomer: Text;
        Text003: TextConst ENU = 'The current balance exceeds the credit limit.';
        Text004: TextConst ENU = 'Change credit limit';
    begin
        Customer.GET("Sell-to Customer No.");
        if Customer."Balance (LCY)" > Customer."Credit Limit (LCY)" then begin
            //Create the notification
            CreditBalanceNotification.MESSAGE(Text003);
            CreditBalanceNotification.SCOPE := NOTIFICATIONSCOPE::LocalScope;
            //Add a data property for the customer number
            CreditBalanceNotification.SETDATA('CustNumber', Customer."No.");
            //Add an action that calls the ActionHandler codeunit, which you define in the next step.
            CreditBalanceNotification.ADDACTION('Text004', CODEUNIT::"ActionHandler", 'OpenCustomer');
            //Send the notification to the client.
            CreditBalanceNotification.SEND;
        end;
    end;
}

```

2. Create a codeunit called **ActionHandler** for handling the notification action. Add a global method called **OpenCustomer** that has a **Notification** data type parameter called **CreditBalanceNotification** for receiving the Notification object, and include the following code on the method:

```

codeunit 50100 ActionHandler
{
    trigger OnRun()
    begin
    end;

    procedure OpenCustomer(CreditBalanceNotification: Notification)
    var
        CustNumber: Text;
        CustNo: Text;
        CustRec: Record Customer;
        CustPage: Page "Customer Card";
    begin
        //Get the customer number data from the SETDATA call.
        CustNo := CreditBalanceNotification.GETDATA(CustNumber);
        // Open the Customer Card page for the customer.
        if CustRec.GET(CustNo) then begin
            CustPage.SETRECORD(CustRec);
            CustPage.RUN;
        end else begin
            ERROR('Could not find Customer: ' + CustNo);
        end;
    end;
}

```

See Also

[Notification Data Type](#)
[Developing Extensions](#)
[Getting Started with AL](#)

Task Scheduler

3/31/2019 • 3 minutes to read

The task scheduler enables you to control when certain operations or processes (in other words *tasks*) are run. Basically, a task is a codeunit or report that is scheduled to run at a specific date and time. Tasks run in a background session between the Dynamics 365 Business Central service instance and database. Behind the scenes, the task scheduler is used by the job queue to process job queue entries that are created and managed from the clients.

In AL code, you create and manage tasks by using the AL methods that are available for the **TASKSCHEDULER** data type.

METHOD	DESCRIPTION	FOR MORE INFORMATION, SEE
CreateTask	Adds a task to run a codeunit at a specified date and time.	CreateTask Method
SetTaskReady	Sets a task to the Ready state. A task cannot run until it is Ready .	SetTaskReady Method
TaskExists	Checks whether a specific task exists.	TaskExists Method
CancelTask	Cancels a scheduled task.	CancelTask Method

How task scheduler works

To set up a task, you create a codeunit that contains the logic that you want to run at a scheduled time. Optionally, you can create a second codeunit that contains the logic to handle the task if an error occurs for any reason. This codeunit is referred to as a *failure codeunit*. Once you have the codeunits, you can add AL code to the application that calls the CREATETASK method to schedule a task to run the codeunits. The [CreateTask](#) method can also specify the earliest date to run the task, and whether the task is in the ready state.

Task flow

Here is an overview of the process that a task goes through:

1. After you add a task, the task is recorded in table **2000000175 Scheduled Task** of the database.
2. If the task is in the ready state, when the scheduled time occurs, a new background session is started and the task codeunit is run.

You can view the session in the table **2000000111 Session Event**.

3. If an error occurs, the following happens:
 - a. If a failure codeunit is not specified, then the retry flow is initiated.
 - b. If a failure codeunit has been specified, the error is passed in a call to the failure codeunit, and the failure codeunit is run.

If the failure codeunit does not handle the error or fails itself, then the retry flow is initiated.

Error conditions and retry process

A task can fail under the following conditions:

- The company cannot be opened.
- A SQL connection or transient error occurred with the database.
- The Dynamics 365 Business Central service instance restarted while the task was being run.

When an error occurs, unless the task is interrupted by the failure codeunit, the server instance will rerun the task according to the following retry flow:

1. Two minutes after the first failure.
2. Four minutes after the second failure.
3. Fifteen minutes after the third failure and any subsequent failures up to a maximum of 10 times, after which the task is canceled.

About task sessions and permissions

The task runs in a background session, which means that there is no user interface. The behavior is similar to that of the `STARTSESSION` method, where any dialog boxes that would normally appear are suppressed. For more information about specific dialog boxes, see [StartSession](#) method.

The session runs by using the same user/credentials that are used when calling AL code. The user must have appropriate permissions to the codeunit and any other objects that are associated with the operation of the codeunit.

See Also

[Task Scheduler Data Type](#)

[Developing Extensions](#)

[Getting Started with AL](#)

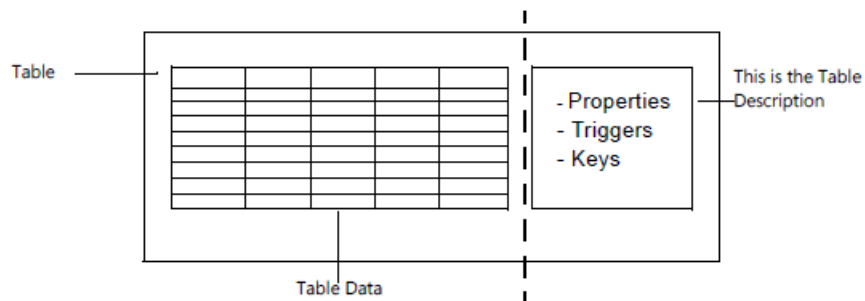
Tables Overview

4/24/2019 • 3 minutes to read

Tables are the fundamental objects in any database. They are the objects in which you store and manipulate data. This is true no matter what kind of data you need to manage. When you create a new database, you begin by building the tables. Later, you create pages and reports in order to access and view the data in the tables.

A table can be visualized as a two-dimensional matrix, consisting of columns and rows. The following illustration shows a table where each row is a record and each column is a field.

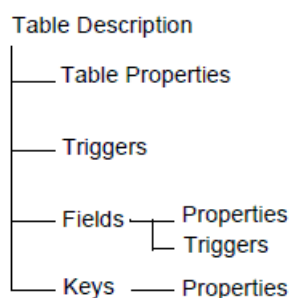
A table consists of two parts: the table data and a table description. The table data is the part users often think of as comprising the database, because it contains the actual records with their data fields. The layout and properties of those fields, however, are specified by the table description. The table description is not directly visible to the user. The following illustration shows how the table data and the table description together form a table.



When you design a table, you assign a number of characteristics to it, such as a name, an ID number, and the fields it contains. You also assign a number of characteristics (such as name, ID number, data type, and initial value) to each field. When you design a new table, you also specify which keys you want the system to maintain. All these characteristics are stored in the table description when you save your table design.

The information in the table description is used by SQL Server and occasionally by database users who need information about the table structure. The table description makes the database flexible, as it lets the system access tables with different structures. The database can extract the definitions of the table structure from the table description and thereby correctly access any table.

The following illustration shows that a table description contains properties, triggers, fields, and keys and shows how these are related.



The table description contains some properties that are related to the table, others that are related to the fields in the table, and other properties related to keys. You can also see that triggers are defined both for the table and for the fields in the table.

Creating tables

In AL code, you can create new tables or modify existing tables. Read more about creating and modifying tables in the following sections.

TO	SEE
Create a new table object	Table Object
Modify an existing table object	Table Extension Object
Decide which field data type you want to apply to your data	Field Data Types
Apply table and field properties	Table and Table Extension Properties
Set primary and secondary table keys	Table Keys

Using triggers in database design

Dynamics 365 Business Central supports setting up actions to take place in response to specific events. These are known as triggers. The following topics help to explain how Dynamics 365 Business Central implements this feature of database design.

TO	SEE
Learn about the set of triggers that Dynamics 365 Business Central supports for tables and fields.	Table and Field Triggers

Creating relationships between tables

In Dynamics 365 Business Central, the primary way to establish a connection between tables is to use the **TableRelation** property. The following topics go into detail about how this works.

TO	SEE
Get a brief introduction to relational database design in Dynamics 365 Business Central.	Setting Relationships Between Tables

See Also

[Developing Extensions in AL](#)

Table Object

3/31/2019 • 2 minutes to read

Tables are the core objects used to store data in Dynamics 365 Business Central. Regardless of how data is registered in the product - from a web service to a finger swipe on the phone app, the results of that transaction will be recorded in a table.

The structure of a table has four sections. The first block contains metadata for the overall table; the table type. The fields section describes the data elements that make up the table; their name and the type of data they can store. The keys section contains the definitions of the keys that the table needs to support. The final section details the triggers and code that can run on the table.

NOTE

Extension objects can have a name with a maximum length of 30 characters.

IMPORTANT

System and virtual tables cannot be extended. System tables are created in the ID range of 2.000.000.000 and above. For more information about object ranges, see [Object Ranges](#).

Snippet support

Typing the shortcut `ttable` will create the basic layout for a table object when using the AL Language extension in Visual Studio Code.

Table syntax

```

table id MyTable
{
    DataClassification = ToBeClassified;

    fields
    {
        field(1;MyField; Integer)
        {
            DataClassification = ToBeClassified;

        }
    }

    keys
    {
        key(PK; MyField)
        {
            Clustered = true;
        }
    }

    var
        myInt: Integer;

    trigger OnInsert()
    begin

    end;

    trigger OnModify()
    begin

    end;

    trigger OnDelete()
    begin

    end;

    trigger OnRename()
    begin

    end;

}

```

Table example

This table stores address information and has four fields; Address, Locality, Town/City, and County.

```

table 50104 Address
{
    caption = 'Sample table';
    DataPerCompany = true;

    fields
    {
        field(1; Address; Text[50])
        {
            Description = 'Address retrieved by Service';
        }
        field(2; Locality; Text[30])
        {
            Description = 'Locality retrieved by Service';
        }
        field(3; "Town/City"; Text[30])
        {
            Description = 'Town/City retrieved by Service';
        }
        field(4; County; Text[30])
        {
            Description = 'County retrieved by Service';

            trigger OnValidate();
            begin
                ValidateCounty(County);
            end;
        }
    }
}
keys
{
    key(PrimaryKey; Address)
    {
        Clustered = TRUE;
    }
}

var
    Msg: TextConst = 'Hello from my method';

trigger OnInsert();
begin

end;

procedure MyMethod();
begin
    Message(Msg);
end;
}

```

See Also

[AL Development Environment](#)

[Table Overview](#)

[Table Extension Object](#)

[Table Keys](#)

[Table Properties](#)

Table Extension Object

3/31/2019 • 2 minutes to read

The table extension object allows you to add additional fields or to change some properties on a table provided by the Dynamics 365 Business Central service. In this way, you can add data to the same table and treat it as a single table. For example, you may want to create a table extension for a retail winter sports store. In your solution you want to have `ShoeSize` as an additional field on the customer table. Adding this as an extension allows you to write code for the customer record and also include values for the `ShoeSize`.

Along with defining other fields, the table extension is where you write trigger code for your additional fields.

When developing a solution for Dynamics 365 Business Central, you will follow the code layout for a table extension as shown in the example below.

NOTE

Extension objects can have a name with a maximum length of 30 characters.

IMPORTANT

System and virtual tables cannot be extended. System tables are created in the ID range of 2,000,000,000 and above. For more information about object ranges, see [Object Ranges](#).

IMPORTANT

Extending tables from Dynamics 365 for Sales is currently not supported.

Snippet support

Typing the shortcut `tableext` will create the basic layout for a table extension object when using the AL Language extension in Visual Studio Code.

Properties

Using a table extension allows you to overwrite some properties on fields in the base table. For a list of Table properties, see [Table and Table Extension Properties](#).

Table extension syntax

```
tableextension Id MyExtension extends MyTargetTable
{
    fields
    {
        // Add changes to table fields here
    }

    var
        myInt: Integer;
}
```

Table extension example

This table extension object extends the Customer table object by adding a field `ShoeSize`, with ID 50116 and the data type `Integer`. It also contains a procedure to check if the `ShoeSize` field is filled in.

```
tableextension 50115 RetailWinterSportsStore extends Customer
{
    fields
    {
        field(50116;ShoeSize;Integer)
        {
            trigger OnValidate();
            begin
                if (rec.ShoeSize < 0) then
                    begin
                        message('Shoe size not valid: %1', rec.ShoeSize);
                    end;
                end;
            end;
        }
    }

    procedure HasShoeSize() : Boolean;
    begin
        exit(ShoeSize <> 0);
    end;

    trigger OnBeforeInsert();
    begin
        if not HasShoeSize then
            ShoeSize := Random(42);
        end;
    end;
}
```

Applies to

Tables

See Also

- [AL Development Environment](#)
- [Table Overview](#)
- [Table Object](#)
- [Table and Table Extension Properties](#)
- [Table Keys](#)

Setting Relationships Between Tables

3/31/2019 • 2 minutes to read

It is common to distinguish among the following types of relationships between tables in relational database design:

- One-to-many relationships
- Many-to-many relationships
- One-to-one relationships

The one-to-many relationship is the most common. If your database design model indicates that you need to set up a many-to-many relationship, then your design is probably inefficient. You can typically break down a many-to-many relationship into two one-to-many relationships. A one-to-one relationship is usually not optimal and can often be avoided by combining the two tables.

Using Relationships

If your database contains tables with related data, then you can define a relationship between them. You relate tables by specifying one or more fields that contain the same value in related records. These matching fields often have the same name in each table. You can use relationships to:

- Validate data entries
- Perform lookup functions in other tables
- Propagate changes automatically from one table to other tables

Table relationships and the TableRelation property

Table relationships are defined in the AL Language development environment using the **TableRelation** property. This property allows you to define both simple and advanced table relations.

NOTE

You can define a relationship only to a field that is a member of the primary key group.

Advanced table relations are typically prefixed with a conditional statement and include filters. The following syntax is for table relations.

```
<TableRelation> =  
    <TableName>[.<FieldName>] [WHERE(<TableFilters>)] |  
    IF (<Conditions>) <TableName>[.<FieldName>]  
    [WHERE(<TableFilters>)] ELSE <TableRelation>  
<Conditions> ::=  
    <TableFilters>  
<TableFilters> ::=  
    [<TableFilter> {,<TableFilter>}]  
<TableFilter> ::=  
    <DstFieldName>=CONST(<FieldConst>) |  
    <DstFieldName>=FILTER(<Filter>)
```

For example:

```

table 50120 TableWithRelation
{
  fields
  {
    field(1; Id; Integer) { }
    field(2; Type; enum TypeEnum) { }
    field(3; Relation; Code[20])
    {
      TableRelation =
        if (Type = const (Customer)) Customer
        else if (Type = const (Item)) Item;
    }
  }
}

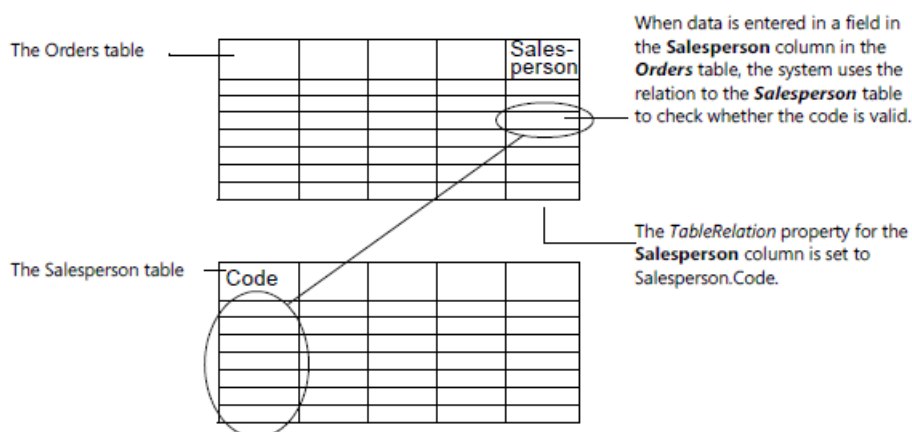
```

The following table describes each of the symbols.

SYMBOL	DESCRIPTION
TableName	Specifies the related table.
FieldName	Specifies a field in the related table.
Conditions	Table relations can be conditional.
TableFilters	A list of table filters.
TableFilter	A constant expression or a filter expression.
DstFieldName	Specifies the destination field name.
Filter	A filter expression, such as 10 20..30.

Examples of table relationships

For example, you have an **Orders** table that stores orders and a **Salesperson** table that stores the names of all salespeople in your company. In the **Orders** table, you can include a **Salesperson** field that identifies the salesperson. By setting up a relationship between these two tables, you can check whether the **Salesperson** field in the **Orders** table contains a valid code.



For example, you have a **Vendors** table with all your vendors and a **Currency Code** table. You can create a relationship between a **Currency Code** field in the **Vendors** table and the **Currency Code** table. This will allow users to look up information about valid currency codes.

Furthermore, if you change one of the currency codes in the **Currency Code** table, then the change is automatically propagated to all tables that refer to this code.

See Also

[Overview of Tables](#)

Viewing Table Data

6/17/2019 • 3 minutes to read

For developers, administrators, and support personnel, it can be useful to inspect table data in the tenant database, particularly when debugging or troubleshooting. To support this need, you can view table objects in the Web client. This lets you to see the data in all rows and columns of a specific table, including any columns that are added by table extensions.

- In a production environment, administrators and support can view a table directly from the Web client.
- In a development environment, in addition to viewing a table directly from the Web client, developers can view a table automatically when they publish/debug an AL project from Visual Studio Code.

NOTE

The table appears as read-only in the client, so modifications, insertions, and deletions cannot be made.

IMPORTANT

Data in the tables can be sensitive. Be sure to follow your organization's guidelines for handling such data.

Required permissions

Whether viewing the table directly from the client or from Visual Studio Code, your Dynamics 365 user account must have the following permissions:

- Read permission on the table that you want to view.
- Execution permission (direct) on the System object **1350 Run table**.

Any end-user that is assigned these permissions will be able to view that table in the browser.

For information about assigning permissions, see [Manage Users and Permissions](#).

View a table object directly from the client

To view a table, you add the `table=<TableID>` parameter to the client's address (URL), replacing `<TableID>` with the ID of the table that you want to view.

For example, if your URL starts with `https://businesscentral.dynamics.com`, then to view table **18 Customer** in your current company, you could use the following URL:

```
https://businesscentral.dynamics.com/?table=18
```

Or for a specific company, such as "CRONUS Inc.":

```
https://businesscentral.dynamics.com/?company=CRONUS%20Inc.&table=18
```

Note the use of `&` when `table=<TableID>` is not located directly after the domain name.

View a table object from an AL project in Visual Studio Code

You can configure an AL project to view a table when you publish or debug the project (pressing F5 or Ctrl+F5).

In the `launch.json` file for the project, set the `"startupObjectType"` parameter to `"table"` and the `"startupObjectId"` parameter to the ID of the table. For example:

```
{
  "version": "1.0.0",
  "configurations": [
    {
      "type": "al",
      "request": "launch",
      "name": "Publish to Microsoft cloud sandbox",
      "serverInstance": "dynamics",
      "startupObjectType": "Table"
      "startupObjectId": 18
    }
  ]
}
```

For more information about the `launch.json` file, see [Launch.json file](#).

Constraints

You cannot view virtual tables or the following system tables:

ID	NAME
2000000170	Configuration Package File
2000000170	Configuration Package File
2000000173	Data Sensitivity
2000000100	Debugger Breakpoint
2000000103	Debugger Watch
2000000130	Device
2000000114	Document Service
2000000190	Entitlement Set
2000000191	Entitlement
2000000180	MediaSet
2000000181	Media
2000000195	Membership Entitlement
2000000162	Nav App Capabilities

ID	NAME
2000000152	Nav App Data Archive
2000000161	Nav App Dependencies
2000000150	Nav App Object Metadata
2000000163	Nav App Object Prerequisites
2000000142	Nav App Resource
2000000151	Nav App TenantApp
2000000160	Nav App
2000000071	Object Metadata
2000000079	Object Tracking
2000000001	Object
2000000198	Page Documentation
2000000186	Profile Page Metadata
2000000082	Report Layout
2000000065	Send To Program
2000000112	Server Instance
2000000066	Style Sheet
2000000197	Token Cache
2000000081	Upgrade Blob Storage
2000000121	User Property
2000000076	Web Service
2000000194	Webhook Notification
2000000199	Webhook Subscription

See Also

[Developing Extensions](#)

Insert, Modify, ModifyAll, Delete, and DeleteAll Methods

3/31/2019 • 4 minutes to read

The following methods maintain the database by adding, modifying, and removing records:

- Insert
- Modify
- ModifyAll
- Delete
- DeleteAll

These methods are some of the most frequently used AL methods.

Some of these methods return an optional Boolean value that indicates whether the method succeeded. If you do not handle the return value in your code, a run-time error occurs when a method returns **false**. If you handle the return value by testing its value in an **if** statement, no error will occur, and you must take corrective action in the code.

Insert method

Insert inserts a record in a table. For more information, see [Insert Method](#). Insert has the following syntax.

```
[Ok :=] Record.Insert([RunTrigger])
```

The following example inserts a new record, with the **No.** and **Name** fields specified in the assigned values, while other fields will have their default values. If the **No.** field is the primary key of the **Customer** table, then the record will be inserted in the **Customer** table unless the table already contains a record with the same primary key. In this case you receive an error message because the return value is not tested.

This example requires that you create the following variable.

VARIABLE	DATA TYPE	SUBTYPE
Customer	Record	Customer

```
Customer.Init;  
Customer."No." := '4711';  
Customer.Name := 'Andrew Dixon';  
Customer.Insert;
```

Modify method

Modify modifies a record that already exists. For more information, see [Modify Method](#). Modify has the following syntax.

```
[Ok :=] Record.Modify([RunTrigger])
```

Modify returns an optional Boolean value. It returns **true** if the record to be modified exists; otherwise, it returns **false**.

The following example changes the name of customer 4711 to Richard Roe. This example requires that you create the following variable.

VARIABLE	DATA TYPE	SUBTYPE
Customer	Record	Customer

```
Customer.Get('4711');  
Customer.Name := 'Richard Roe';  
Customer.Modify;
```

ModifyAll method

ModifyAll performs a bulk update of records. For more information, see [ModifyAll Method](#).

ModifyAll has the following syntax.

```
Record.ModifyAll(Field, NewValue [, RunTrigger])
```

ModifyAll uses the current filters. This means that you can perform the update on a specified set of records in a table. ModifyAll returns no value, nor does it cause an error if the set of records to be changed is empty.

In the following example, the `SetRange` statement selects the records where Salesperson Code is PS. The ModifyAll statement changes the Salesperson Code of these records to JR. The example requires that you create the following variable.

VARIABLE	DATA TYPE	SUBTYPE
Customer	Record	Customer

```
Customer.SetRange("Salesperson Code", 'PS', 'PS');  
Customer.ModifyAll("Salesperson Code", 'JR');
```

Delete method

Delete deletes a record from the database. For more information, see [Delete Method](#) Delete has the following syntax.

```
[Ok :=] Record.Delete([RunTrigger])
```

The record that you want to delete must be specified by using the values in the primary key fields before you call this method. This means that Delete does take filters into consideration.

The following example shows how to use Delete to delete the record for customer number 4711. This example requires that you create the following variable.

VARIABLE	DATA TYPE	SUBTYPE
Customer	Record	Customer

```
Customer."No." := '4711';
Customer.Delete;
```

Delete returns an optional Boolean value. It returns **true** if the record could be found; otherwise, it returns **false**. Unless you test this value in your code, a run-time error occurs when Delete fails.

When you are developing your own applications, you should consider the following scenario:

1. Retrieve a record from the database.
2. Perform various checks to determine whether the record should be deleted.
3. If step 2 indicated that you should delete the record, then delete it.

This can cause problems in a multi-user environment. Another user can modify or delete the same record between your performing steps 2 and 3. If the record is modified, then perhaps the new contents of the record would have changed your decision to delete it. If it has been deleted by the other user, you can get a run-time error if you have just verified that the record existed (in step 1). If the design of your application indicates that you can encounter this problem, you should consider using the LockTable method. LockTable should be used sparingly because this method degrades performance. For more information about the LockTable method, see [LOCKTABLE Method](#).

DeleteAll method

DeleteAll deletes all the records that are specified by the filter settings. If no filters are applied, it deletes all the records in the table. For more information, see [DeleteAll Method](#). DeleteAll has the following syntax.

```
Record.DeleteAll([RunTrigger])
```

The following example deletes all the records from the **Customer** table where the Salesperson Code is PS. This example requires that you create the following variable.

VARIABLE	DATA TYPE	SUBTYPE
Customer	Record	Customer

```
Customer.SetRange("Salesperson Code", 'PS', 'PS');
Customer.DeleteAll;
```

NOTE

When you use DeleteAll (true), a copy of the AL variable with its initial values is created. This means that when you use DeleteAll(true) to run the OnDelete trigger, all the changes that were made to the variables in the method or codeunit that is making the call cannot be seen in the OnDelete trigger. If you want to see the changes that you made to the variables, you must use Delete(true) in a loop. There is no difference in performance between using DeleteAll(true) and using Delete(true) in a loop.

See Also

Get, Find, and Next Methods

3/31/2019 • 2 minutes to read

The following methods are used to search for records:

- Get
- Find
- Next

These methods are some of the most frequently used AL methods. When you search for records, you must know the difference between Get and Find and to know how to use Find and Next in conjunction.

Get method

The [Get Method \(Record\)](#) retrieves one record based on values of the primary key fields.

Get has the following syntax.

```
[Ok :=] Record.Get([Value],...)
```

For example, if the **No.** field is the primary key of the **Customer** table and if you have created a record variable called **CustomerRec** that has a subtype of Customer, then you can use Get in the following way.

```
CustomerRec.Get('4711');
```

The result is that the record of customer 4711 is retrieved.

Get produces a run-time error if it fails and the return value is not checked by the code. In the previous example, the actual code that you write should resemble the following.

```
if CustomerRec.GET('4711') then  
.... // Do some processing.  
else  
.... // Do some error processing.
```

Get searches for the records, regardless of the current filters, and it does not change any filters. Get always searches through all the records in a table.

Find methods

The [Find Method \(Record\)](#) locates a record in a table that is based on the values stored in the keys.

Find has the following syntax.

```
Ok := Record.Find([Which])
```

The *Which* parameter specifies how to perform the search. You can search for values that are greater than, less than, or equal to the key value, or for the first or last record in a table.

The important differences between Get and Find are as follows:

- Find uses the current filters.
- Find can look for records where the key value is equal to, greater than, or smaller than the search string.
- Find can find the first or the last record, depending on the sort order defined by the current key.

When you are developing applications in a relational database, there are often one-to-many relationships defined between tables. An example could be the relationship between an **Item** table, which registers items, and a **Sales Line** table, which registers the detailed lines from sales orders. One record in the **Sales Line** table can only be related to one item, but each item can be related to any number of sales line records. You would not want an item record to be deleted as long as there are still open sales orders that include the item. You can use Find to check for open sales orders.

The OnDelete trigger of the **Item** table includes the following code that illustrates using Find.

```
SalesOrderLine.SetCurrentKey(Type,"No.");
SalesOrderLine.SetRange(Type,SalesOrderLine.Type::Item);
SalesOrderLine.SetRange("No.,"No.");
IF SalesOrderLine.Find('-') THEN
  ERROR(Text001,TableCaption,"No.",SalesOrderLine."Document Type");
```

If you want to find the first record in a table or set, then use the [FindFirst Method \(Record\)](#). If you want to find the last record in a table or set, then use the [FindLast Method \(Record\)](#).

Next method

The [Next Method \(Record\)](#) is often used with FIND to step through the records of a table.

Next has the following syntax.

```
Steps := Record.Next([Steps])
```

In the following example, Find is used to go to the first record of the table. Next is used to step through every record, until there are no more. When there are no more records, Next returns 0 (zero).

```
if (Rec.FindSet) then
  repeat
    // process record
  until (Rec.Next = 0);
```

See Also

[AL Methods](#)

Retaining table data after publishing

3/31/2019 • 3 minutes to read

When developing an extension, you debug several times using the F5 shortcut key, and you also test your app by adding some sample data every time. To simplify the extension development process in Dynamics 365 Business Central, you can synchronize the sample data specified in the extension when you do subsequent publishing from Visual Studio Code.

How data synchronization works

The data synchronization between each publish is controlled by the `schemaUpdateMode` setting, which is specified in the `launch.json` file. This setting consists of three options; **Synchronize**, **Recreate**, and **ForceSync**.

The default value for `schemaUpdateMode` is set to the **Synchronize** mode, which means that every time you publish an extension to the development server, the data you entered previously stays. If you do not want to synchronize the sample data with each publish, you can change the `schemaUpdateMode` setting from `Synchronize` to, for example, `Recreate` with the syntax shown in the example below.

```
{
  "type": "al",
  "request": "launch",
  "name": "your own server",
  "server": "http://localhost",
  "serverInstance": "Nav",
  "authentication": "UserPassword",
  "startupobjectId": 22,
  "schemaUpdateMode": "Recreate"
}
```

Recreate mode

When you set the schema update mode to **Recreate**, all the tables and table extensions are recreated at every publish, which means that all the data in those tables are lost. This means that you will get empty records when you publish your extension.

ForceSync mode

ForceSync is similar to the existing Synchronize schema update mode, but contains more freedom to make schema changes while retaining data. To enable this mode, set `schemaUpdateMode` to `"ForceSync"` and then set the `"version"` parameter in the `app.json` file to a fixed number. Data will be preserved in almost all cases with the exception of changing the main table's primary key, in which case the data from the extension tables will be lost. Field renames are allowed and supported in this mode, but the data can only be preserved if you maintain the same ID for the field. If you change both the name and the ID of the field then the data will be lost.

NOTE

This schema update mode is only meant for testing and development and should never be used in production.

In addition to the `launch.json` file setting, the **ForceSync** switch is available through the PowerShell cmdlet `Sync-NavApp -Mode ForceSync`.

Things to be aware of

Synchronize is the default schema update mode for syncing the database and the extension. There are some key factors to consider when you work with the **Synchronize** mode.

- After publishing, the field data and the primary key information synchronizes with all the tables and the table extensions. This means that you can do additions easily, but not deletions. Breaking changes are never supported in synchronize mode. For example, you can add a field and sync that with the extension just by pressing the F5 shortcut key, but if a field is removed then the table data cannot be synchronized. If you, during development, for example, discover that you no longer want field **X**, and you then mark field **X** as obsolete, you may still want to write an [upgrade codeunit](#) to move the data from the obsolete field to a new field **Y** that you introduce. Later, the obsoleted field will not be available. But if you do not want the data, you can choose to use the **Recreate** mode instead.
- When you make changes to the data types, you can only *enlarge* the unit size, and *not decrease* the unit size. For example, you can set a text type from `Code[20]` to `Code[50]` or `Text[32]` to `Text[87]`, and you cannot set a text type from `Code[50]` to `Code[30]` or `Text[87]` to `Text[40]`.
- Making major table structural changes could lead to compilation errors. For example, if you want to update a primary key. In this case, the table data cannot be synchronized, and if you want to publish the extension, you must change the `schemaUpdateMode` to `Recreate`.

See Also

[AL Development Environment](#)

[Upgrading Extensions](#)

[Debugging](#)

Classifying Data in Dynamics 365

4/24/2019 • 5 minutes to read

Dynamics 365 includes development features for tagging business data with specific classifications. Specifically, this includes data that is stored in table fields of the database and telemetry data that is emitted from the application.

About Data Classification

Classifying data serves different purposes. It can make data easier and more efficient to locate and retrieve, and also help to add another layer of protection and security for handling private and sensitive data. It can supplement your process for making the application compliant with legislative and regulatory requirements for collecting, storing, and using personal information.

IMPORTANT

You should consider the data classification features offered in Dynamics 365 as the first layer of classification - done by developers (Dynamics 365 and partners) on customizations, add-ons, and extensions. The second layer is to classify the sensitivity of the data itself. For more information, see [Classifying Data Sensitivity](#). It is also important to consider end-users, and how they handle data they provide and that is made available to them.

What are the different data classifications?

The following table describes the different classifications that you can apply to data:

DATA CLASSIFICATION	DESCRIPTION	EXAMPLE
CustomerContent	Content directly provided/created by admins and users.	<ul style="list-style-type: none">• Customer generated BLOB or structured storage data• Customer-owned/provided secrets (passwords, certificates, encryption keys, storage keys)
EndUserIdentifiableInformation	(EUII) Data that identifies or could be used to identify the user of a Microsoft service. EUII does not contain Customer content.	<ul style="list-style-type: none">• User name or display name (DOMAIN\UserName)• User principle name (name@company.com)• User-specific IP address
AccountData	Customer billing information and payment instrument information, including administrator contact information, such as tenant administrator's name, address, or phone number.	<ul style="list-style-type: none">• Tenant administrator contact information (for example, tenant administrator's name, address, e-mail address, phone number)• Customer's provisioning information

DATA CLASSIFICATION	DESCRIPTION	EXAMPLE
EndUsePseudonymousIdentifiers	(EUPI) An identifier created by Microsoft tied to the user of a Microsoft service. When EUPI is combined with other information, such as a mapping table, it identifies the end user. EUPI does not contain information uploaded or created by the customer (Customer content or EUPI)	<ul style="list-style-type: none"> User GUIDs, PUIDs, or SIDs Session IDs
OrganizationIdentifiableInformation	(OII) Data that can be used to identify a tenant, generally config or usage data. This data is not linkable to a user and does not contain Customer content.	<ul style="list-style-type: none"> Tenant ID (non-GUID) Domain name in e-mail address (xxx@contoso.com) or other tenant-specific domain information
SystemMetadata	Data generated while running the service or program that is not linkable to a user or tenant.	<ul style="list-style-type: none"> Database table names, database column names, entity names

Classifying data in tables and fields

Table objects and field controls include the `DataClassification` property that you can use to tag data with one of the classifications previously described.

Dynamics 365 operates with some standard rules for classification:

- When you add a new field to a table, the field is assigned an initial value of **ToBeClassified**.
- FlowField and FlowFilter fields are automatically set to the **SystemMetadata** data classification. This cannot be changed.
- Existing tables and fields (except for FlowFields and FlowFilters) in an application that has been upgraded from a Dynamics 365 version without the `DataClassification` property, will automatically be assigned the **CustomerContent** classification.

IMPORTANT

Microsoft is providing this `DataClassification` property as a matter of convenience only. It is your responsibility to classify the data appropriately and comply with any laws and regulations that are applicable to you. Microsoft disclaims all responsibility towards any claims related to your classification of the data.

For more information about this property, see [DataClassification Property](#).

Data classification on upgrade

When you upgrade an application from a Dynamics 365 version that does not contain the `DataClassification` property, existing tables and fields (except for FlowFields and FlowFilters) will automatically be assigned the **CustomerContent** classification. You can then access the `DataClassification` property on these tables and fields, and change the classification as needed. FlowFields and FlowFilters will be assigned the **SystemMetadata** classification automatically.

IMPORTANT

After upgrade or import of objects, using fob files, that introduce new tables and/or fields, make sure to synchronize new tables and/or fields to enable [Data Sensitivity Classification](#) by running **SynchAllFields** function in Data Classification Mgt. Codeunit (Codeunit 1750). No action is needed when extensions are installed, as installation of extension automatically triggers **SynchAllFields** function. See example below

Run the script below from Developer Shell:

```
Invoke-NAVCodeunit -Tenant <TenantID> -CompanyName <CompanyName> -CodeunitID 1750 -MethodName 'SyncAllFields'  
-ServerInstance <ServerInstance>
```

Bulk-classifying data

The Field Data Classification report, which is described in the *Viewing current field classifications* section in this topic, provides an overview of the data classifications for fields. The report also lets you assign data classifications for more than one field. For example, this is useful if you are assigning classifications for the first time, or have changed several fields and want to update their classifications. You can bulk-edit classifications only for fields in CSIDE. The script does not update fields in extensions.

To bulk-edit classifications, export the report to Excel, update the classifications, and then save your changes. Then, in Windows PowerShell, run the following commands to run the Import-Module script and set the classifications on the fields.

To run the script from the default folder on the DVD, run:

```
Import-Module WindowsPowerShellScripts\DataClassification\DataClassification.psm1
```

To update the `DataClassification` property, run the following command. Replace `<FilePath>` with the full path to the client files. For example, `C:\Program Files\Microsoft Dynamics NAV\110\RoleTailored Client`.

```
Set-FieldDataClassificationFromExcelFile -ExcelFilePath "C:\NAV\W1 Fields (Main).xlsx" -SheetName 'Field Data Classification' -RTCFolder "<FilePath>" -DBName Navision_NAV2 -OutputFolder C:\Nav2\Classifications
```

Viewing current field classifications

To view the data classification on all fields, you can do one of the following:

- From Dynamics NAV Development Environment, in the **Tools** menu, select **Show Field Data Classification**.
- From the client, search for and open the **Field Data Classification** page.
- Create a page that has the virtual table **Field** (ID 2000000041) as its source, and open the page in the client.

Classifying data in custom telemetry trace events

Custom telemetry trace events are defined by calls to the SENDTRACETAG function/method in the application code. The SENDTRACETAG function/method includes an optional parameter called `DataClassification` that you can use to tag the telemetry trace event with a data classification.

For more information, see [SendTraceTag](#) and [Instrumenting an Application for Telemetry](#).

See Also

[Data Classification](#)

Integrating Dynamics 365 for Sales for Extension Development

3/31/2019 • 2 minutes to read

Develop extensions and streamline the workflow by synchronizing the Sales data from Microsoft Dynamics 365 for Sales with Dynamics 365 Business Central.

For developing extensions to integrate with sales data, you simply enable the tables used in Dynamics 365 for Sales. The extension development process includes the following set of properties to enable field mapping. You can enable the field mapping by using the following properties.

IMPORTANT

Extending tables from Dynamics 365 for Sales is currently not supported.

Associated table and field properties

The following properties are used for integrating with Microsoft Dynamics 365 for Sales:

PROPERTIES	APPLIES TO	DESCRIPTION
TableType Property	Tables	Specifies the table type. This enables the table to integrate with the external database. For example, <code>CRM</code> .
ExternalName Property	Tables, Fields	<p>Specifies the name of the original table in the external database when used as a table property.</p> <p>Specifies the field name of the corresponding field specified in the external table when used as a field property.</p>
ExternalAccess Property	Fields	Specifies the access to the underlying CRM entity when CRM tables are generated using the cmdlet.
ExternalType Property	Fields	Specifies the data type of the corresponding field in Dynamics 365 for Sales table.
OptionMembers Property	Fields	Sets the option values for a field, text box or variable.
OptionOrdinalValues Property	Fields	Specifies the list of option values. You can set this property, if the <code>ExternalType</code> is set to <code>Picklist</code> .

Enabling the entity

Typically in Dynamics 365 for Sales, entities handle the internal processes. In order to access to the underlying CRM entity, you use the `TableType` property and select the value called **CRM**. This enables the table as an integration table for integrating Dynamics 365 Business Central with Dynamics 365 for Sales. The table is mainly based on an entity in Dynamics 365 for Sales, such as the Accounts entity.

Snippet support

Typing the shortcut `ttable` will create the basic layout for a table object when using the AL Language extension in Visual Studio Code.

Example

In the following example, the `SalesIntegration` table uses the `TableType` and `ExternalName` properties to link the underlying **CRM** entity for mapping the fields from the `Sales` table with the specified fields.

```
table 50100 SalesIntegration
{
    TableType = CRM;
    ExternalName = 'Sales';

    fields
    {
        field(1; ActualSales; Integer)
        {
            ExternalName = 'ActualSale';
            ExternalAccess = Full;
            ExternalType = 'String';
        }

        field(2; SalesCategories; Option)
        {
            ExternalName='SalesCategory';
            ExternalAccess = Read;
            ExternalType = 'Picklist';
            OptionMembers = Manufacturing, Marketing, Support;
            OptionOrdinalValues = -1, 1, 2;
        }
    }
}
```

See Also

[Table Properties](#)

[TableType Property](#)

Pages Overview

3/31/2019 • 9 minutes to read

In Dynamics 365 Business Central, pages are the main way to display and organize data. Pages are the primary object that a user will interact with and have a different behavior based on the type of page that you choose. Pages are designed independently of the device they are to be rendered on, and in this way the same page can be reused across phone, tablet, and web clients.

A page is defined in code as an object composed of controls, properties, actions, and triggers. You can also use Designer in Dynamics 365 Business Central to create a page. For more information, see [Using Designer](#).

Whether you are creating a new page, or extending an existing page, you will add a new .al file to your project and describe the [page object](#) in code. The difference is basically that for a new page, you need to define the entire page, whereas when modifying an existing page, you only add the extra functionality or modify the existing.

The structure of a page is hierarchical and breaks down in to three sections. The first block contains metadata for the overall page. The metadata describes the page type and the source table it is showing data from. The next section; the layout, describes the visual parts on the page. The final section details the actions that are published on the page.

Furthermore, the page has properties. Properties work in the same way for pages as they do for other Dynamics 365 Business Central objects. For more information, see [Page Properties](#).

Page Metadata

For a new page object, you must at least specify the type of page; `PageType` and the data source; `SourceTable` of the page. And you can also set other metadata at the beginning of the declaration of the page object.

```
page 50102 PageName
{
    PageType = List;
    SourceTable = TableName;
    Editable = true;
    ContextSensitiveHelpPage = 'feature-overview';
    ...
}
```

Types of Pages

Which page type you choose depends on the application task that you want to support, the content that you want to display, and how you want to display it. The Role Center page is the main or home page and it helps the user focus on the most important daily tasks and activities. Other types of pages, such as list pages or card pages are typically linked from the home page for easy access. The following page types are available:

PAGE TYPE	DESCRIPTION
<code>RoleCenter</code>	The Role Center page is the main page.
<code>Card</code>	A Card page is used to view and edit one record or entity from a table.

PAGE TYPE	DESCRIPTION
<code>CardPart</code>	A Card Part page is used in a FactBox on another page to view or edit additional fields associated with a selected entity in the page.
<code>List</code>	A List page displays content from a table in a list format.
<code>ListPart</code>	Similar to a List page, a List Part page displays content from a table in a list format. The difference is that you use the List part page as another page in a FactBox or as a part of the Role Center page.
<code>ListPlus</code>	Similar to a List page, a List Plus page displays content from a table in a list format. The difference is that the List Plus page type can contain two lists in one page, and can be used as a two-dimensional matrix.
<code>Document</code>	A Document page usually consists of two separate pages combined into one, with one page nested in the other. A Document page is suitable for use when you want to display data from two tables that are linked together.
<code>WorkSheet</code>	You use a Worksheet page type for creating worksheet or journal task pages.
<code>ConfirmationDialog</code>	You use the ConfirmationDialog page to display messages or prompt users with a confirmation before they continue with the task that they are working on.
<code>StandardDialog</code>	The StandardDialog is a simple page type that you use when users only need to input data and do not need to perform other actions from the page.
<code>NavigatePage</code>	You use a Navigate page type to create a wizard that leads the user through a sequence of steps for completing a task.
<code>HeadlinePart</code>	You use a HeadlinePart page type to display a set of changing headlines on a Role Center. For more information, see Creating a Role Center Headline
<code>API</code>	Pages of this type are used to generate web service endpoints and cannot be shown in the user interface. This page type should not be extended by creating a page extension object. Instead, create a new API by adding a page object.

NOTE

For backwards compatibility we continue to support adding non-part pages as parts. We do, however, recommend that you redesign your page to only use Card part or List part, as we may remove support in a future update.

Page Layout

The page layout of the page object determines what the page will look like and is specified in the `layout` section. The `layout` contains one or more `area` sections that define a certain placement on the page.

You can choose between the following `area` categories:

AREA TYPE	PLACEMENT ON THE PAGE
<code>Content</code>	The content area displays the content of a RoleCenter or a List page.
<code>FactBoxes</code>	The factbox area is placed to the right-most side of a page. Displays content related to an item on the main content page.
<code>RoleCenter</code>	The RoleCenter is the main page of the application and is used for quick access to frequently used information and tasks.

Page Actions

All pages contain menu items and navigation controls called actions. In Dynamics 365 Business Central, actions are displayed at the top of each page in the ribbon or in the navigation pane. The `actions` section of the page describes what the user is able to do on a page and must be designed with the user's need for process support in mind.

Actions can be displayed in the ribbon of all pages and grouped together under the following actions tabs:

- Home
- Actions
- Navigate
- Report

Creating actions can include adding activity buttons/cues to a page, configuring navigation items on a user role center, or adding Reports to a page. To learn how you can enable users to quickly locate the actions they want to use, see [Actions](#).

Using Keywords to place Actions and Controls

You can use the following keywords in the `layout` section to place and move fields and groups on the page. Similarly, in the `actions` section, you use these keywords to place actions in the ribbon.

KEYWORDS	SYNTAX	APPLIES TO
<code>addfirst</code>	<code>addfirst(Anchor)</code>	Anchor: areas and groups
<code>addlast</code>	<code>addlast(Anchor)</code>	Anchor: areas and groups
<code>addafter</code>	<code>addafter(Anchor)</code>	Anchor: controls, actions and groups
<code>addbefore</code>	<code>addbefore(Anchor)</code>	Anchor: controls, actions and groups
<code>movefirst</code>	<code>movefirst(Anchor; Target1, Target2)</code>	Anchor: area, group Target: list of actions or list of controls
<code>movelast</code>	<code>movelast(Anchor; Target1, Target2)</code>	Anchor: area, group Target: list of actions or list of controls

KEYWORDS	SYNTAX	APPLIES TO
<code>moveafter</code>	<code>moveafter(Anchor; Target1, Target2)</code>	Anchor: controls, actions and groups Target: list of actions or list of controls
<code>movebefore</code>	<code>movebefore(Anchor; Target1, Target2)</code>	Anchor: controls, actions and groups Target: list of actions or list of controls
<code>modify</code>	<code>modify(Target)</code>	Target: controls, actions and groups

Example

To modify the existing fields and groups on a page, you use the `modify` keyword. See the code snippet below for `addlast`, `modify` and `action` syntax. In the following example, `action` creates a new group in the ribbon and places it last in the `Creation` group.

```

pageextension 70000020 CustomerCardExtension extends "Customer Card"
{
    layout
    {
        // Adding a new control field 'ShoeSize' in the group 'General'
        addlast(General)
        {
            field("Shoe Size"; ShoeSize)
            {
                Caption = 'Shoe size';

                trigger OnValidate();
                begin
                    if ShoeSize < 10 then
                        Error('Feet too small');
                    end;
                }
            }

            // Modifying the caption of the field 'Address 2'
            modify("Address 2")
            {
                Caption = 'New Address 2';
            }

            // Moving the two fields 'CreditLimit' and 'CalcCreditLimitLCYExpendedPct'
            // to be the first ones in the 'Balance' group.
            movefirst(Balance; CreditLimit, CalcCreditLimitLCYExpendedPct)
        }
    }
    actions
    {
        // Adding a new action group 'MyNewActionGroup' in the 'Creation' area
        addlast(Creation)
        {
            group(MyNewActionGroup)
            {
                action(MyNewAction)
                {
                    Caption = 'My New Action';

                    trigger OnAction();
                    begin
                        Message('My message');
                    end;
                }
            }
        }
    }
}

tableextension 70000020 CustomerTableExtension extends Customer
{
    fields
    {
        // Adding a new table field in the 'Customer' table
        field(50100; ShoeSize; Integer) { }
    }
}

```

Adding Help to the page objects

The Business Central user assistance model expects your solution to include tooltips and links to context-sensitive Help. For more information, see [User Assistance Model](#).

Context-sensitive Help

To apply context-sensitive Help to your app, you specify a URL to your Help library in the app.json file, and you then set the relevant target Help files as property values for each of your page objects and page extension objects. Between them, these two settings then give users access to context-sensitive Help for the features in your app at runtime. For more information, see [Configure Context-Sensitive Help](#).

Tooltips

In combination with descriptive captions and instructional text, tooltips are our current implementation of *embedded user assistance*, which is an important principle in today's world of software design. The tooltips are there to help users unblock themselves by providing an answer to the most likely questions the users might have, such as "What data can I input here?" or "What is the data used for?".

The base application has set the Tooltip property for all controls on (almost) all page objects. Most system actions also include tooltips so that users get a consistent experience. Your extensions are expected to also include tooltips for the same reason. For more information, see [ToolTip Property](#).

Instructional text

The base application has applied instructional text to setup guides and certain other types of page objects. Your extensions are expected to also include instructional text to setup guides for the same reason. For more information, see [InstructionalText Property](#).

Example

The following example shows how you can apply user assistance and link to Help in a page object:

```
page 50101 "Reward Card"
{
    PageType = Card;
    SourceTable = Reward;
    ContextSensitiveHelpPage = 'sales-rewards';

    layout
    {
        area(content)
        {
            group(Reward)
            {
                InstructionalText = 'Fill in the fields so that you can reward customers with discounts.';
                field("Reward Id"; "Reward ID")
                {
                    ApplicationArea = All;
                    ToolTip = 'Specifies the unique ID of the reward.';
                }

                field(Description; Description)
                {
                    ApplicationArea = All;
                    ToolTip = 'Specifies what this type of reward is used for.';
                }

                field("Discount Percentage"; "Discount Percentage")
                {
                    ApplicationArea = All;
                    ToolTip = 'Specifies the impact of the reward on the customer's price.';
                }
            }
        }
    }
}
```

In this example, the app.json file has specified a link to where the *sales-rewards* target file is published, such as

```
"contextSensitiveHelpUrl": "https://mysite.com/documentation" .
```

Best practices for designing pages

We recommend that you simplify the user experience by reducing what users see by default. You can promote the information that the users most frequently need to see and hide the less important information. For example:

- Place common tasks in the ribbon
- Organize information pages under FastTabs and, by default, hide the FastTabs that are infrequently visited.
- Use one to three FactBoxes on a page to provide supplementary information and a place for adding notes
- Add a target Help file for context-sensitive Help for the feature that the page object supports

See Also

[Page Properties Overview](#)

[Actions Overview](#)

[Using Designer](#)

[Adding a Factbox to a Page](#)

[Designing Role Centers](#)

[Configure Context-Sensitive Help](#)

Page Object

3/31/2019 • 2 minutes to read

Pages are the main way to display and organize visual data in Dynamics 365 Business Central. They are the primary object that a user will interact with and have a different behavior based on the type that you choose. Pages are designed independently of the device they are to be rendered on, and in this way the same page can be reused across phone, tablet, and web clients.

The structure of a page is hierarchical and breaks down in to three sections. The first block contains metadata for the overall page; the type of the page and the source table it is showing data from. The next section; the layout, describes the visual parts on the page. The final section details the actions that are published on the page.

When developing a solution for Dynamics 365 Business Central, you will follow the code layout for a page as shown in the page example below, but for more details on the individual controls and properties that are available, see [Page Property Overview](#).

NOTE

Extension objects can have a name with a maximum length of 30 characters.

Snippet support

Typing the shortcut `tpage` will create the basic layout for a page object when using the AL Language extension in Visual Studio Code.

Card page syntax

```

page Id MyPage
{
    PageType = Card;
    ApplicationArea = All;
    UsageCategory = Administration;
    SourceTable = TableName;
    ContextSensitiveHelpPage = 'my-feature';

    layout
    {
        area(Content)
        {
            group(GroupName)
            {
                field(Name; NameSource)
                {
                    ApplicationArea = All;
                }
            }
        }
    }

    actions
    {
        area(Processing)
        {
            action(ActionName)
            {
                ApplicationArea = All;

                trigger OnAction()
                begin
                    end;
            }
        }
    }

    var
        myInt: Integer;
}

```

List page syntax

```

page Id PageName
{
    PageType = List;
    ApplicationArea = All;
    SourceTable = TableName;

    layout
    {
        area(Content)
        {
            repeater(Group)
            {
                field(Name; NameSource)
                {
                    ApplicationArea = All;

                }
            }
        }
        area(Factboxes)
        {

        }
    }

    actions
    {
        area(Processing)
        {
            action(ActionName)
            {
                ApplicationArea = All;

                trigger OnAction();
                begin

                end;
            }
        }
    }
}

```

Page example

```

page 50101 SimpleCustomerCard
{
    PageType = Card;
    SourceTable = Customer;
    ContextSensitiveHelpPage = 'my-feature';

    layout
    {
        area(content)
        {
            group(General)
            {
                field("No."; "No.")
                {
                    ApplicationArea = All;
                    CaptionML = ENUS = 'Hello';

                    trigger OnValidate()
                    begin
                        if "No." < '' then
                            Message('Number too small')
                        end;
                    end;
                }

                field(Name; Name)
                {
                    ApplicationArea = All;
                }
                field(Address; Address)
                {
                    ApplicationArea = All;
                }
            }
        }
    }
    actions
    {
        area(Navigation)
        {
            action(NewAction)
            {
                ApplicationArea = All;
                RunObject = codeunit "Document Totals";
            }
        }
    }
}

```

See Also

[AL Development Environment](#)
[Adding Help Links from Pages, Reports, and XMLports](#)
[Page Extension Object](#)
[Page and Page Extension Properties Overview](#)
[Page Properties](#)
[Developing Extensions](#)
[Configure Context-Sensitive Help](#)

Page Extension Object

3/31/2019 • 2 minutes to read

The page extension object extends a Dynamics 365 Business Central page object and adds or overrides the functionality.

The structure of a page is hierarchical and breaks down into three sections. The first block contains metadata for the overall page; the type of the page and the source table it is showing data from. The next section; the layout, describes the visual parts on the page. The final section details the actions that are published on the page.

For more information about the Page and Page Extension objects, see [Pages Overview](#).

NOTE

Extension objects can have a name with a maximum length of 30 characters.

IMPORTANT

The API page type should not be extended by creating a page extension object. Instead, create a new API by adding a [page object](#).

Snippet support

Typing the shortcut `tpageext` will create the basic layout for a table object when using the AL Language extension in Visual Studio Code.

Page extension syntax

```
pageextension Id MyExtension extends MyTargetPage
{
    layout
    {
        // Add changes to page layout here
    }

    actions
    {
        // Add changes to page actions here
    }

    var
        myInt: Integer;
}
```

Page extension examples

The following page extension object extends the Customer Card page object by adding a field control `ShoeSize` to the `General` group on the page. The field control is added as the last control in the group using the `addLast` method. In the actions area, you can see what the syntax looks like for actions that execute triggers and actions that run objects.

```

pageextension 50110 CustomerCardExtension extends "Customer Card"
{
    layout
    {
        addlast(General)
        {
            field("Shoe Size"; ShoeSize)
            {
                ApplicationArea = All;
                Caption = 'ShoeSize';

                trigger OnValidate();
                begin
                    if ShoeSize < 10 then
                        Error('Feet too small');
                    end;
                end;
            }
        }

        modify("Address 2")
        {
            Caption = 'New Address 2';
        }
    }

    actions
    {
        addlast(Creation)
        {
            group(MyActionGroup)
            {
                Action(MyAction1)
                {
                    ApplicationArea = All;
                    Caption = 'Hello!';

                    trigger OnAction();
                    begin
                        Message('My message');
                    end;
                }

                Action(MyAction2)
                {
                    ApplicationArea = All;
                    RunObject = codeunit "Activities Mgt.";
                }
            }
        }
    }

    var
        Msg: TextConst = 'Hello from my method';

    trigger OnOpenPage();
    begin
        Message(Msg);
    end;
}

```

You can reference Report and XMLPort objects and use these objects in the **RunObject** property, as well as, declare variables of the types **Report** and **XMLPort** and call AL methods on them. This page extension object extends the Customer List page object by adding two actions; the first action calls the **Customer - List** report, the second action calls the **Export Contact** xmlport.

```

pageextension 50114 AddCustomerReport extends "Customer List"
{
    actions
    {
        AddLast("&Customer")
        {
            action("Customer List Report")
            {
                trigger OnAction();
                var
                    rep : Report "Customer - List";
                begin
                    rep.Run;
                end;
            }

            action("Export Contact List")
            {
                trigger OnAction();
                var
                    xml : XmlPort "Export Contact";
                begin
                    xml.Run;
                end;
            }
        }
    }
}

```

Applies To

Pages

See Also

[Page Object](#)

[Page and Page Extension Properties](#)

[Developing Extensions](#)

[AL Development Environment](#)

Page Customization Object

3/31/2019 • 2 minutes to read

The page customization object in Dynamics 365 Business Central allows you to add changes to the page layout and actions. The page customization object has more restrictions than the [page extension object](#); when you define a new page customization object, you cannot add variables, procedures, or triggers.

NOTE

Extension objects can have a name with a maximum length of 30 characters.

Snippet support

Typing the shortcut `tpagecust` will create the basic layout for a page customization object when using the AL Language extension in Visual Studio Code.

Page customization syntax

```
pagecustomization MyCustomization customizes MyTargetPage
{
    layout
    {
        // Add changes to page layout here
    }

    actions
    {
        // Add changes to page actions here
    }

    //Variables, procedures and triggers are not allowed on Page Customizations
}
```

Page customization example

The following page customization example `MyCustomization` is initialized to perform changes to **Customer List**. By using the `moveafter` method, `Blanket Orders` is moved next to the `Aged Accounts Receivable` action item. And the `modify` method is used to hide the `NewSalesBlanketOrder` action item.

```
pagecustomization MyCustomization customizes "Customer List"
{
    actions
    {
        moveafter("Blanket Orders"; "Aged Accounts Receivable")

        modify(NewSalesBlanketOrder)
        {
            Visible = false;
        }
    }
}
```

See Also

[Developing Extensions](#)

[AL Development Environment](#)

[Page Object](#)

[Page Extension Object](#)

[Page Extension Properties](#)

API Page Type

6/17/2019 • 2 minutes to read

Pages of the type `API` are used to create versioned, webhook-supported, OData v4 enabled REST web services. This type of page cannot be displayed in the user interface, but is intended for building reliable integration services. When creating this page type, you must specify a number of properties that provide information for the web service endpoint. Use the snippet `tpage - Page of type API` to get the right template and the list of these properties automatically filled in. This page type cannot be extended by creating a page extension object. Instead, you must create a new API by adding a page object.

Naming conventions

For the API page type, the following naming conventions exist:

- camelCase for naming attributes, tables, as well as `APIPublisher`, `APIGroup`, `EntityName`, and `EntitySetName`.
- Alphanumeric characters allowed (A-Z+a-z+0-9) in above elements.
- `APIVersion` follows the pattern `vX.Y` or `beta`.

At design time, the compiler will show warnings on casing violations and errors on naming violations. Once an API page is deployed, the corresponding `$metadata` is exposed on the endpoint of the page.

Example of the API page type

The following page example publishes an API available at: `../contoso/app1/v2.0/companies({id})/customers`. The `APIVersion` can be specified as one version, or a list of versions, if the API is supported through multiple versions.

```

page 50120 MyCustomerApi
{
    PageType = API;
    Caption = 'My Customer API';
    APIPublisher = 'contoso';
    APIGroup = 'app1';
    APIVersion = 'v2.0';
    EntityName = 'customer';
    EntitySetName = 'customers';
    SourceTable = Customer;
    DelayedInsert = true;

    layout
    {
        area(Content)
        {
            repeater(GroupName)
            {
                field(id; Id)
                {
                    Caption = 'ID';
                }
                field(name; Name)
                {
                    Caption = 'Name';
                }
            }
        }
    }
}

```

See Also

[AL Development Environment](#)

[API Query Type](#)

[Page Extension Object](#)

[Developing Extensions](#)

Designing Role Centers

5/3/2019 • 8 minutes to read

The strength of Dynamics 365 is its role-tailored experience that helps users focus on the work that is important to them. The Role Center is an integral part of the role-tailored experience. And as a developer, role-tailoring should be the foundation for your Role Center design.

About the Role Center

The Role Center is the user's entry point and home page for Dynamics 365. You can develop several different Role Centers, where each Role Center is customized to the profile of the intended users. For example, you could have Role Centers that target the different levels within an organization, such business owners, department leads, and information workers.

Role Centers are based on a user-centric design model. You should design a Role Center to give users quick access to the information that is most important to them in their daily work - displaying information that is pertinent to their role in the company and enabling them to easily navigate to relevant pages for viewing data and performing tasks.

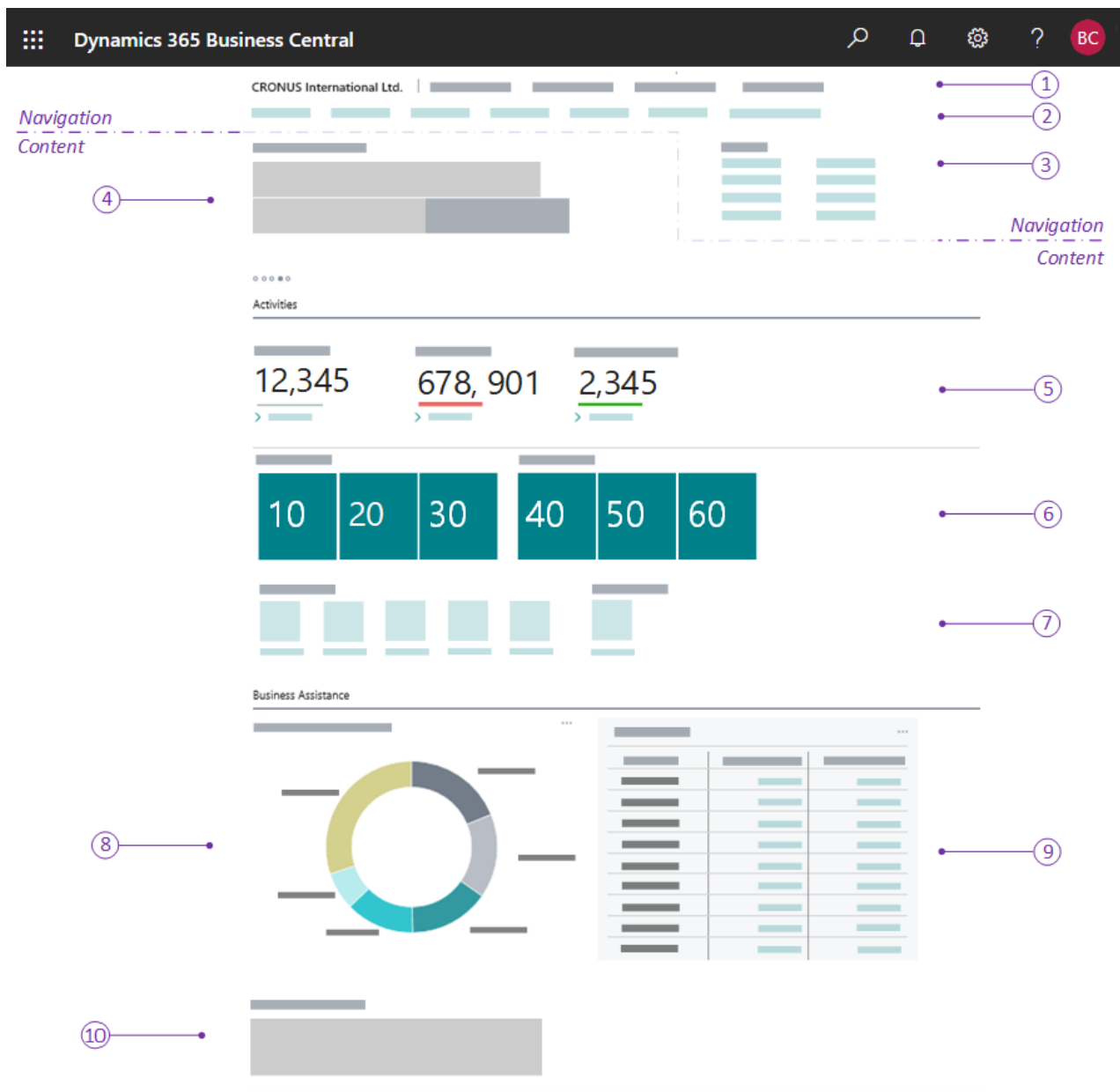
Customizing a Role Center from the client

In the client, users who work across multiple roles can easily switch Role Centers to shift their focus to different tasks. Users can also personalize their Role Centers by rearranging or hiding content as they like. For more information, see [Personalizing Your Workspace](#).

As a developer or administrator, you can use Designer to customize a Role Center the same way that individual users personalize their own workspaces. The difference is that changes you make are applied to all users assigned to the Role Center. For more information, see [Using Designer](#).

Role Center structure

A Role Center is defined by page that has the [PageType property](#) set to `RoleCenter`. The Role Center page is divided into two main areas: navigation/actions area and content area. The following figure illustrates the general layout and elements of a Role Center page.



Navigation and Actions area

The navigation and actions area appears at the top of the Role Center page, and provides links to other objects, such as pages, reports, and codeunits. You define the navigation area by adding actions to the Role Center page code, under the `actions` control in the page code. The navigation and actions area is subdivided into smaller areas by using different `area()` controls as described in the following table:

	AREA	DESCRIPTION	USAGE GUIDELINES
1	Navigation menus	<p>The top-level navigation consists of one or more root items that expand to display a sub-menu of links to other pages. The pages targeted by the sub-menus will open in the content area of the Role Center.</p> <p>You define this area with an <code>area(sections)</code> control in the page code.</p>	<p>The top-level navigation should provide access to relevant entity lists for the role's areas of business. For example, typical root items for a business manager could be finance, sales, and purchasing. You should place the root items in order of importance, starting from the left.</p>

	AREA	DESCRIPTION	USAGE GUIDELINES
2	Navigation bar	<p>The second-level navigation displays a flat list of links to other pages. The pages targeted by the links will open in the content area of the Role Center.</p> <p>You define this area with an <code>area(embedding)</code> control in the page code.</p>	<p>You should use these items to link to users' most useful entity lists in their business process. For example, with a business manager, these could be links to customers, sales orders, and bank accounts. You should place items in the order that reflects the business process sequence. Try to limit the number of second-level items, and consider placing items in the top-level navigation instead, if the number gets too large.</p>
3	Actions	<p>The actions area provides links to pages, reports, and codeunits. The links can be displayed on the root-level or grouped in a sub-menu. The objects targeted by these links will open in a separate window in front of the Role Center page.</p> <p>You can define the actions by using the three different <code>area()</code> controls that are described below:</p>	<p>The action area is designed for running the most important or most often used tasks and operations required by users. Actions will typically target card type pages that enable users to create new entities, such as customers, invoices, and sales orders, or run reports. Place the most important action at the root-level, and group closely related actions in a sub-menu.</p>
		<p><code>area(creation)</code> - Actions in this control will appear first in the action area, and will display with a plus (+) icon.</p>	<p>Use this control to target pages that enable the user to create new entities.</p>
		<p><code>area(processing)</code> - Actions in this control will appear after the <code>area(creation)</code> items. You can group actions in sub-menus by using a <code>group</code> control.</p>	<p>Use this control to target pages that are associated with the work flow for processing documents, such as payments or sales orders. Use the <code>group</code> control to organize similar actions under a common parent.</p>
		<p><code>area(reporting)</code> - Actions in this control will appear last in the action area. They display with a default report icon.</p>	<p>Use this control to target report objects.</p>

For more information about navigation, see [Adding to Navigation](#).

Behavioral points of interest

- The order of the `area()` controls in the page code is not important. However, the order of the individual actions and groups is important because they will appear in the order in which they appear in page code.

- In page code, if the first part in the content area is a Headline part, then in the client, the actions area will be automatically positioned either to the right of the Headline part or after the Headline part, depending on the browser window size. If the first part is not a Headline, the actions area will appear directly after the navigation area, and extend the width of the workspace.

Content area

The content area consists of one or more parts that display content from other pages. Unlike the navigation and actions area that is completely defined in the Role Center page code, the content area consists of self-contained, independent page part objects that can be used across Role Centers and in other pages. You define the content area by adding a `layout` control in the page code, and then a `part` control for each individual part to display.

The following table describes some of the most common parts for Role Centers, as illustrated in the previous figure.

	ELEMENT	DESCRIPTION	MORE INFORMATION	
4	Headline	Displays a series of automatically changing headlines that provide users with up-to-date information and insight into the business and daily work. This is created by a <code>HeadlinePart</code> page type.	Creating Role Center Headlines	
5	Wide data cues	A set of cues for displaying large numbers, like monetary values. This is created by using a <code>cuegroup</code> control on a <code>CardPart</code> page type, where the Layout property is set to <code>wide</code> .	Wide Cues	
6	Data cues	Provide a visual representation of aggregated business data, such as the number of open sales invoices or the total sales for the month. These are created by using a <code>cuegroup</code> control on a <code>CardPart</code> page type.	Creating Cues	

	ELEMENT	DESCRIPTION	MORE INFORMATION	
7	Action cues	Tiles that link to tasks or operations, like opening another page, starting a video, targeting another URL, or running code. These are created by using a <code>cuegroup</code> control on a <code>CardPart</code> page type	Action Tiles	
8	Chart	A graphical and interactive representation of your business data that can be sourced by a custom business chart control add-in or an embedded Power BI report.		
9	CardPart or ListPart page	Displays data fields in a form or tabular layout.	Page Object	
10	Control add-in	Displays custom content by using HTML-based control add-in.	Control Add-in Object	

Behavioral points of interest

- In general, the parts will appear in the client according to the order in which they are defined in the Role Center page code and will automatically rearrange horizontally and vertically to fill the available workspace.
- However, in the Web client, page parts that contain cues are automatically grouped under a common **Activities** section, no matter where they are placed in the code. All other page parts are grouped under the **Business Assistance** section. Within **Activities** and **Business Assistance** sections, the parts will arrange according to the order in which they are defined in the page code.

Development tips for overall page design

- Do not apply grouping to parts in the content area because this prevents parts from flowing to fill the available space. This gives the best experience to users with different screen resolutions or those on mobile devices.
- To achieve the best readability and discoverability, place Headlines first, followed by cues, and then the remaining parts.
- You cannot add custom logic directly to a Role Center page code. Code is limited to defining navigation, actions, and parts. All other code is ignored.
- Role Centers can be highly specialized, in the fact that all navigation, actions, and content is optional. For example, you could have a single part that fills the entire workspace.

Design for all display targets

- Role Center pages are also the primary entry point on mobile devices. Mobile devices will display the same content as the Web client, but is presented in a different way to suit how users hold and interact with their

mobile device.

- You can preview how your Role Center will look on mobile devices directly in Designer.
- Some limitations on mobile devices include the following:
 - On tablets, there is a limit on the number of cues that can be displayed.
 - On phones, there is a limit on the number of parts in the content area that can be displayed.
 - Role Center pages cannot be displayed when they are embedded in Outlook or SharePoint.

Using the Role Center in the client

To use or test the new Role Center in the client, you must first associate the Role Center page with a profile.

Profiles define user roles and each profile is associated with a single Role Center page. Create a new [profile object](#) that references your page. Then, go to **My Settings** and select the new profile.

See Also

[AL Development Environment](#)

[Page Extension Object](#)

[Actions Overview](#)

[Adding Pages and Reports to Tell Me](#)

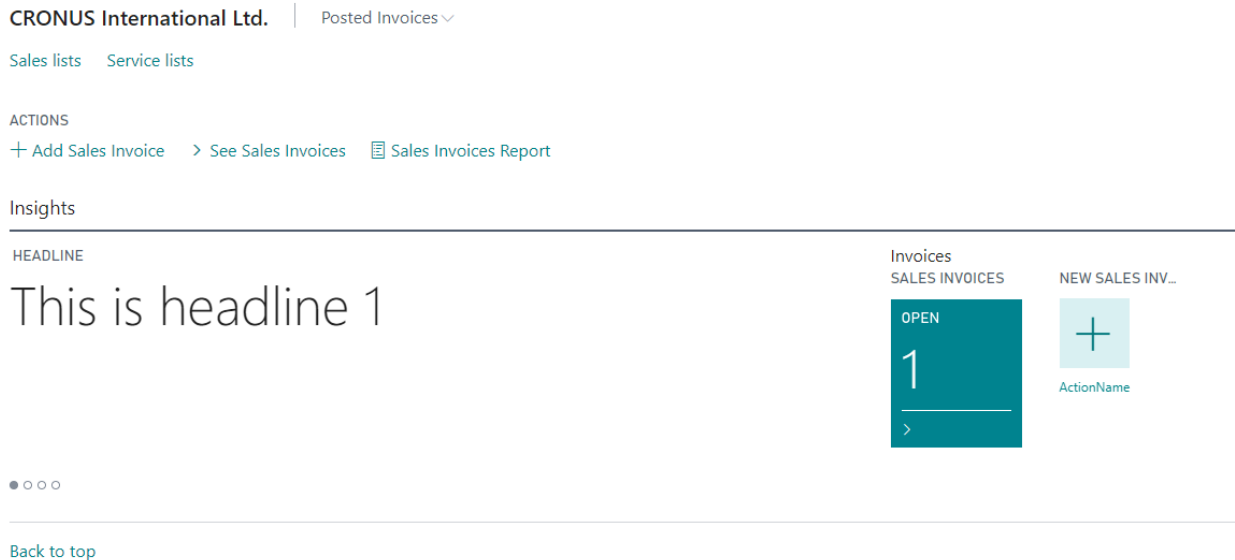
[Personalizing Your Workspace](#)

[Using Designer](#)

Simple Role Center Code Example

5/21/2019 • 2 minutes to read

The AL code in this article creates a simple Role Center customized for users assigned to a new profile.



For a more detailed explanation of Role Centers, see [Designing Role Centers](#).

This example uses the **RoleCenterHeadline** page code [example](#) to display the headline and the **SalesInvoiceCuePage** page and the following code [example](#) for the Cue and Action tile.

```
page 50125 MyRoleCenter
{
    PageType = RoleCenter;
    Caption = 'My Role Center';

    layout
    {
        area(RoleCenter)
        {
            group(Group1)
            {
                part(Part1; RoleCenterHeadline)
                {
                    ApplicationArea = All;
                }

                part(Part2; SalesInvoiceCuePage)
                {
                    Caption = 'Invoices';
                }
            }
        }
    }

    actions
    {
        area(Sections)
        {
            group(PostedInvoices)
            {
```

```

        Caption = 'Posted Invoices';
        Image = RegisteredDocs;
        action(PostedServiceInvoices)
        {
            Caption = 'Posted Service Invoices';
            RunObject = Page "Posted Service Invoices";
            ApplicationArea = All;
        }

        action(PostedSalesInvoices)
        {
            Caption = 'Posted Sales Invoices';
            RunObject = Page "Posted Sales Invoices";
            ApplicationArea = All;
        }
    }
}

area(Embedding)
{
    action(Sales)
    {
        Caption = 'Sales lists';
        RunObject = Page "Sales list";
        ApplicationArea = All;
    }

    action(Services)
    {
        Caption = 'Service lists';
        RunObject = Page "Service list";
        ApplicationArea = All;
    }
}

area(Processing)
{
    action(SeeSalesInvoices)
    {
        Caption = 'See Sales Invoices';
        RunObject = Page "Posted Sales Invoices";
    }
}

area(Creation)
{
    action(AddSalesInvoice)
    {
        Caption = 'Add Sales Invoice';
        Image = NewInvoice;
        RunObject = Page "Sales Invoice";
        RunPageMode = Create;
    }
}

area(Reporting)
{
    action(SalesInvoicesReport)
    {
        Caption = 'Sales Invoices Report';
        Image = "Report";
        RunObject = Report "Sales - Invoice";
    }
}
}
}

```

See Also

[AL Development Environment](#)

[Page Extension Object](#)

[Actions Overview](#)

[Adding Pages and Reports to Tell Me](#)

[Personalizing Your Workspace](#)

[Using Designer](#)

Adding Menus to the Navigation and Actions Area

5/3/2019 • 3 minutes to read

The navigation area appears at the top of the Dynamics 365 Business Central window, and contains multiple sections that enable users to quickly navigate and perform actions in Dynamics 365 Business Central. A single section in the navigation area can be defined as a menu group that contains multiple sub-menu items.

Adding to the top-level navigation

The top-level navigation area displays the Home menu items by default; the other menu items can be accessed by clicking on the small drop-down arrow placed next to the *selected* menu category in Dynamics 365 Business Central. For users, the menu groups that display in the navigation area could change depending on the Role Center page that they access.

Example

The example below explains how to add the menu group called `My Customers` to the top-level navigation area. The sub-menu items for `My Customers` contain the `Customer Bank Account List` and `Customer Ledger Entries` actions, each opening the corresponding page object. In this example, the `My Customers` menu will appear in the navigation area for the **Sales Order Processor** Role Center.

```
pageextension 50120 ExtendNavigationArea extends "Order Processor Role Center"
{
    actions
    {
        addlast(Sections)
        {
            group("My Customers")
            {
                action("Customer Bank Account List")
                {
                    RunObject = page "Customer Bank Account List";
                    ApplicationArea = All;
                }
                action("Customer Ledger Entries")
                {
                    RunObject = page "Customer Ledger Entries";
                    ApplicationArea = All;
                }
            }
        }
    }
}
```

You can also enable pages and reports to appear in the Dynamics 365 Business Central search for a quick navigational support. For more information, see [Adding Pages and Reports to Tell Me](#).

Adding to the secondary-level navigation

The second-level navigation offers a flat list of links to other pages. These should be the most relevant pages needed for a user's business process. We recommend to have only the most important items on this level and to place the others in the top-level navigation instead.

Example

The following code adds a new link to the secondary-level navigation by defining this area with an `area(Embedding)` control in the page code. The object targeted in this case is the `Sales Cycles` page and it will appear as the last one.

```
...
addlast(Embedding)
{
    action("Sales Cycles")
    {
        RunObject = page "Sales Cycles";
        ApplicationArea = All;
    }
}
```

Adding to actions

The actions area displays the most important or most often used tasks and operations required by users. It contains links to pages, reports, and codeunits. The links are placed on the root-level, and they can be grouped in a submenu.

You can define the actions by using three different `area()` controls. The first action area that appears at the top of the Role Center page is `area(Creation)`. The following example adds the item last, and it allows opening the `Sales Journal` page.

Example

```
...
addlast(Creation)
{
    action("Sales Journal")
    {
        ApplicationArea = All;
        RunObject = page "Sales Journal";
    }
}
```

The actions in the `area(Processing)` control appears after the `area(Creation)` items. The example below shows how you can use the group control to organize similar actions under a common parent. The created group is placed at the end of this action area, and it targets pages needed for processing sales documents.

Example

```

...
addlast(Processing)
{
    group(Documents)
    {
        action("Sales Document Entity")
        {
            ApplicationArea = All;
            RunObject = page "Sales Document Entity";
        }
        action("Sales Document Line Entity")
        {
            ApplicationArea = All;
            RunObject = page "Sales Document Line Entity";
        }
    }
}

```

The actions in the `area(Reporting)` control will appear last in the action area and they display with a default report icon. This control's purpose is to target report objects and the following example opens the `Customer Sales Statistics` report.

Example

```

...
addlast(Reporting)
{
    action("Customer Statistics")
    {
        ApplicationArea = All;
        RunObject = report "Customer Sales Statistics";
    }
}

```

See Also

[AL Development Environment](#)

[Page Extension Object](#)

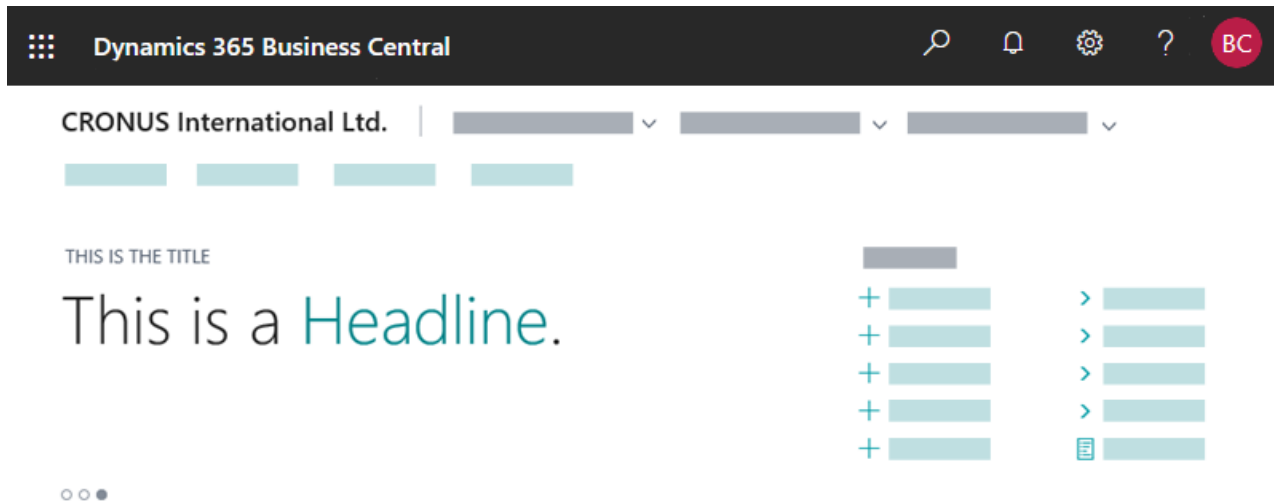
[Actions Overview](#)

[Adding Pages and Reports to Tell Me](#)

Creating a Role Center Headline

3/31/2019 • 5 minutes to read

You can set up a Role Center to display a series of headlines, where headlines appear one at a time for a predefined period of time before displaying the next.



The headlines can provide users with up-to-date information and insight into the business and daily work. Typical categories of headlines might include:

- My performance
- My workday
- Organizational health
- Productivity tips
- Cross-tenant insights (performance relative to peers)
- Getting started information

IMPORTANT

Headlines will only appear in Web client; they will not be shown on other client types.

Design concept

In development

In short, the Headline is basically a page that contains one or more fields. The page must be the **HeadlinePart** type page. Each field defines an individual headline to be displayed. The source for a field can be an expression or a field in an underlying table.

- The **HeadlinePart** page is designed for Role Centers, that is, pages that have the type **RoleCenter**. If you use a **HeadlinePart** page on another page type, the part will not render in the client.
- Using the OnDrillDown trigger, headlines can be made interactive, meaning that users can select the headline to dig deeper into numbers or values that are shown in the headline or link to another page or URL.
- You can dynamically toggle visibility of a specific headline, for example based its relevancy, by setting the

Visible property on the field.

- There are only a few field properties that apply to fields that are used on a **HeadlinePart** type page, including Expression, Visible, ApplicationArea, Drilldown, and DrillDownPageID. All other properties are ignored.

In the client

The Role Center will start by displaying the first visible headline that is defined on the HeadlinePart page. The headline will appear for 5 seconds, then the next headline will appear for 5 seconds, and so on. When all the headlines have been displayed, it will cycle back to the first headline, and continue from there.

- If a headline is interactive, users can select the headline to open the target defined in the headline.
- Users can pause on a headline by pointing to it.
- Users can manually switch among headlines by selecting a corresponding dot that is displayed under the headlines.
- Users can personalize their Role Center to show or hide the Headline part as they like.

Creating a HeadlinePart page

1. Implement the logic that resolves field expressions for the headlines that you will use on the page.

You can apply more flexible and complex patterns, such as having data tables drive the text, drill-down and relevance engine for headlines.

2. Create a page that has the [PageType property](#) set to `HeadlinePart`.
3. For each headline, add a field, and set the [Expression property](#). The order of the fields, determines the order in which they appear.

The following example shows the AL code for a simple **HeadlinePart** page that consists of four fields that display static text.

```

page 50100 RoleCenterHeadline
{
  PageType = HeadLinePart;

  layout
  {
    area(content)
    {
      field(Headline1; text001)
      {
      }
      field(Headline2; text002)
      {
      }
      field(Headline3; text003)
      {
      }
      field(Headline4; text004)
      {
      }
    }
  }
}

var
  text001: TextConst ENU='This is headline 1';
  text002: TextConst ENU='This is headline 2';
  text003: TextConst ENU='This is headline 3';
  text004: TextConst ENU='This is headline 4';
}

```

4. You can now add the **HeadlinePart** page to the **RoleCenter** page.

Constructing Headlines with the Expression property

The `Expression` property supports the following syntax that enables you to specify a title for the headline, the headline text itself, and emphasize a string of text in the headline:

```
'<qualifier>Title</qualifier><payload>This is the <emphasize>Headline</emphasize>.</payload>'
```

TAG	DESCRIPTION
<code><qualifier></qualifier></code>	Specifies the title that appears above the headline. If you omit this tag, the text HEADLINE will be used by default.
<code><payload></payload></code>	Specifies the actual headline text.
<code><emphasize></emphasize></code>	Applies the style to the text.

The `Expression` property must evaluate to the correct syntax. For example, looking back at the previous example, the text constant `text001` could be:

```

text001: TextConst ENU='<qualifier>The first headline</qualifier><payload>This is the <emphasize>Headline
1</emphasize>.</payload>';

```

Making headlines interactive

You can use the [OnDrillDown trigger](#) of a headline field to link the headline to more details or relevant information about what is shown in the headlines. For example, if the headline announced the largest sales order for the month, you could set up the headline to open a page that shows a sorted list of sales order for the month.

The following code uses the OnDrillDown trigger to link `Headline1` to the Dynamics 365 online help.

```
field(Headline1; text001)
{
    trigger OnDrillDown()
    var
        DrillDownURL: TextConst ENU='https://go.microsoft.com/fwlink/?linkid=867580';
    begin
        Hyperlink(DrillDownURL)
    end;
}
```

Changing the visibility of headlines

You can use the [Visible property](#) to show or hide headlines that are defined on the **HeadlinePart** page. With the `Visible` property, you can show or hide the control either statically by setting the property to **true** or **false**, or dynamically by using a `Boolean` variable.

Static visibility

With static visibility, you can simply set the `Visible` property on specific fields. For example, following code hides `Headline3`:

```
{
    field(Headline1; text001)
    {

    }
    field(Headline2; text002)
    {

    }
    field(Headline3; text003)
    {
        Visible=false;
    }
    field(Headline4; text004)
    {

    }
}
```

By adding fields under `Group` controls, you can hide or show more than one headline by setting the `Visible` property on the `Group` control. For example, the following code hides headings `Headline3` and `Headline4`:

```

group(Group1)
{
    field(Headline1; text001)
    {

    }
    field(Headline2; text002)
    {

    }
}
group(Group2)
{
    Visible=false;
    field(Headline3; text003)
    {

    }
    field(Headline4; text004)
    {

    }
}
}

```

IMPORTANT

On **HeadlinePart** type pages, the `group` control has no effect on the UI, like with other page types. Its primary purpose is to enable developers to group headlines for controlling visibility.

Dynamic visibility

With dynamic visibility, you can show or hide a headline based on a condition that evaluates to `true` or `false`.

- To dynamically show or hide a headline when the **HeadlinePart** page opens, the headline field must be in `group` control, and you set the `Visible` property on the `group` control to the `Boolean` variable that determines the visibility. For example, you could add code on the page's `OnAfterGetRecord` trigger that evaluates the relevance of displaying `Headline3` and results in a `Boolean` variable being set to `true` or `false`.
- To dynamically show or hide a headline while a page is open, you set the `Visible` property on the `field` control to the `Boolean` variable that determines the visibility.

```
group(Group1)
{
    field(Headline1; text001)
    {
    }
    field(Headline2; text002)
    {
    }
}
group(Group2)
{
    // Determines visibility when the page opens
    Visible=ShowHeadline3;
    field(Headline3; text003)
    {
        // Determines visibility while the page is open
        Visible=ShowHeadline3;
    }
    field(Headline4; text004)
    {
    }
}
```

See Also

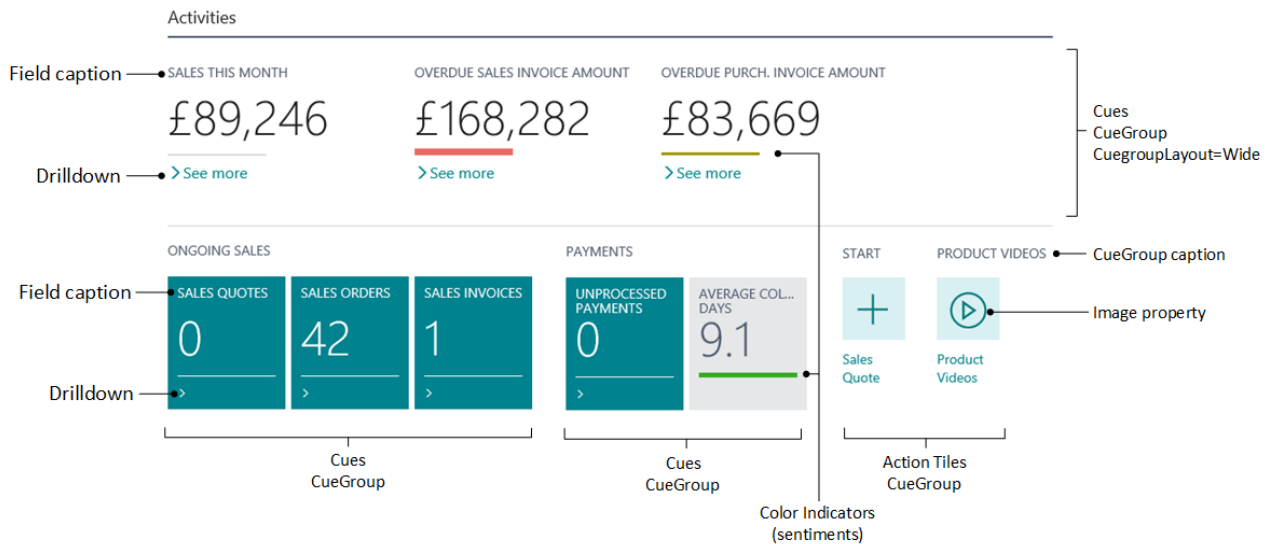
[Pages Overview](#)

[Page Object](#)

Creating Cues and Action Tiles on Role Centers

3/31/2019 • 9 minutes to read

This article provides an overview of Cues and Action tiles, and the tasks involved in creating and customizing them for displaying on Role Centers, as illustrated in the following figure:



Designing Cues

A Cue provides a visual representation of aggregated business data, such as the number of open sales invoices or the total sales for the month. Cues are interactive, meaning that you can select the Cue to drill down to data or open another page, run code, and more. Cues display data that is contained in a table field. This can be raw data or calculated data.

Normal and wide layout

There are two layout options that influence how Cues appear in the client: *normal* and *wide*.

- The *normal* layout displays Cues as tiles. With this layout, Cue groups are automatically arranged to fill in the width of the workspace, which means there can be more than one group horizontally across the workspace.
- The *wide* layout is designed to display large values, such as monetary values. The wide layout gives you a way emphasize a group of Cues. Wide and normal Cue groups can be interleaved. However, wide groups that precede all normal groups will appear in their own section of the workspace, spanning the entire width - providing space for the large values. Wide groups that are placed after normal groups will behave just like the normal layout groups. With this in mind, it is good practice to place Cue groups that use the wide layout, above those that use the normal layout. The wide layout is specified by setting the `CuegroupLayout` property to `wide`.

NOTE

The wide layout is only supported in the Web client.

The `Caption` and `CaptionML` properties of the `cuegroup` control are ignored when the layout is wide.

Supported data types

You can only base Cues on integer and decimal data types. Other data types are not supported and will not display in a Cue.

FlowFields versus normal fields

A Cue can be based on a FlowField or Normal field. If you base the Cue on a FlowField, then you add the logic that calculates the data for the Cue to the [CalcFormula property](#) of the FlowField. If you use a Normal field, then you will typically add the logic that calculates the Cue data to an AL trigger or method. Unlike a FlowField, where data is extracted from tables, a Normal field enables you to extract data from other objects such as queries.

Creating a Cue

The implementation of a Cue involves the following elements:

- A table object with a field that holds the data that is contained in the Cue at runtime.
- A page object that contains the table field and displays the Cue in the client.
- Logic that calculates the data to display in the Cue at runtime.

The logic can consist of a combination of AL code and objects, such as tables, queries, and codeunits. How and where you implement the logic will depend on whether the Cue is based on a FlowField or Normal field and what you want to achieve.

NOTE

The examples in this section will set up a Cue that extracts the number of open sales invoices from the **Sales Header** table.

Create a table for Cue data

The first thing that you must do is to create a table that contains fields that will hold the calculated data to display in the Cues at runtime.

1. Create a table object or use an existing one.
2. Add fields for the Cue data.

For each Cue that you want to display on the page, you must add a **Field** control in the table object. When you add the **Field** control, specify the following properties:

- Set the [Data Type property](#) to **Decimal**, **Integer**, or **Text**, depending on the type of data the Cue will display.
- Set the [FieldClass property](#) to **FlowField** or **Normal**.

If field is a FlowField, then set the `CalcFormula` property to calculate the Cue data. For more information, see [Calculation Formulas and the CalcFormula Property](#).

3. Add a primary key field for FlowFields.

A table must have at least one data field. Because a **FlowField** is based on a calculation, it is not considered an actual data field. Therefore, if the Cue table only includes FlowFields, you must add "dummy" primary key field that does not yield any data.

To add primary key, for example, add a field with the name **Primary Key**, and then set its data type to **Code**.

Example

```

table 50100 SalesInvoiceCueTable
{
    DataClassification = ToBeClassified;

    fields
    {
        field(1;PrimaryKey; Code[250])
        {
            DataClassification = ToBeClassified;
        }
        field(2; SalesInvoicesOpen ; Integer)
        {
            FieldClass = FlowField;
            CalcFormula = count("Sales Header" where("Document Type"=Filter(Invoice), Status=FILTER(Open)));
        }
    }

    keys
    {
        key(PK; PrimaryKey)
        {
            Clustered = true;
        }
    }
}

```

Add Cues to a Page object

After you have a table for holding the Cue data, you create a page that you associate the table, and then add Cue fields on the page. Typically, you will create Card Part type page that will be part of the Role Center page. Cues are arranged into one or more groups on the page. Each group will have its own caption.

1. Create a page object that has the [SourceTable property](#) set to the Cue data table.
2. Add a `cuegroup` control.
3. Under the `cuegroup` control, for each Cue that you want to display, add a `field` control.
4. If you want to set the `cuegroup` to use the wide layout, set the `CuegroupLayout` property to `wide`.

Repeat steps 2-4 to add additional Cue groups.

5. Initialize the Cue fields.

You must initialize the Cue fields on the page. To do this, for example, you can add the following AL code to the [OnOpenPage Trigger](#).

```

    RESET;
    if not get then begin
        INIT;
        INSERT;
    end;

```

Example

```

page 50105 SalesInvoiceCuePage
{
    PageType = CardPart;
    SourceTable = SalesInvoiceCueTable;

    layout
    {
        area(content)
        {
            cuegroup(SalesCueContainer)
            {
                Caption='Sales Invoices';
                // CuegroupLayout=Wide;
                field(SalesCue; SalesInvoicesOpen)
                {
                    Caption='Open';
                    DrillDownPageId="Sales Invoice List";
                }
            }
        }
    }

    trigger OnOpenPage();
    begin
        RESET;
        if not get then begin
            INIT;
            INSERT;
        end;
    end;
}

```

Designing Action tiles

Action tiles promote an action or operation to the user on the Role Center. Action tiles act as links that perform a task or operation, like opening another page, starting a video, targeting an another resource or URL, or running code. They will arrange on the workspace just like that use the normal layout.

Similar to Cues, Actions tile can be grouped together, under a common caption, by using the `cuegroup` control. The difference is that instead adding field controls under the `cuegroup` control, you create Action tiles by adding actions to the `cuegroup` control.

Create an Action tile

1. Develop or locate the functionality that you want to Action tile to perform.

For example, create the page object that you want the Action tile to open, add AL code that you want the Action tile to run, find the URL to the video.

2. Open the page on which you want to display the Action tiles. For example, this could be the page that you created in the previous task.
3. In the location where you want the Action group, add a `cuegroup` control.
4. Configure the control to the desired operation.

For example, if it should open a page, set the control's [RunObject property](#) to the appropriate page. Or, set it to call a function or method.

Example

The following code adds an Action tile that opens **Sales Invoice** page.

```

cuegroup(SalesActionontainer)
{
    Caption='New Sales Invoice';

    actions
    {
        action(ActionName)
        {
            RunObject=page "Sales Invoice";
            Image=TileNew;

            trigger OnAction()
            begin
                end;
        }
    }
}

```

Styling an Action tile

You can use the [Image property](#) on an `action` control to change the look of the Action tile. For Action tiles, the `Image` property supports several standard values that start with the text `Tile`, such as `TileNew` and `TileYellow`. These values change the Action's background color and icon as follows:

- A value that has the format `Tile[color]` will set the Action tile to use the circle icon and a background that is specified by `[color]`. For example, `TileBlue` will display a circle icon in a blue background.
- A value that has the format `Tile[picture]` will set the Action tile to use an icon that is specified by `[picture]` and a neutral background color. For example, `TileCamera` will display a camera icon on the neutral background.

NOTE

If you use a value that is not valid or recognized, the Action tile will default to display the circle icon on the neutral background.

See Also

[FlowFields](#)

[Page Object](#)

[Pages Overview](#)

[Table Object](#)

Designing List Pages

6/25/2019 • 8 minutes to read

The *List* page type displays records from an underlying table, either as rows and columns or as individual tiles.

- [Overview](#)
- [Structure](#)
- [Behavior points](#)
- [Developer tips](#)
- [Designing for devices](#)

You design list pages when you want to provide users with a collection of data, enabling them to get an overview of and find entities to work with, such as customers, vendors, or sales orders. Typically, a list page will link to an associated card page that lets users view or modify specific entities in the list.

There are different ways to incorporate a list page into that application:

- Make the list page available from the navigation of a Role Center page.

This gives users quick access to the page. With this implementation, the list page opens in the content area of the Role Center page, where the Role Center's navigation area is still present and accessible at the top of the page. For more information about Role Centers, see [Designing Role Centers](#).

- Make the list page available from an action on another page.

With this implementation, the list page opens in a separate window in front of the current page.

- Make the list page searchable from the **Tell me what you want to do** feature.

With this implementation, the list page also opens in a separate window. For more information, see [Adding Pages and Reports to Search](#).

Customizing a list pages from the client

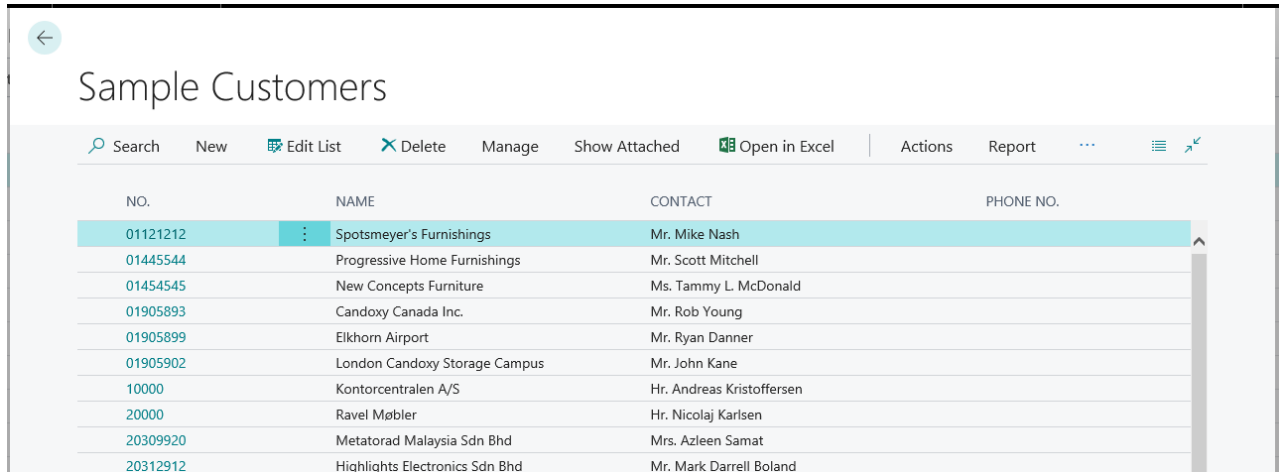
In the client, users can personalize list pages by rearranging or hiding records or FactBoxes as they like. For more information, see [Personalizing Your Workspace](#).

As a developer or administrator, you can use Designer to customize the list page the same way that individual users personalize their own work spaces. The difference is that changes you make are applied to all users. For more information, see [Using Designer](#).

Simple List Page Code Example

5/3/2019 • 2 minutes to read

The AL code in this article creates a simple list page that displays records from an existing table.



NO.	NAME	CONTACT	PHONE NO.
01121212	Spotsmeyer's Furnishings	Mr. Mike Nash	
01445544	Progressive Home Furnishings	Mr. Scott Mitchell	
01454545	New Concepts Furniture	Ms. Tammy L. McDonald	
01905893	Candoxy Canada Inc.	Mr. Rob Young	
01905899	Elkhorn Airport	Mr. Ryan Danner	
01905902	London Candoxy Storage Campus	Mr. John Kane	
10000	Kontorcentralen A/S	Hr. Andreas Kristoffersen	
20000	Ravel Møbler	Hr. Nicolaj Karlsen	
20309920	Metatorad Malaysia Sdn Bhd	Mrs. Azleen Samat	
20312912	Highlights Electronics Sdn Bhd	Mr. Mark Darrell Boland	

For a more detailed explanation of the list page, see [Designing List Pages](#).

```
page 50111 SampleCustomerList
{
    PageType = List;
    ApplicationArea = All;

    // Specifies the page to display records from the Customer table.
    SourceTable = Customer;

    // Makes the page searchable from the Tell me what you want to do feature.
    UsageCategory = Lists;

    // Specifies the card page Sample Customers to be uses for modifying or creating new customer records.
    CardPageId = 50112;

    // Sets the title of the page to Sample Customers.
    Caption = 'Sample Customers';

    layout
    {
        area(Content)
        {
            // Sets the No., Name, Contact, and Phone No. fields in the Customer table to be displayed as
            columns in the list.
            repeater(Group)
            {
                field("No."; "No.")
                {
                    ApplicationArea = All;
                }

                field(Name; Name)
                {
                    ApplicationArea = All;
                }

                field(Contact; Contact)
                {
                    ApplicationArea = All;
                }
            }
        }
    }
}
```

```

        field(Phone; "Phone No.")
        {
            ApplicationArea = All;
        }
    }
}

actions
{
    // Adds an action on the Actions menu of the action bar that opens the page Customer Ledger Entries.
    area(Processing)
    {
        action("Ledger Entries")
        {
            ApplicationArea = All;
            RunObject = page "Customer Ledger Entries";
            trigger OnAction();
            begin

            end;
        }
    }

    // Promotes an action for creating a sales quote to promoted action menu called New.
    area(Creation)
    {
        action("New Sales Quote")
        {
            ApplicationArea = All;
            RunObject = page "Sales Quote";
            Promoted = true;
            PromotedCategory = New;
            Image = NewSalesQuote;
            trigger OnAction();
            begin

            end;
        }
    }

    // Adds an action on the Report menu that opens the Top 10 List report.
    area(Reporting)
    {
        action("Top 10 List")
        {
            ApplicationArea = All;
            RunObject = report "Customer - Top 10 List";
            trigger OnAction();
            begin

            end;
        }
    }
}

```

See Also

[AL Development Environment](#)

[Page Extension Object](#)

[Actions Overview](#)

[Adding Pages and Reports to Tell Me](#)

[Personalizing Your Workspace](#)

[Using Designer](#)

Displaying Data as Tiles

6/25/2019 • 4 minutes to read

In the client, on list type pages (such as `List`, `ListPart`, and `ListPlus`), users have the option to view the page in the tile view. The tile view shows records as tiles (or bricks) instead of as rows. Tiles optimize space and readability of data, and is especially useful for images, like on a page that show items, customers, and contacts. The tile view compresses up to six columns of data. By default, the tile view will display the first six fields of the page's source table. This article describes how you can customize the tile view for list type pages.

Tile view in the client

Users switch between the list and tile view by selecting the **View layout options** icon in the action bar at the top right-hand corner of the page. If tiles contain a media field type, then there are two tile view options: **Tiles** and **Tall Tiles**. The same information is displayed except with **Tall Tiles**, images are larger and display at the top of the tiles.

Tiles are interactive. A context menu is available in the upper right corner. The context menu contains the actions that are defined for the record, just as in the list view. To drill down to a card page for a record, the user selects the tile.

Customizing the tile view in AL

You specify the data that you want shown in the tile view in the source table of the page by adding a `Field Group` that has the name `Brick`:

```
fieldgroups
{
    fieldgroup(Brick; <field 1>, <field 2>, <field 3>, <field 4>, <field 5>, <field 6>)
    {
    }
}
```

You can specify up to six fields.

IMPORTANT

By default, the `Field Group` named `DropDown` is interpreted as `Brick` when a `Brick` definition has not been set. The `DropDown` is typically set on entities such as customer, vendor, and items. For more information, see [Field Groups \(Drop-Down Controls\)](#).

Field layout in tiles

The order of the fields determines how they appear in the layout of the tile, regardless of the order the fields appear in the page object definition. Depending on the number of columns that you define in the `Field Group`, the layout will dynamically change. Up to 6 fields can be displayed in a tile, and therefore, there are six possible layouts as shown in this illustration:

Field 1 (small font)

Field 2 (large font)

Field 4 (secondary field)

Field 6 (secondary field)

Field 3 (large font and right-aligned)

Field 5 (secondary field and right-aligned)

The fields 2 and 3 are shown in a large font and should contain data that identifies the brick, for example, the Customer Name and Balance as you can see in the Customer list in, for example, the Business Central Web client.

Including images in tiles

To display an image in the brick, you include a `Media` data type field in the `Field Group` definition. You do not have to include a field control for the media field in the page object, because the image will be shown in the tile view automatically.

The image will be displayed on the left side of the tile (or at the top in the **Tall Tiles** view), regardless of its position in the `Field Group` definition. If an image does not exist for a certain record, a default picture is displayed instead.

For information including media on records, see [Working With Media on Records](#).

Styling text in tiles

Just as in the list view, the tile view supports the [Style Property](#) and [StyleExpr Property](#) that you apply on the page field controls. These properties, for example, let you mark numbers as favorable or unfavorable.

Example

The following code is a simple example of a table that includes `Field Group` control for displaying data in the tile view of a list page.

```
Table 50100 MyTable
{
    fields
    {
        field(1; Number; Integer)
        {
        }

        field(2; Description; Text[50])
        {
        }
        field(3; Inventory; Integer)
        {
        }
        field(4; Image; Media)
        {
        }
    }

    keys
    {
        key(PK; Number)
        {
        }
    }

    fieldgroups
    {
        fieldgroup(Brick; Number, Description, Inventory, Image)
        {
        }
    }
}

page 50100 MyListPage
{
    PageType = List;
    ApplicationArea = All;
    UsageCategory = Lists;
    SourceTable = BrickTableTest;
```

```
Editable = true;
CardPageId = MyCardPage;

layout
{
    area(Content)
    {
        repeater(GroupName)
        {
            field(Number; Number)
            {
                ApplicationArea = All;
            }
            field(Description; Description)
            {
                ApplicationArea = All;
            }
            field(Inventory; Inventory)
            {
                ApplicationArea = All;
                Style = Attention;
            }
        }
    }
}
```

See Also

[Designing List Pages](#)

[Working With Media on Records](#)

Views

5/21/2019 • 2 minutes to read

Views in Dynamics 365 Business Central are used on list pages to define a different view of the data on a given page. Views can be defined for [Pages](#), [Page Extensions](#), and [Page Customization](#). Views are defined on page extension objects to provide an alternative view of data and/or layout on an existing page, and in views on page customization objects, they can be used to provide an alternative view for a certain profile.

A view offers:

- Filtering on multiple table fields on the source table defined for the page.
- Sorting of the data on multiple table fields, but only in one direction; either *ascending* or *descending*.
- Layout changes, modifying page columns, moving them, etc.

Views are defined directly in code, on the list page that they modify. The defined view or views are available to the user through **Filter Pane** on a page and appear in the sequence that they are defined in code.

Snippet support

Typing the shortcut `tview` will create the basic layout for a view when using the AL Language extension in Visual Studio Code.

Filtering and sorting

You can filter on the data in a view by using the `Filters` property. The following is an example of the syntax:

```
Filters = where ("Balance (LCY)" = filter (> 500), Name = filter ('G*'));
```

For more information, see [Filters](#).

You can sort on the data in a view by using the `OrderBy` property. The following is an example of the syntax:

```
OrderBy = ascending ("Balance (LCY)", Name);
```

For more information, see [OrderBy](#).

NOTE

All filters are applied to the table field(s), not the page field(s), which allows filtering on a table field not shown on the page.

View example

The following example shows a page customization of the **Customer List** page, which is available for a specific role center only; the **My Role Center**. Change the role center view under **My Settings**. The definition of the view adds a caption which is displayed on the left side in the UI. The view sorts the customer balance in ascending mode and the view modifies the layout by moving the customer balance first and adding a freeze column after it.

```

profile MyProfile
{
    Description = 'My Role Center';
    RoleCenter = "Order Processor Role Center";
    Customizations = MyCustomization;
}

pagecustomization MyCustomization customizes "Customer List"
{
    views
    {
        addfirst
        {
            view(BalanceLCY)
            {
                Caption = 'Ordered Balance LCY';
                OrderBy = ascending ("Balance (LCY)");

                layout
                {
                    movefirst(Control1; "Balance (LCY)")

                    modify(Control1)
                    {
                        FreezeColumn = "Balance (LCY)";
                    }
                }
            }
        }
    }
}

```

Limitations

In general, views can in several ways be compared to page customizations. These are the limitations of views:

- For views you can modify the same control properties as for page customization objects independently of where the view has been defined (page, page extension, or page customization level). This is validated by the compiler.
- It is not possible to use variables or methods in a view. When writing client-side expressions for properties like **Visibility**, it will only be possible to use constant values or table field references. This is validated by the compiler.
- It is not possible to create new controls for a page from a view.

See Also

[AL Development Environment](#)

[Developing Extensions](#)

[Page Object](#)

[Page Extension Object](#)

[Page Customization Object](#)

Adding Filter Tokens

3/31/2019 • 2 minutes to read

In the client, when filtering lists using the filter pane, users can enter filter tokens, which are special words that resolve to one or more values. This powerful feature makes filtering easier by reducing the need to navigate to other pages to look up values to enter as filter criteria.

There are several useful filter tokens available in Business Central. For example, entering **%mycustomers** in a **Customer No.** field will resolve to the set of customers in the user's **My Customers** list such as `1001|1002`, making it easy to find relevant sales orders for customers 1001 and 1002.

You can add custom filter tokens and make these available in any language and across the application. To add your custom filter token, you need to define the token word that users will enter as filter criteria, and define a handler that resolves the token to a concrete value at runtime.

Defining the token word and the handler

To create the desired token word, start by defining a multi-language text string for your word. Subscribe to the `OnBeforeMakeTextFilter` or `OnAfterMakeTextFilter` events associated with the `MakeTextFilter` method from the `TextManagement` codeunit.

In the event subscriber, if the value of the `TextFilterText` parameter contains the token string proceed to process its value and construct the final filter string. If the filter string must contain multiple values, you must handle the operators that join them together, by inserting the `|` filter symbol (OR operation). Complete the operation by setting the value of the `TextFilterText` parameter to the value of the final filter string.

TIP

Filter criteria will often contain symbols along with filter tokens. It is recommended that you only modify the filter token you have introduced and preserve the rest of the filter string.

Example

This example shows how you can use the guidelines above to create the **%MYTOKEN** filter token. This will return a filter with the accounts marked as favorite by the user.

NOTE

To keep this sample short and simple, the entire filter string is overwritten.

```

codeunit 50101 MyAccountFilterTokenSimple
{
    [EventSubscriber(ObjectType::Codeunit, Codeunit::TextManagement, 'OnAfterMakeTextFilter', '', true, true)]
    local procedure FilterMyAccountsOnAfterMakeTextFilter(var Position: Integer; var TextFilterText: Text)
    var
        MyAccountTxt: Label 'MYTOKEN';
        MyAccount: Record "My Account";
        MaxCount: Integer;
    begin
        if StrLen(TextFilterText) < 3 then
            exit;
        if StrPos(UpperCase(MyAccountTxt), UpperCase(TextFilterText)) = 0 then
            exit;

        MaxCount := 2000;
        MyAccount.SetRange("User ID", UserId);
        if MyAccount.FindSet() then begin
            MaxCount -= 1;
            TextFilterText := MyAccount."Account No.";
            if MyAccount.next <> 0 then
                repeat
                    MaxCount -= 1;
                    TextFilterText += '|' + MyAccount."Account No."
                until (MyAccount.Next = 0) or (MaxCount <= 0);
            end;
        end;
    end;
}

```

To try it out in the client, open the Charts of Accounts page, filter on No. field, and type in a substring that starts the same way with the chosen token word, like **%MYTO**.

See Also

[Sorting, Searching and Filtering Lists](#)

Designing Card and Document Pages

3/31/2019 • 7 minutes to read

The *card* page type displays selected fields from an underlying table. The *document* page type is very similar in structure to the card page, but in addition to fields, it also includes a part that includes another page, called a sub-page.

- [Overview](#)
- [Structure](#)
- [Behavior points](#)
- [Developer tips](#)
- [Designing for devices](#)

Card pages

You design card pages when you want to enable users to view, create, and modify records (master and reference data) in a table, such as a customer, vendor, or item.

Document pages

Design document pages when you want to represent a transaction or other important event in the domain of business. Document pages are the computerized counterpart to paper-based documents, such as quotes, invoices, orders, and so on. As such, document pages often have associated workflow or audit trail requirements.

Associate with a list page

Both page types are typically associated with list pages (like the customers or sales orders list) that uses the same table as their source. From the list page, users can select a record and open it the card or document page for viewing and editing.

Customizing a card and document pages from the client

In the client, users can personalize card pages by rearranging or hiding content as they like. For more information, see [Personalizing Your Workspace](#).

As a developer or administrator, you can use Designer to customize a card and document page the same way that individual users personalize their own work spaces. The difference is that changes you make are applied to all users. For more information, see [Using Designer](#).

Simple Card Page Code Example

4/4/2019 • 2 minutes to read

The AL code in this article creates a simple card page that displays records from an existing table.

The screenshot shows a card page titled '01121212 · Spotsmeyer's Furnishings'. At the top, there is a navigation bar with a back arrow, the text 'SAMPLECUSTOMERCARD', and icons for edit, add, and delete. Below the title, there is a horizontal menu with options: 'New', 'Customer', 'Show Attached', 'Actions', 'Report', and 'Less options'. The main content area is divided into two sections: 'General' and 'Contact'. The 'General' section contains two fields: 'No.' with the value '01121212' and 'Name' with the value 'Spotsmeyer's Furnishings'. The 'Contact' section contains two fields: 'Contact' with the value 'Mr. Mike Nash' and 'Phone No.' which is empty.

For a more detailed explanation of the list page, see [Designing Card and Document Pages](#).

```
page 50112 SampleCustomerCard
{
    PageType = Card;
    ApplicationArea = All;
    UsageCategory = Administration;
    SourceTable = Customer;

    //Defines the names for promoted categories for actions.
    PromotedActionCategories = 'New,Process,Report,Manage,New Document,Request Approval,Customer';

    layout
    {
        area(Content)
        {
            //Defines a FastTab that has the heading 'General'.
            group(General)
            {
                field("No."; "No.")
                {
                    ApplicationArea = All;
                }
                field(Customer; Name)
                {
                    ApplicationArea = All;
                }
            }
        }

        //Defines a FastTab that has the heading 'Contact'.
        group(Contact)
        {
            field(Name; Contact)
            {
                ApplicationArea = All;
            }
            field(Phone; "Phone No.")
            {
                ApplicationArea = All;
            }
        }
    }
}
```

```

    }
  }
}

actions
{
  area(Processing)
  {
    //Defines action that display under the 'Actions' menu.
    action("New Sales Quote")
    {
      ApplicationArea = All;
      RunObject = page "Sales Quote";
      Promoted = true;
      PromotedCategory = New;
      Image = NewSalesQuote;
      trigger OnAction();
      begin

      end;
    }
    action("Banks Account")
    {
      ApplicationArea = All;
      RunObject = page "Customer Bank Account List";
      Promoted = true;

      //Promotes the action to the category named 'Customer'.
      PromotedCategory = Category7;
      Image = BankAccount;
      trigger OnAction();
      begin

      end;
    }
  }
  area(Reporting)
  {
    //Defines action that display under the 'Report' menu.
    action("Statement")
    {
      ApplicationArea = All;
      RunObject = codeunit "Customer Layout - Statement";
      trigger OnAction();
      begin

      end;
    }
  }
}
var
  myInt: Integer;
}

```

See Also

[AL Development Environment](#)

[Page Extension Object](#)

[Actions Overview](#)

[Adding Pages and Reports to Search](#)

Personalizing Your Workspace

Using Designer

Adding a FactBox to a Page

6/4/2019 • 2 minutes to read

A FactBox is the area that is located on the right-most side of a page and it is divided into one or more parts that are arranged vertically. This area is used to display content including other pages, charts, and system parts such as Microsoft Outlook, Notes, and Record Links. Typically, you can use a FactBox to display information that is related to an item on the main content page. For example, on a page that shows a sales order list, you can use a FactBox to show sell-to customer sales history for a selected sales order in the list as shown below.

The screenshot shows the Dynamics 365 Business Central interface for a sales order (S-ORD101006) for Adatum Corporation. The main content area displays the sales order details, including customer information, posting date, and a table of sales order lines. The FactBox area on the right contains the following sections:

- Attachments:** Documents (0)
- Sell-to Customer Sales History:** A table showing various sales metrics.
- Customer Details:** Information about the customer, including name, phone number, email, and credit limit.
- Sales Line Details:** (Section header visible)

Metric	Value
Ongoing Sales Quotes	0
Ongoing Sales Blanket Orders	0
Ongoing Sales Orders	3
Ongoing Sales Invoices	2
Ongoing Sales Return Orders	0
Ongoing Sales Credit Memos	0
Posted Sales Shipments	33
Posted Sales Invoices	33
Posted Sales Return Receipts	0
Posted Sales Credit Memos	0

Field	Value
Customer No.	10000
Name	Adatum Corporation
Phone No.	
Email	robert.townes@contoso.com
Fax No.	
Credit Limit (\$)	0.00
Available Credit (\$)	0.00
Payment Terms Code	1M(8D)
Contact	Robert Townes

The following list highlights a few categories of FactBoxes:

- Show related records/fields which are modeled as ListParts or CardParts.
- Show related KPIs which are modeled as CardParts with charts or Cues. For more information, see [Designing Role Centers](#).
- Visualize related data or display from external sources which are modeled as CardParts containing a Client AddIn. For example, Bing maps, PowerBI, Microsoft Social Engagement, and more.

Adding a FactBox area to a page

You define the FactBox by adding a FactBox area container control to the page. The FactBox area control acts as a placeholder to which you can add different parts for the FactBox. You can add a FactBox area control on the following page types.

- Card

- Document
- List
- ListPlus
- Navigate
- Worksheet

NOTE

Only one FactBox area control is allowed on a page.

WARNING

You can add a part to the FactBox area that displays an existing page of the CardPart or ListPart type only. If you attempt to use another page type, you will get an error.

Example

```
page 50100 "Simple Customercard Page"
{
    PageType = Card;

    layout
    {
        area(FactBoxes)
        {
            part(MyPart; "Acc. Sched. KPI Web Srv. Lines")
            {
                ApplicationArea = All;
                SubPageView = SORTING ("Acc. Schedule Name");
            }
            systempart(Links; Links)
            {
                ApplicationArea = All;
            }
            systempart(Notes; Notes)
            {
                ApplicationArea = All;
            }
        }
    }
}
```

TIP

When used on Lists, Factboxes can be used to show information about the entire list, or more contextually about the user's current selection; the currently selected rows. You can control the filter which gets passed to the FactBox that determines its contextual contents.

Filtering data that is displayed on a page in a FactBox

In many cases, you want to change the content that is displayed on the page in the FactBox based on the content of the main page. For example, if the main page is a Customer List, you can have a FactBox that includes the Customer Details page that shows information about a customer. When a user selects a customer in the Customer List, the Customer Details page displays information about the selected customer. To implement this functionality,

you set up a table filter that associates a field in the table that is used by the Customer Details page with a field in the table that is used by the Customer List page, as shown in the example below. You can also filter on a constant value or set of conditions.

Example

```
page 50101 "Simple Customerlist Page"
{
    PageType = List;
    SourceTable = Customer;

    layout
    {
        area(content)
        {
            repeater(Control)
            {
                field("No."; "No.")
                {
                    ApplicationArea = All;
                }
            }
        }

        area(FactBoxes)
        {
            part(CustomerList; "Customer Details FactBox")
            {
                ApplicationArea = All;
                SubPageLink = "No." = FIELD ("No.");
            }
        }
    }
}
```

See Also

[Pages Overview](#)

[Page and Page Extension Properties Overview](#)

[Designing Role Centers](#)

[Using Designer](#)

Copyright 2016 Microsoft Corporation. All rights reserved.

[Actions Overview](#)

Adding Pages and Reports to Tell me

3/31/2019 • 3 minutes to read

The Business Central client includes the **Tell me** feature that lets users find objects and online help articles by entering search terms. When you have added a page or a report in your extension, you most likely want it to be discoverable to users in **Tell me**. In AL, you make a page or report searchable from **Tell me** by setting the [UsageCategory](#) property in code. The **UsageCategory** setting will make the page or report searchable, and the value chosen for the setting will further sub categorize the item.

TELL ME WHAT YOU WANT TO DO

cust

On current page (Business Manager)

Customer

Register a new customer.

Register Customer Payments

Process your customer payments by matching amounts received on your bank account wi...

Go to Pages and Tasks

Show all (46)

> Customers

Lists

> Customer Disc. Groups

Administration

> Customer Order Status

Tasks

Go to Reports and Analysis

Show all (50)

Customer Labels

Reports and Analysis

Customer Listing

Reports and Analysis

Customer Register

Reports and Analysis

Documentation

Show all (20)

Transferring and Posting Cost Entries

Before you define cost allocations, you must understand where cost entries come from.

Manually Adjust the Costs of Items

You can adjust the inventory valuation of an item using the FIFO or Average costing meth...

Managing Inventory Costs

Cost management, also referred to as "costing", is concerned with recording and reportin...

Get from Microsoft AppSource

Show all (48)

dynamic commerce Shipping Costs

Manage shipping cost directly in Dynamics 365 Business Central.

Advanced Inventory Count

Simplify inventory counts with comprehensive data entry, reconciliation, posting and anal...

Cash Basis Accounting

Cash Basis Accounting

Tell me finds pages and reports by searching the captions that are specified on page and report objects by the [CaptionML property](#).

Working with the UsageCategory property

When you create a [Page](#) or a [Report](#), you add the [UsageCategory Property](#). If the **UsageCategory** is set to **None**, or if you do not specify **UsageCategory**, the page or report will not show up when you search in Dynamics 365 Business Central.

UsageCategory property values

The values for the **UsageCategory** property are listed below. The sub category will help the user navigate through the search results and it is a best practice to be consistent when categorizing the pages and the reports that you add. A consistent approach will help guiding the user and improve productivity.

VALUE	DESCRIPTION
None	The page or report is not included in search.
Lists	The page or report is listed as Lists under the Pages and Tasks category.
Tasks	The page or report is listed as Tasks under the Pages and Tasks category.
ReportsAndAnalysis	The page or report is listed as Reports and Analysis under the Reports and Analysis category.
Documents	The page or report is listed as Documents under the Reports and Analysis category.
History	The page or report is listed as Archive under the Reports and Analysis category.
Administration	The page or report is listed as Administration under the Pages and Tasks category.

Adding additional search terms

You can specify other words or phrases that can help users find a page or report by using the [AdditionalSearchTerms](#) and [AdditionalSearchTermsML](#) properties. If the page or report is searchable by **Tell me** (that is, the **UsageCategory** property is set a value other than `None`), the search terms specified by these properties are used in addition to the caption of the page or report. These properties are useful when the caption does not always reflect what users will look for. A good example of this in Business Central is pages and reports associated with **Item**. Users unfamiliar with Business Central might look for 'product' or 'merchandise' instead of 'item'.

NOTE

For Business Central on-premises, the Business Central Web Server configuration file (navsettings.json) includes a setting called `UseAdditionalSearchTerms` that enables or disables the use of additional search terms by the **Tell me**. For more information, see [Configuring Business Central Web Server Instances](#).

Example

The following example creates a `SimpleItemList` page and sets a `UsageCategory` property to the page, so that the `SimpleItemList` page is discoverable through search using the **Tell me** feature. Also, the example sets the `AdditionalSearchTerms` property to add two search terms for the page.

```
page 50210 SimpleItemList
{
    PageType = List;
    SourceTable = Item;
    UsageCategory = Lists;
    AccessByPermission = page SimpleItemList = X;
    ApplicationArea = All;
    AdditionalSearchTerms = 'product, merchandise';

    layout
    {
        area(content)
        {
            group(General)
            {
                field("No."; "No.") {}
                field(Name; Name) {}
                field(Description; Description) {}
            }
        }
    }
}
```

Optional settings

In addition to making a page or report searchable, you can control the access of an object by providing **Read**, **Insert**, **Modify**, **Delete**, and **Execute** (RIMDX) permissions by adding the [AccessByPermission property](#). Likewise, control the application area access on the specified object by adding the [ApplicationArea Property](#).

The **AccessByPermission** property and **ApplicationArea** property are the optional settings, which can be applied with the **UsageCategory** property. These settings are used to set restrictions on an object when you enable the Search functionality.

Working in the Dynamics NAV Development Environment

If you are using the Dynamics NAV Development Environment, you can also set `UsageCategory`, `AdditionalSearchTerms`, `AccessByPermission`, and `ApplicationArea` properties on pages and reports to control their search.

After you change these properties by using the Dynamics NAV Development Environment, before the changes take effect in the client, you must run **Build Object Search Index** from the **Tools** menu.

See Also

[Adding Menus to the Navigation Pane](#)

[UsageCategory Property](#)

[Page Object](#)

[Report Object](#)

[AL Development Environment](#)

Field Arrangement on FastTabs

3/31/2019 • 2 minutes to read

FastTabs in Dynamics 365 Business Central allow users to find key information on a page by displaying the data in separate groups. This topic describes how individual fields are arranged on a FastTab and ways that you can change the layout.

How fields are arranged on a FastTab of a page

By default, a FastTab is divided into two columns for containing fields. Fields are automatically distributed between the left and right columns based on their order in Page Layout and height on the rendered page. Starting with the first field in the Page Layout and working downward, fields are positioned in the left column and then in the right column so that the area that is occupied by the fields in each column is as equal as possible. Field captions are positioned to the left of fields.

EDIT - CUSTOMER CARD

10000 · Adatum Corporation

General

Name	Adatum Corporation	Total Sales	78,771.10
Balance (\$)	0.00	Costs (\$)	40,255.70
Balance Due (\$)	0.00	CFDI Purpose	--
Credit Limit (\$)	0.00	CFDI Relation	--
Blocked			

Address & Contact >

Invoicing >

Payments >

Shipping >

Statistics >

Special Prices & Discounts

LINE TYPE	SALES TYPE	TYPE	CODE	UNIT OF MEASURE CODE	MINIMUM QUANTITY	LINE DISCOUNT %	UNIT PRICE	STARTING DATE	ENDING DATE
-----------	------------	------	------	----------------------	------------------	-----------------	------------	---------------	-------------

FastTabs

Promoted Fields

Show more

Grouping fields on a FastTab

By using the Group subtype control in a FastTab, you can include fields on a FastTab in separate groups. This gives you control over how fields are distributed between the left and right columns. When you group fields on a FastTab, the groups and not the individual fields are distributed evenly between the left and right columns.

FastTabs replace tabs and enable you to place data in groups on card or task pages. If a group is expanded you see all the fields in a group. If it is collapsed you just see the summary line. The summary line is the header that displays a name for the group, such as 'Communication' and can include promoted fields such as 'E-mail'. Promoting fields to the summary line enables you to present key information to the user, even if the control is collapsed. You can also specify fields that only appear when the users selects the **Show more** action on the FastTab. Promoting a field or displaying it only when **Show more** is selected is specified by the [Importance property](#) of the field.

Organizing data using FastTabs helps users to find key information quickly, while at the same time giving an overview of areas that otherwise would remain hidden. For example, the customer card page displays customer information in the following categories: General, Communication, Invoicing, Payments, Shipping, and Foreign Trade. Each category is a separate group that can be expanded or collapsed, making it easier for users to focus on one area at a time. On task pages, a FastTab typically represents a single step in a task.

Manually arranging fields in multiple rows and columns

Using the GridLayout or FixedLayout controls, you can arrange fields in multiple rows and columns in a grid-like

format. For more information, see [Arranging Fields Using Grid and Fixed Controls](#).

See Also

[Arranging Fields Using Grid and Fixed Controls](#)

[Pages Overview](#)

[Using Designer](#)

[Table in Dynamics 365 Business Central](#)

Arranging Fields in Rows and Columns Using the Grid Control

4/24/2019 • 2 minutes to read

By default, fields in a FastTab are arranged automatically in two columns that are based on the number of fields. For more information, see [Field Arrangement on a Fasttab](#). You can use a Grid control or a Fixed control to arrange fields in rows and columns on a page and design it to look like a grid-like format or a matrix-like format. To understand the differences between the two controls to help you determine which control to use, see [Comparing Grid and Fixed controls](#).

NOTE

Grid control for arranging page fields is partially supported.

Using the **Grid** control, you can arrange the fields manually in one or more rows and columns. The **Grid** control gives you the following options:

- Set up your grid row-by-row or column-by-column.
- Span a field across multiple rows and columns.
- Show or hide field captions.

Setting-up fields in rows and columns in a FastTab

To set up a grid in row-by-row or column-by-column format, you define the **Grid** control in a FastTab of a page. You must define the **Grid** control in a group and specify how you want to arrange the fields by using the **GridLayout** property. For more information, see [GridLayout Property](#).

Example

The following example demonstrates how to structure a page in a grid-like format.

```

page 50113 "Customers Page"
{
    PageType = Card;
    SourceTable = Customer;
    layout
    {
        area(content)
        {
            group(General)
            {
                grid(MyGrid)
                {
                    group("General Info")
                    {
                        field("No."; "No.")
                        {
                            ApplicationArea = All;
                        }
                        field(Name; Name)
                        {
                            ApplicationArea = All;
                        }
                        field("E-Mail"; "E-Mail")
                        {
                            ShowCaption = false;
                            ApplicationArea = All;
                        }
                    }
                    group("Address Details")
                    {
                        grid(Grid2)
                        {
                            group(GridForm)
                            {
                                field(Address; Address)
                                {
                                    ApplicationArea = All;
                                }
                                field("Post Code"; "Post Code")
                                {
                                    ApplicationArea = All;
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

```

Setting fields to span multiple rows and columns

You can set a field to span multiple rows or columns. When you set a field to span multiple rows, the field occupies the cells in the rows below it, and existing fields in the occupied cells are moved to the right. When you set a field to span multiple columns, the field occupies the cells in the columns to the right, and existing fields in the occupied cells are moved to the right. You can also set a field to span multiple rows and columns.

IMPORTANT

The Dynamics 365 Business Central web client does not support row and column spanning for fields. If the page displays in the Dynamics 365 Business Central web client, the fields appear without spanning.

To set a field to span rows and columns

When you set the **Grid** control, the fields of that group can be set to span rows or columns.

- To set a field to span one or more rows, set the value of the **RowSpan** property to the number of rows. For more information, see [RowSpan Property](#).
- To set a field to span one or more columns, set the value of the **ColumnSpan** property to the number of columns. For more information, see [ColumnSpan Property](#).

NOTE

The **RowSpan** and **ColumnSpan** properties on fields in the grid layout are not supported in the Dynamics 365 Business Central web client. The **Rows** layout on the grid control itself is not supported.

Hiding field captions

You can hide the caption of a group or a field. To hide the caption of a field, set the value of the **ShowCaption** property to **false**. For more information, see [ShowCaption Property](#).

See Also

[Field Arrangement on FastTabs](#)

[Arranging Fields Using Grid and Fixed Controls](#)

[Arranging Fields in Rows and Columns Using the Fixed Control.](#)

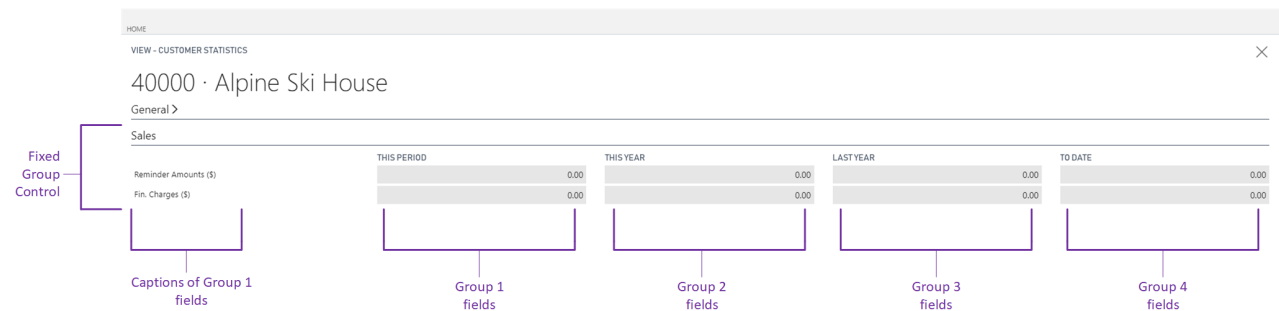
Arranging Fields in Rows and Columns Using the Fixed Control

3/31/2019 • 3 minutes to read

By default, fields on a FastTab are arranged automatically in two columns based on the number of fields. For more information on how the fields are placed on a page, see [Field Arrangement on a Fasttab](#). To manually arrange fields, you can either use a Grid control to design the page to look like a grid-like format, or a Fixed control to design the page to look like a matrix-like format. To understand the differences between the two controls to help you determine which control to use, see [Comparing Grid and Fixed controls](#).

How Fixed control works

You use the Fixed control to arrange page fields in rows and columns to form a matrix-like layout except that the Fixed control contains a specific number of fields, and a matrix can contain an unspecified number of fields. A Fixed group control is typically used to display statistical data. The following illustration shows an example of a page that uses a Fixed control to show sales totals for different time periods.



You can also use a Fixed control to display information in the details section of a Worksheet page. If you are using the CRONUS International Ltd. demonstration database, then you can see examples of these uses in page 151, Customer Statistics, and page 40, Item Journal.

Adding fields

You can add fields directly in the Fixed control. However, when you add fields directly in the Fixed control, all the fields will display in an equal size and the larger fields will get compressed. The following illustration shows the resulting field layout on a page.



Grouping fields in a Fixed control

By placing the fields in a Fixed control throughout a group control, you can define separate rows and columns to create a matrix-like arrangement. The group control caption appears as the column header, and the field control captions appear as the row headers. If you add two more group controls that contain fields, then the layout on the page will display like a table format.

Example

The following AL code uses **Fixed** control to display four fields on a page inside the group called Fixed Layout.

```

page 50114 "Fixed Control Example"
{
    layout
    {
        area(content)
        {
            group("Fixed Layout")
            {
                fixed(DefiningFixedControl)
                {
                    group("Group Caption")
                    {
                        field("Field 1"; "Field 1")
                        {
                            ApplicationArea = All;
                        }
                        field("Field 2"; "Field 2")
                        {
                            ApplicationArea = All;
                        }
                        field("Field 3"; "Field 3")
                        {
                            ApplicationArea = All;
                        }
                        field("Field 4"; "Field 4")
                        {
                            ApplicationArea = All;
                        }
                    }
                }
            }
        }
    }
    var
        "Field 1": Integer;
        "Field 2": Integer;
        "Field 3": Integer;
        "Field 4": Integer;
}

```

The following illustration shows the resulting field layout on a page.

Fixed Layout	
Group caption	
Field 1 caption	Field 1
Field 2 caption	Field 2
Field 3 caption	Field 3
Field 4 caption	Field 4

The group control caption appears as the column header, and the field control captions appear as the row headers. If you add two more group controls that contain fields, then the layout on the page will resemble the following illustration.

NOTE

Only the captions of fields in the first column define the row headings. Therefore, only the field captions for the first group control appear. The field captions in other group controls are ignored.

Fixed Layout caption			
	Group 1 caption	Group 2 caption	Group 3 caption
Field 1 caption	Field 1	Field 1	Field 1
Field 2 caption	Field 2	Field 2	Field 2
Field 3 caption	Field 3	Field 3	Field 3
Field 4 caption	Field 4	Field 4	Field 4

Using multiple Fixed controls

You can also set up more than one Fixed control in a group control. The page area will then divide the fields into two columns that contain the separate Fixed fields. For example, the following illustration shows the page layout if you have four Fixed controls.

Fixed Layout 1	Fixed Layout 2
Field 1 Field 2 Field 3 Field 4	Field 1 Field 2 Field 3 Field 4
Fixed Layout 3	Fixed Layout caption 4
Field 1 Field 2 Field 3 Field 4	Field 1 Field 2 Field 3 Field 4

NOTE

The fields in the Fixed controls in the illustration are not in a group control. If they were in a group control, then they would follow the same principle as described in the previous section about how to group fields.

IMPORTANT

In previous versions, having a Fixed control directly under a content area was supported. However, in Dynamics 365 Business Central, you must make sure that the Fixed control is nested in a Group control. For more information, see [Supported Structure for Using the Grid and Fixed Controls](#).

Editing fields in a Fixed control

Fields in a fixed layout are not editable even if the **Editable** property is set to **true**. However, if the field drills down to a page where the field source is defined, then you can modify the field. For more information, see [Editable Property](#).

See Also

[Field Arrangement on a FastTab](#)

[Pages Overview](#)

[Arranging Fields Using Grid and Fixed Controls](#)

[Arranging Fields in Rows and Columns Using the GridLayout Control](#)

Field Groups (Drop-Down Controls)

6/25/2019 • 2 minutes to read

A field group in table or table extension objects defines the fields to display in a drop-down control on pages that use the table.

NOTE

A field group can also be used to specify fields that display when list type pages are shown in the tile view. For more information, see [Displaying Data as Tiles](#).

In a table object, you define field groups by first adding a `fieldgroups` control, and then adding one or more `fieldgroup(<Name>; <Field>` keyword for each group, where:

- `<Name>` can be either `DropDown`, for adding fields to the drop-down control.
- `<Field>` is a comma-separated list of the fields, by name, to include in the group.

```
fieldgroups
{
  fieldgroup(Dropdown; Field1, Field2)
  {
  }
  fieldgroup(Brick; Field1, Field2)
  {
  }
}
```

NOTE

The `fieldgroups` keyword cannot be inserted before the `key` control.

In a table extension object, the `fieldgroups` control allows you to add more fields to a field group defined for the table object. This can be done by using the `addlast(<name>; <field>)` keyword.

WARNING

The server will remove the duplicates, if multiple extensions attempt to add the same field more than once. A field can only be added to the field group once.

Define fields for a drop-down control

You define a field to include in a drop-down control by using the `DropDown` field group name in the keyword.

The following example illustrates how to add the field

```
tableextension 50100 CustomerExercise extends Customer
{
    fields
    {
        field(50100); "V02Max"; Decimal) { }
    }

    fieldgroups
    {
        addlast(DropDown; V02Max) { }
    }
}
```

See Also

[Debugging in AL](#)

[Developing Extensions](#)

[Microsoft .NET Interoperability from AL](#)

CalcFields, CalcSums, FieldError, FieldName, Init, TestField, and Validate Methods

5/3/2019 • 5 minutes to read

The following methods perform various actions on fields:

- CalcFields
- CalcSums
- FieldError
- FieldName
- Init
- TestField
- Validate

CalcFields method

CalcFields updates FlowFields. FlowFields are automatically updated when they are the direct source expressions of controls, but they must be explicitly calculated when they are part of a more complex expression. For more information about Flowfields, see [FlowFields](#).

CalcFields has the following syntax.

```
[Ok :=] Record.CalcFields(Field1, [Field2],...)
```

When you use FlowFields in AL methods, you must use the CalcFields method to update them.

In the following example, the SETRANGE method sets a filter and then the CalcFields method calculates the Balance and Balance Due fields by using the current filter and performing the calculations that are defined as the CalcFormula properties of the FlowFields. This example requires that you create the following variable.

VARIABLE	DATA TYPE	SUBTYPE
Customer	Record	Customer

```
Customer.Get('01454545');  
Customer.SetRange("Date Filter",0D,TODAY);  
Customer.CalcFields(Balance,"Balance Due");  
Message('The Balance is %1 and your Balance Due is %2',Customer.Balance,Customer."Balance Due");
```

The following message is displayed:

The Balance is 342,529.44 and your Balance Due is 342,529.44

CalcSums method

CalcSums calculates the sum of one or more fields that are SumIndexFields in the record.

CalcSums has the following syntax.

```
[Ok :=] Record.CalcSums (Field1, [Field2],...)
```

For CalcSums, a key that contains the SumIndexFields must be selected as the current key. Similar to CalcFields, CalcSums uses the current filter settings when it performs the calculation.

In the following example, an appropriate key is selected, some filters are set, the calculation is performed and then a message is displayed. This example requires that you create the following variable.

VARIABLE	DATA TYPE	SUBTYPE
custledgerentry	Record	Cust. Ledger Entry

```
custledgerentry.SetCurrentKey("Customer No.");  
custledgerentry.SetRange("Customer No.", '10000', '50000');  
custledgerentry.SetRange("Posting Date", 0D, TODAY);  
custledgerentry.CalcSums("Sales (LCY)");  
Message('%1 calculated sales', custledgerentry."Sales (LCY)")
```

FieldError method

FieldError triggers a run-time error after it displays a field-related error message.

FieldError has the following syntax.

```
Record.FieldError(Field, [Text])
```

This method is very similar to the Error method. However, in the FieldError method, if the name of a field is changed, for example, translated to another language, in the Table Designer, the message from the FieldError method will reflect the current name of the field.

The following examples show how to use the FieldError method. These examples require that you create the following variable.

VARIABLE	DATA TYPE	SUBTYPE
Item	Record	Item

```
Item.Get('70000');  
IF Item.Class <> 'HARDWARE' THEN  
    Item.FieldError(Class);
```

If item 70000 has a Class other than HARDWARE, then you receive the following error message:

Class must not be OTHER in Item No. = '70000'.

If the text or code field contains the empty string, then you receive the following error message:

You must specify Class in Item No.= '70000'.

If the field is a numeric field and is empty, it is treated as if it contains the value 0 (zero), and then you receive the following error message:

Class must not be 0 in Item No.= '70000'.

You can change the default text that is displayed in the error message. The following example shows how to use the

FieldError method and change the default text. This example requires that you create the following variable.

VARIABLE	DATA TYPE
Class	Code

```
if Item.Class < '4711' then  
    Item.FieldError(Class,'must be greater than 4711');
```

The following error message is displayed:

Class must be greater than 4711 in Item No.='70000'.

FieldName

FieldName returns the name of a field. It has the following syntax.

```
Name := Record.FieldName(Field)
```

You could just use the name of the field. However, using FieldName lets you create messages that always contain the name of the field, even if the name of the field is changed.

This example shows how to use FieldName together with FieldError.

```
FieldError(  
    Quantity,'must not be less than ' + FieldName("Quantity Shipped"));
```

Init

Init initializes a record. It has the following syntax.

```
Record.Init()
```

If a default value for a field has been defined by using the **InitValue** property, this value is used for the initialization. Otherwise, the default value of each data type is used.

NOTE

Init does not initialize the fields of the primary key.

TestField method

TestField tests whether a field contains a specific value. It has the following syntax.

```
Record.TestField(Field, [Value])
```

If the test fails, that is, if the field does not contain the specified value, an error message is displayed and a run-time error is triggered. This means that any changes that were made to the record are discarded. If the value that you test against is an empty string, the field must have a value other than blank or 0 (zero).

The following example tests the Language Code field for customer number 10000 in the Customer table and tests

whether the Language Code is ZX. This example requires that you create the following variable.

VARIABLE	DATA TYPE	SUBTYPE
customer	Record	Customer

```
customer.Get('10000')  
customer.TestField("Language Code", 'ZX');
```

Validate method

Validate calls the OnValidate trigger of a field. It has the following syntax.

```
Record.Validate(Field [, NewValue])
```

When you enter an account number in a ledger, code in a table trigger is executed to transfer the name of the account from the chart of accounts. If you enter an account number in a batch job, the code which transfers the name of the account is not automatically executed. The following example executes the appropriate field-level trigger code. This example requires that you create the following variable.

VARIABLE	DATA TYPE	SUBTYPE
GeneralLedgerEntry	Record	G/L Entry

```
GeneralLedgerEntry.Validate("G/L AccountNo", '100');
```

This corresponds to the following code.

```
GeneralLedgerEntry."G/L AccountNo" := '100';  
GeneralLedgerEntry.Validate("G/L AccountNo");
```

The Validate method is useful for centralizing processing, which makes your application easier to maintain.

For example, if the OnValidate trigger of the Total Amount field performs a calculation that uses values from three other fields as operands, the calculation must be performed again if the contents of any one of these fields changes. You should avoid entering the calculation formula in the OnValidate triggers of each field because this can create errors if the calculation formula has to be changed later and you have to update the code in all the triggers. Instead, you should enter the calculation formula in the OnValidate trigger of only one of the fields and call this trigger code from the OnValidate triggers of the other fields.

See Also

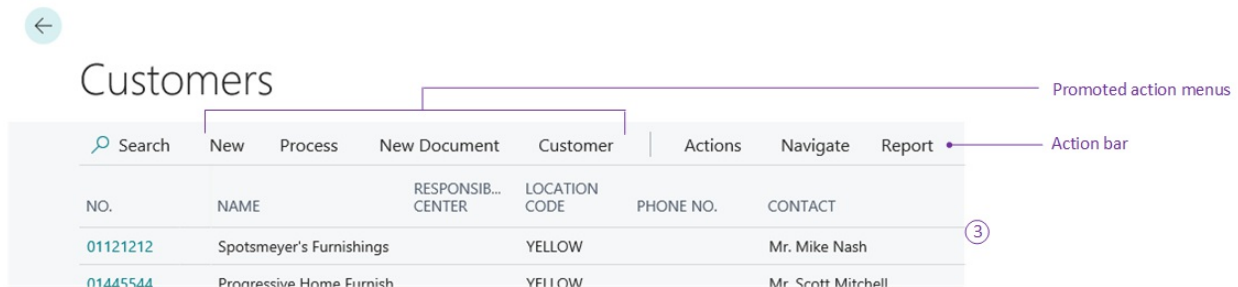
[AL Methods](#)

Actions Overview

5/21/2019 • 7 minutes to read

In Dynamics 365 Business Central, actions are displayed at the top of each page, referred to as the action bar. In this topic, you learn about different types of actions, and how you can enable users to quickly locate the actions they want to use.

The actions can be displayed in different menus on the action bar.



You can choose from the following action menus to place the actions in the specified area.

AREA	SYNTAX	USED ON	DESCRIPTION	EXAMPLE
Actions menu	<code>area(processing)</code>	Role Center, list, card, and task pages	User tasks	Post a sales order
New document group in Actions menu	<code>area(creation)</code>	List, card, Role Center pages, and task pages	Actions that appear under the New group. Opens a new Dynamics 365 document.	New sales invoice
Navigate menu	<code>area(navigation)</code>	List, card, and task pages	Links to other pages in Dynamics 365 Business Central.	Prices
Report menu	<code>area(reporting)</code>	Role Center, list, card, and task pages	A list of available reports.	Customer Top 10 List

The following Actions are related to the Role Center page.

AREA	SYNTAX	USED ON	DESCRIPTION	EXAMPLE
Navigation menus	<code>area(sections)</code>	Role Center pages	The top-level navigation consists of one or more root items that expand to display a submenu of links to other pages.	Posted sales invoices
Navigation bar	<code>area(embedding)</code>	Role Center pages	The second-level navigation displays a flat list of links to other pages.	Customers

For more information about actions used on the role center page, see [Designing Role Centers](#).

TIP

If you used to work in Microsoft Dynamics NAV, you can get an overview of the mapping between actions in the [Differences in the Development Environments](#) topic.

Types of Actions

Each page has a different set of actions depending on the page type, and the processes that the page supports. In order to create the appropriate set of actions for a particular page, you should have a good understanding of your customer's business processes.

Each process in an organization has several actions associated with it. You should try to create a full set of actions that mirror all tasks and processes that are performed.

Example: The Sales Orders list page at CRONUS International contains all actions related to processing sales orders. During user configuration and personalization, some of these actions may be hidden or promoted to the ribbon. Therefore, you must create a full set of actions for the customer.

Pages can have the following actions as described in each section below.

Actions menu

The Actions menu is displayed in the action bar on all page types, and contains relevant tasks for the current page. Typically, you add processing tasks and creation tasks in the Actions menu. To add processing actions such as posting a sale order, you must use the `processing` action area. They are regular daily tasks. Therefore, they must be on the Actions menu. For examples on how to add actions to the Actions menu, see [Adding Actions to a Page](#).

Some examples from the Customer page are as follows:

- Sales Invoice
- Sales Quote
- Sales Credit Memo
- Ledger Entries
- Invoice Discounts
- Prices
- Line Discounts

You can add actions to the Actions menu, group actions together under action sub menus, or promote them to the ribbon. For examples of how to use actions, see [Page Object](#) and [Page Extension Object](#).

New Document menu

The New Document menu is often displayed both as a top-level menu in the actions bar and as a sub menu in the Actions menu. You can use this menu to open new documents within Dynamics 365. You can add an action to create a new document such as creating a new sales invoice. This action displays in a separate menu called **New document** in the Actions menu. To add to the New document menu, you must use the `creation` action area.

Example: On the Customers page, if the order processor wants to create a new invoice, she can open the new page directly from the Actions menu. This is useful as she creates new sales invoices daily.

Navigate menu

The Navigate menu is displayed after the Actions menu in the action bar. Rather than providing tasks for the user, this menu provides additional information by taking the user to a specific page in Dynamics 365. To add a page link in the Navigate menu, you must use the `navigation` action area. These actions act like a bookmark to enable quick access to view a page.

NOTE

You should not add a Navigation action to a Role Center page.

Report menu

The Report menu is displayed after the Navigate menu in the action bar. The Reports menu lists the reports most relevant to a page. If a user does not require a Report menu, then the menu is hidden. Sometimes it is relevant to promote the most important reports to the top-level in the action bar to save the user from too many clicks. To create an action in the Report menu, you must use the `reporting` action area.

Promoted Actions

Promoted actions are actions that are set up on the Actions, Navigate, or Reports menus in the action bar, but are also configured to display in custom menus in the action bar. Although the actions are set up on the Actions, Navigate, or Report menus, you can choose to hide them on these menus and only show them in custom menus. For more information on how to add promoted actions, promoted categories and example, see [Promoted Actions](#).

Actions at runtime

An action can trigger code to run, such as posting a document or otherwise modifying a record in a table. When a user chooses an action, one of the following pieces of logic will happen in addition to the code that the action itself triggers:

- If the page is empty and no longer shows any records, the page is re-initialized with default values.
- If the page does show records, and the current state is within the page filters boundary, the **OnAfterGetRecord** trigger is executed on the page.
- If the current record that the page showed is now outside the filter but there are other records within the filter, the **OnFindRecord** trigger is called and the **OnAfterGetRecord** trigger is run on the next record with the given filters.

The logic runs in the transaction that the action triggered. This can cause the application code to result in users locking the whole table when they thought they were only modifying one record.

To avoid users accidentally locking tables, you can use the [SetSelectionFilter](#) method before your code passes the record variable to the processing codeunit, for example. The following code example illustrates the code on the [OnAction](#) trigger on an action on a page.

```
if confirm('Are you sure you want to call this codeunit?', true) then begin
    CurrPage.SetSelectionFilter(Rec);
    codeunit.Run(50000, Rec);
end;
```

See Also

AL Development Environment

Developing Extensions in AL

Pages Overview

Adding Actions to a Page

3/31/2019 • 4 minutes to read

This topic shows how to create new actions, how to add actions to a page, and how to preview them in the Dynamics 365 web client. In Dynamics 365, actions can be displayed in the action bar of all pages and grouped together under the following actions menus:

- Promoted action categories
- Actions
- Navigate
- Report

Before putting an action on a page you should think about the business processes that the action supports. For example, on page 42, the Sales Orders list page, the Actions button contains actions for all tasks related to processing sales orders. Creating these actions can make it easier for the order processor to perform their daily tasks, such as posting sales orders and creating new customer orders.

For more information about different types of actions and where to use them, see [Actions Overview](#).

TIP

After you have added actions to a page, you can use Designer to alter the actions, like moving an action to or from a promoted category, hiding and action or action group, and more. For more information, see [Using Designer](#).

To add Actions to a Page

The page actions are displayed on the header section. There are multiple tabs to help navigate to the right item.

In order to add actions to the action bar, you must use the keywords with Anchors or Targets. These keywords are used to place and move the actions around in the tab groups. For more information about adding, moving, and modifying actions, see [Using Keywords to Place Actions and Controls](#).

NOTE

Actions can only be linked to a page, or to a group control. Actions cannot be linked to fields, or parts on a page.

Set an icon to an action

Dynamics 365 Business Central includes images that you can use on actions in command bar menus and promoted actions on the ribbon. To add an image to an action, you add the **Image** property and you must provide the name of the image you that want to use from the Dynamics 365 Business Central Action icon library. By default, the size of images is 16 pixels high by 16 pixels wide. For promoted actions, you can choose to display larger images that are 32 pixels high and 32 pixels wide. For more information, see [Image Property](#).

Example

The following example shows how to use different action areas on a **page object of the PageType Card**. These actions will display in the following menus in the action bar.

1. Actions menu: The `area(Processing)` action area is used to display the action in the Actions menu. This action uses the **Promoted** and **PromotedCategory** properties in order to display the action in the promoted actions

menu called **Process**.

2. New Document group: The `area(Creation)` action area is used to display the action in the **New document** group in the Actions menu. Also, this action uses the **Image** property to display a form icon instead of a default icon.
3. Navigate menu: The `area(Navigation)` action area is used to display the action in the Navigate tab. This action and other actions in this example uses the **RunObject** property to assign a page to the action.
4. Report menu: The `area(Reporting)` action area is used to display this action in the Report menu, and also a Group control is added as a submenu to this menu. It sets the **Caption** property to make the action group visible in the Reports menu.

```

page 50110 PageName
{
    PageType = Card;

    actions
    {
        // Adds the action called "My Actions" to the Action menu
        area(Processing)
        {
            action("My Actions")
            {
                Promoted = true;
                PromotedCategory = Process;
                ApplicationArea = All;
                trigger OnAction()
                begin
                    Message('Hello World');
                end;
            }
        }

        area(Creation)
        {
            // Adds the action "My New document" to the New Document group in the Actions menu.
            action("My New document")
            {
                ApplicationArea = All;
                RunObject = page "Customer Card";
                Image = "1099Form";
            }
        }

        area(Navigation)
        {
            // Adds the action called "My Navigate" to the Navigate menu.
            action("My Navigate")
            {
                ApplicationArea = All;
                RunObject = page "Customer Card";
            }
        }

        area(Reporting)
        {
            // Adds a submenu called "My Label" to the Report menu.
            group(NewSubGroup)
            {
                Caption = 'My label';
                group(MyGroup)
                {
                    // Adds the action "My Report" to the My Label submenu.
                    action("My Report")
                    {
                        ApplicationArea = All;
                        RunObject = page "Customer Card";
                    }
                }
            }
        }
    }
}

```

NOTE

Actions can be assigned to a page by setting the RunObject property, or by adding a trigger to a Codeunit. For more information, see [RunObject Property](#) and [Codeunit Triggers](#).

The promoted action menus are always displayed first so the promoted actions provide quick access to common tasks, and users do not have to browse through a menu to access them. Add the Promoted property to add actions to the a promoted action menu. For more information on how to add promoted actions, promoted categories, and examples, see [Promoted Actions](#).

You can assign different icons for your actions from the Dynamics 365 image library. For more information, see [Image Property](#).

See Also

[Actions Overview](#)

[Pages Overview](#)

[Promoted Actions](#)

Promoted Actions

3/31/2019 • 4 minutes to read

Promoted actions are actions that are set up on the Actions, Navigate, or Report menus in the action bar, but are also configured to display on the Home tab. Although the actions are set up on the Actions, Navigate, or Report tabs, you can choose to hide them on these menus and only show them on the Home tab.

The following table describes where you can use promoted actions.

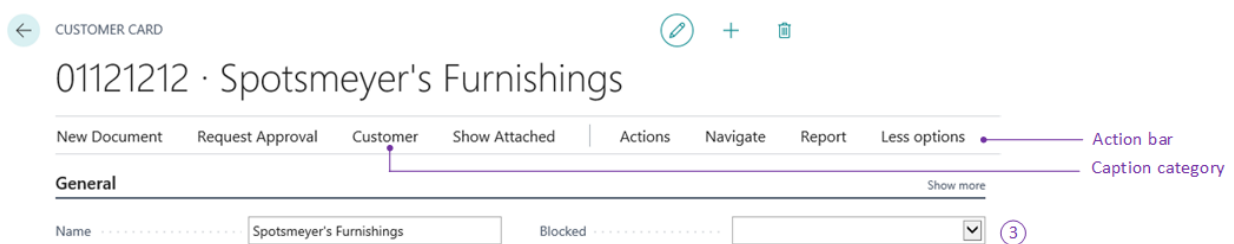
ACTION TYPE	USED ON	DESCRIPTION	EXAMPLE
Promoted Actions	List, card, Role Center pages, and task pages	Provide quick access to common tasks that appear under the Home tab.	Post and print a sales order

You can promote any command from the existing actions menus to the ribbon. If there are no promoted actions, the ribbon remains hidden. To promote an action on the Home tab, you set the **Promoted** property of the action. If you want to display the action only on the Home tab, then you add an additional step to set the **PromotedOnly** property. For more information, see [Promoted Property](#) and [PromotedOnly Property](#).

Promote actions by category

Promoted actions can be grouped. You can add promoted actions by different grouped categories. Typically, promoted actions are displayed in the ribbon of the role center client. You can organize promoted actions into different categories, where each category is indicated by a caption in the ribbon. You define up to 10 categories for a page. The following figure illustrates a page that has promoted actions under the following categories.

- New Document
- Request Approval
- Customer



You assign a promoted action to a category by setting the **PromotedCategory** property of the action. By default, these category names correspond to the captions that are displayed for the category on the page in Dynamics 365 Business Central. You will typically want to change the captions, especially the Category4 through Category10 captions. See the table below for the default **PromotedCategory** values. To change the default captions, set the **PromotedActionCategories** property. You type the values of the **PromotedActionCategories** where each caption is separated with a comma as shown below:

```
PromotedActionCategories =  
'New_caption,Process_caption,report_caption,category4_caption,category5_caption,category6_caption,category7_c  
aption,category8_caption,category9_caption,category10_caption';
```

The position of the caption in the list determines its corresponding category setting in the **PromotedCategory**

property for the actions as described in the table below.

PROMOTEDACTIONCATEGORIES CAPTION POSITION	DEFAULT PROMOTEDCATEGORY VALUES	EXAMPLE
First	New	<i>New_caption</i>
Second	Process	<i>Process_caption</i>
Third	Report	<i>Report_caption</i>
Fourth	Category4	<i>Category4_caption</i>
Fifth	Category5	<i>Category5_caption</i>
Sixth	Category6	<i>Category6_caption</i>
Seventh	Category7	<i>Category7_caption</i>
Eighth	Category8	<i>Category8_caption</i>
Ninth	Category9	<i>Category9_caption</i>
Tenth	Category10	<i>Category10_caption</i>

You can change category captions on a page-by-page basis and for each Dynamics 365 Business Central Windows client language.

For more information about these properties, see [PromotedCategory Property](#) and [PromotedActionCategories Property](#).

Assigning an icon to the promoted actions

Each promoted action has an icon associated with it. You can accept a default icon, or choose your own from the Dynamics 365 Business Central image library by using the **Image** property, where each promoted action has an icon associated with it. Also, you can use a larger icon that makes it more prominent to the user by using the **PromotedIsBig** property. For more information, see [Image Property](#) and [PromotedIsBig Property](#).

Example

The example shows how to promote actions on a Customers page using different properties:

1. The actions in the example are promoted to display in the **New Document**, **Request Approval** and **Customer** groups on the Home tab.
2. The **Sales Quote** and **Sales Invoice** actions are promoted to the ribbon and grouped in a category called **New Document**.
3. The `PromotedCategory` value; `Category5` corresponds the caption position in the `PromotedActionCategories` value with the **New Document** caption.
4. Each promoted action in the example is assigned to a unique icon. Additionally, to display bigger icons, the **Sales Quote** and **Contact** actions are set with the `PromotedIsBig` property.
5. The **Sales Quote** and **Send Approval Request** actions are set to appear only on the Home tab.

```
page 50103 Customers
{
    PageType = Card;
    SourceTable = Customer;
```

```

Sourceable = Customer;
PromotedActionCategories = 'New,Process,Report,Manage,New Document,Request Approval,Customer,Page';

actions
{
    area(Creation)
    {
        action("Sales Quote")
        {
            Promoted = true;
            PromotedCategory = Category5; // PromotedActionCategories = New Document
            PromotedOnly = true;
            PromotedIsBig = true;
            Image = NewSalesQuote;
            ApplicationArea = All;
            trigger OnAction()
            begin
                Message('Create sales quote');
            end;
        }
        action("Sales Invoice")
        {
            Promoted = true;
            PromotedCategory = Category5; // PromotedActionCategories = New Document
            Image = SalesInvoice;
            ApplicationArea = All;
            trigger OnAction()
            begin
            end;
        }
    }
    area(Processing)
    {
        action("Send Approval Request")
        {
            Promoted = true;
            PromotedOnly = true;
            PromotedCategory = Category6; // PromotedActionCategories = Request Approval
            Image = SendApprovalRequest;
            ApplicationArea = All;
            trigger OnAction()
            begin
            end;
        }
        action("Cancel Approval Request")
        {
            Promoted = true;
            PromotedCategory = Category6; // PromotedActionCategories = Request Approval
            Image = CancelApprovalRequest;
            ApplicationArea = All;
            trigger OnAction()
            begin
            end;
        }
    }
    area(Navigation)
    {
        action(Contact)
        {
            Promoted = true;
            PromotedCategory = Category7; // PromotedActionCategories = Customer
            PromotedIsBig = true;
            Image = CustomerContact;
            ApplicationArea = All;
            trigger OnAction()
            begin
            end;
        }
        action("Account Details")
        {

```

```
    {  
        Promoted = true;  
        PromotedCategory = Category7; // PromotedActionCategories = Customer  
        Image = Account;  
        ApplicationArea = All;  
        trigger OnAction()  
        begin  
            end;  
        }  
    }  
}
```

For more examples of how to use actions, see [Page Object](#) and [Page Extension Object](#).

See Also

[Actions Overview](#)

[Adding Actions to a Page](#)

[AL Development Environment](#)

[Developing Extensions in AL](#)

[Pages Overview](#)

Inspecting and Troubleshooting Pages

5/3/2019 • 5 minutes to read

The Business Central Web client includes a page inspection feature that lets you get details about a page, providing insight into the page design, the different elements that comprise the page, and the source behind the data it displays. Page inspection helps you:

- Learn the data model behind a page.
- Discover pages and parts that can be reused in your application design.
- Troubleshoot data issues without having to perform tasks like copying the production database, viewing the entire source table, or digging into SQL.
- Debug the application, complementing [Designer](#).

Working with Page Inspection

You start page inspection from the **Help & Support** page. Choose the question mark in the top right corner, choose **Help & Support**, and then choose **Inspect pages and data**. Or, you can just use the keyboard shortcut **Ctrl+Alt+F1**.

The **Page inspection** pane opens on the side. The following figure illustrates the **Page Inspection** pane on the **Sales Order** page.

The screenshot shows the Dynamics 365 Business Central interface. The main window displays the 'SALES ORDER' page for 'S-ORD101004 · Alpine Ski House'. The 'General' tab is active, showing fields for Customer Name, Contact, Posting Date, and Order Date. Below this is a table with columns: TYPE, NO., DESCRIPTION, LOCATION CODE, QUANTITY, and RESE. The table contains two rows of data: 'SYDNEY Swivel Chair, green' and 'PARIS Guest Chair, black'. At the bottom of the table, there are summary rows for 'Subtotal Excl. Tax (US...)', 'Total Excl. Tax (USD)', 'Inv. Discount Amount...', 'Total Tax (USD)', 'Invoice Discount %', and 'Total Incl. Tax (USD)'. On the right side, the 'Page Inspection' pane is open, showing a list of fields and their data types. The fields listed are: Document Type (1, Option, PK), Order, Sell-to Customer No. (2, Code), 40000, Document No. (3, Code, PK), S-ORD101004, Line No. (4, Integer, PK), 10000, Type (5, Option), Item, No. (6, Code), 2000-S, Location Code (7, Code), (Blank), Posting Group (8, Code), RESALE, and Shipment Date (10, Date), 05/13/19.

TYPE	NO.	DESCRIPTION	LOCATION CODE	QUANTITY	RESE
Item	2000-S	SYDNEY Swivel Chair, green		3	
Comment					
Comment					
Item	1900-S	PARIS Guest Chair, black			

Subtotal Excl. Tax (US...)	570.30	Total Excl. Tax (USD)	570.30
Inv. Discount Amount...	0.00	Total Tax (USD)	39.92
Invoice Discount %	0	Total Incl. Tax (USD)	610.22

When the **Page Inspection** pane first opens, it shows information that pertains to the main page object.

Use the keyboard or pointing device to move focus to different elements on the page. When you select a FactBox or a part on the main page, the bounding area is highlighted by a border, and the **Page Inspection** pane shows information about the selected element. For example, the previous figure shows information about the list part in

the **Sales Order** page.

As you navigate to other pages in the application, the **Page Inspection** pane will automatically update with page information as you move along.

What Page Inspection Shows

In short, the page inspection pane shows the information for the main page or sub-page in a part, the page's source table (if any) and fields, extensions that affect the page, and current filters applied to the page. The following sections describe details about what is shown.

NOTE

If you do not see all details described below, you might not have the required permissions. For more information, see [Controlling Access to Page Inspection Details](#).

- [Page](#)
- [Table](#)
- [Table Fields](#)
- [Extensions](#)
- [Page Filters](#)

The **Page** field shows information about the main page or a selected (highlighted) sub-page in a part. The field shows the following information:

- The name, as specified by its [Name property](#)
- The ID as specified by the [ID property](#).
- The type, as specified by the [PageType property](#).

Elements shown with limited information

- Role Center pages

If a page has the type Role Center, the **Table** field does not appear. Because the Role Center consists of several parts, there is no more information shown. To see more details, select the different parts that make up the Role Center.

- Report request pages and previews

If you open a report request page or preview for inspection, the only information that is shown in the Page Inspection pane is the report's name and ID.

- System parts, such as Links or Notes, and parts containing charts.

Control Add-in Style Guide

3/31/2019 • 5 minutes to read

This article offers a variety of stylistic definitions that are used throughout Dynamics 365, which you can apply to your control add-ins to create an experience that complements Dynamics 365.

Introduction

Control add-ins for Dynamics 365 extend a business solution by surfacing contextual functionality alongside business data. Control add-ins empower users to get more done without costly context switching, no matter which device they access Dynamics 365 from. Typical uses of control add-ins include unique data visualizations, surfacing controls from a third party service, or displaying related content from another data source.

Apart from the functionality, an important aspect of creating a control add-in is making sure the control add-in looks good and blends seamlessly into Dynamics 365. To achieve this, you should follow these basic principles:

- Apply similar patterns for command, navigation and presentation of data.
- Favor content over chrome
- Design for all platforms and input methods.
- Make it accessible to all users.
- Make it enjoyable and keep users in control.

Dynamics 365 uses a set of specific colors and fonts. You can employ these colors and fonts in your control add-ins to give it a style that matches the rest of client's user interface.

IMPORTANT



This article is currently in progress and contents will change.

Colors

Choosing the right color gives the interface visual continuity. Color can be used to convey information to users, indicate interactivity, give feedback, and more. The following sections describe the colors used in Dynamics 365. The colors can be used on all aspects of a UI element, such background, border, text, and more.

Main colors



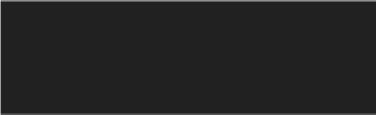





The following colors represent the Dynamics 365 theme main palette.

COLOR	NAME	USE	HEX VALUE
	Primary color	Prominent UI elements and areas.	#00B7C3
	Secondary color	UI elements and areas in default or subdued state.	#505C6D

Style colors



The following colors are used to express or accent conditions or user activity in the UI. For example, these colors

are used as sentiments, or color indication, on Cues.

COLOR	DESCRIPTION	HEX VALUE
	Standard	#212121
	Accent	#00B7C3
	Strong	#212121
	Favorable	#35AB22
	Ambiguous	#9F9700
	Unfavorable	#EB6965
	Attention	#EB6965
	Subordinate	#A7ADB6

More palette colors

The following table includes additional colors that you can use in the UI.

COLOR	DESCRIPTION	HEX VALUE
	Yellow	#C9C472
	Green	#88CE81

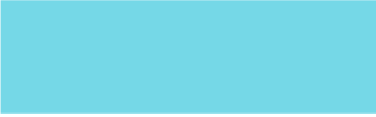


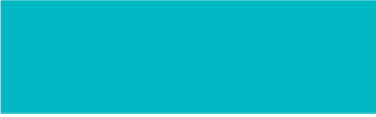



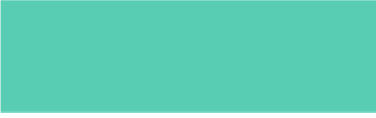
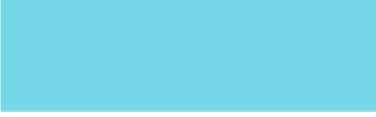
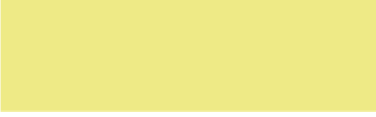

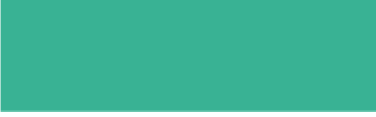

COLOR	DESCRIPTION	HEX VALUE
	Red	#E97768
	Blue	#75B5E7
	Light green	#59CCB4
	Sky	75D8E7
	Egg	EEEE86
	Orange	#E89E63
	Violet	#DBBDEB
	Teal	#39B294
	Grass	#73BA5A
	Scarlet	#E65E6D

Chart colors

The following table describes the colors used in charts.

COLOR	DESCRIPTION	HEX VALUE
	-	#505C6D
	-	#008089
	Primary color	#00B7C3
	Yellow	#C9C472
	Red	#E97768
	Blue	#75B5E7
	Light green	#59CCB4
	Sky	75D8E7
	Egg	EEEEA86
	Violet	#DBBDEB
	Teal	#39B294
	Grass	#73BA5A

Applying colors

To apply a color scheme to the control add-in, you specify CSS rule-sets that use the following properties:

PROPERTY	DESCRIPTION
<code>color</code>	Specifies font color.
<code>background-color</code>	Specifies background colors.
<code>border-color</code>	Specifies Border colors.

For example, to change the background of a part of your UI to use the `Secondary (#505C6D)` color, write the following CSS:

```
.my-ui-part {  
    background-color: #505C6D;  
}
```

If you want to change the text color of a caption to the Primary (#00B7C3) color, use the following CSS:

```
.my-caption {  
    color: #00B7C3;  
}
```

Typography

The main goal of typography is to provide clean and readable text in the user interface. Similar to colors, typography can also be used to convey or communicate conditions to the user.

Font Families

Dynamics 365 uses the following font families to specify the typeface and weight for text elements, such as headings, captions, messages, and so on:

EXAMPLE	NAME	VALUE
This is the Segoe UI font	Segoe UI	"Segoe UI", "Segoe WP", Segoe, device-segoe, Tahoma, Helvetica, Arial, sans-serif
This is the Segoe UI Light font	Segoe UI Light	"Segoe UI Light", "Segoe WP Light", device-segoe-light, "Segoe WP Semilight", "Segoe UI", "Segoe WP", Segoe, Tahoma, Helvetica, Arial, sans-serif
This is the Segoe UI Semi Light font	Segoe UI Semilight	"Segoe UI Semilight", "Segoe WP Semilight", device-segoe-semilight, "Segoe UI", "Segoe WP", Segoe, Tahoma, Helvetica, Arial, sans-serif
This is the Segoe UI Semibold font	Segoe UI Semibold	"Segoe UI Semibold", "Segoe WP Semibold", device-segoe-semibold, "Segoe UI", "Segoe WP", Segoe, Tahoma, Helvetica, Arial, sans-serif

EXAMPLE	NAME	VALUE
This is the Bahnschrift font	Bahnschrift	webclient-standard, device-standard, "Segoe UI", "Segoe WP", Segoe, Tahoma, Helvetica, Arial, sans-serif

Sizes

Dynamics 365 uses the following font sizes for text. The same font family on different clients may apply different sizes.

EXAMPLE	NAME	VALUE
This is 37.5pt	largest-plus-font-size	37.5pt
This is 30pt	largest-font-size	30pt
This is 22.5pt	large-plus-font-size	22.5pt
This is 18pt	large-font-size	18pt
This is 15pt	medium-plus-font-size	15pt
This is 13.5pt	medium-font-size	13.5pt
This is 12pt	small-plus-font-size	12pt
This is 10.5pt	small-font-size	10.5pt
This is 9pt	smallest-font-size	9pt

Applying Font Families and Sizes

To apply fonts and sizes to text elements in the UI, you need specify the following CSS properties:

- Font family. use property `font-family` .
- Font size. use property `font-size` .

For example, to change a UI element for the Web client to use the font family *Segoe UI Light* and the size *Small*

(10.5pt), write the following CSS:

```
.my-ui-part {  
    font-family: "Segoe UI Light", "Segoe WP Light", device-segoe-light, "Segoe WP Semilight", "Segoe UI",  
    "Segoe WP", Segoe, Tahoma, Helvetica, Arial, sans-serif;  
    font-size: 10.5pt;  
}
```

IMPORTANT

To ensure that the correct fonts are used on devices, do not omit fonts or change the order of the fonts.

Example

This examples illustrates how to use CSS to style a simple HTML UI part of a control add-in. The example includes three UI controls, as shown in the following HTML code:

```
<div class="addin">  
    <div class="control">  
        <div class="caption">Name:</div>  
        <div class="value">  
            <input type="text" name="name">  
        </div>  
    </div>  
  
    <div class="control">  
        <div class="caption">Surname:</div>  
        <div class="value">  
            <input type="text" name="name">  
        </div>  
    </div>  
  
    <div class="control">  
        <div class="submit">Submit</div>  
    </div>  
</div>
```

The following is CSS code for styling the controls, including padding, background colors, and fonts:

```
.addin {
  padding: 1em;
  background-color: #505C6D; /* Sets the background color to "Secondary" */
}

.addin .control {
  border-color: #00B7C3; /* Sets the border color to "Primary" */
}

.addin .control .caption {
  color: #00B7C3; /* Sets the captions to "Primary" */

  /* Segoe UI Light, small */
  font-family: "Segoe UI Light", "Segoe WP Light", device-segoe-light, "Segoe WP Semilight", "Segoe UI",
"Segoe WP", Segoe, Tahoma, Helvetica, Arial, sans-serif; /* Sets the font of the caption to ""Segoe UI Light"
*/
  font-size: 10.5pt;
}

.addin .control .value {
  color: #008089; /* Tertiary shade 2 */

  /* Segoe UI, medium */
  font-family: "Segoe UI", "Segoe WP", Segoe, device-segoe, Tahoma, Helvetica, Arial, sans-serif;
  font-size: 12pt;
}

.addin .control .submit {
  color: white; /* Sets the caption text to "white" */
  background-color: #00B7C3; /* Sets the background to "Primary" */

  /* Segoe UI Semibold, medium */
  font-family: "Segoe UI Semibold", "Segoe WP Semibold", device-segoe-semibold, "Segoe UI", "Segoe WP",
Segoe, Tahoma, Helvetica, Arial, sans-serif;
  font-size: 12pt;
  text-transform: uppercase; /* Sets the caption to use all uppercase letters */
}
```

Reports Overview

3/31/2019 • 2 minutes to read

You can use reports to print or display information from a database. Use reports to structure and summarize information to print documents, such as invoices. For example, create a report that lists all customers and orders that have been added by each customer. Also, create a report that is automatically filled with the relevant information for an invoice.

Reports can also be used to process data without printing or displaying content. For example, use a report to automate updating all prices in an item list. It can be easier to create a report to process data instead of a codeunit to do the same processing because you can use:

- Request page functionality to select options and filters for data items, which are available in a report but are difficult to add to a codeunit. For more information, see [Request Pages](#).
- Report data items instead of writing code to open tables and retrieve records.
- Data modeling, which is available when you design reports.

Creating reports

Creating a report involves two primary tasks. First, you create a report object and design the dataset. The dataset determines the data that is extracted or calculated from the Dynamics 365 Business Central database tables that can be used in a report. After the dataset has been designed, you design the visual layout of the report. There are two types of report layouts that you can create: layouts using report definition language (RDL) and Word report layouts.

Getting started

The following table includes links to help you get started with designing the reports.

TO	SEE
Learn the overview of the report design process	Report Design Overview
Understand the report structure and designing the layout for a report.	Report Object
Understanding the data model and dataset of a report	Defining a Report Dataset
Learn how to create a report using a word layout	Creating a Word Layout Report
Learn how to create a report using an RDL layout report.	Creating an RDL Layout Report

See Also

[Report Object](#)

[Creating a Report](#)

[Creating an RDL Layout Report](#)

Report Design Overview

3/31/2019 • 2 minutes to read

A report is composed of the following items:

- A report object
- A report dataset
- A report layout
- A request page
- Properties, triggers, and code

You design a report by first defining the dataset, and then designing the visual layout.

Report object

You create a report object in the AL Language development environment to define the data model, or dataset of a report. You can structure and summarize information in a report and print documents, such as sales quotes and invoices. For more information, see [Report Object](#).

Report dataset

In order to define the underlying data model, you use the report dataset. A report dataset determines the data that is extracted or calculated from the Dynamics 365 Business Central database tables that can be used in a report. You build the report dataset by adding data items and columns. For more information, see [Report Dataset](#).

Report layouts

The visual layout determines the content and format of a report when it is viewed and printed. You build the layout of a report by arranging data items and columns and specifying the general format, such as text font and size. A report that is viewed, printed, or saved from a Dynamics 365 Business Central client must have a report layout. There are two types of report layouts: layouts using report definition language (RDL) and Word layouts.

RDL layout

To create an RDL layout report, you use Visual Studio Report Designer or Microsoft SQL Server Reporting Services Report Builder. For more information, see [Creating an RDL Layout Report](#).

IMPORTANT

RDL layouts can result in slower performance with document reports, regarding actions that are related to the user interface (for example, like sending emails) compared to Word layouts. When developing layouts for document reports, we recommend that you design Word layouts instead of RDL. With Word layouts, reports are not impacted by the security constraints on sandbox appdomains like they are with RDL layouts. From a service perspective, RDL layouts are not trusted, so they will run in a sandbox appdomain that only lives for the current report invocation.

Word report layout

You create Word layouts by using a Word Document. Word layouts are based on a Word document that includes a custom XML parts that represents the report dataset. For more information, see [Creating a Word Layout Report](#).

See Also

[Reports](#)

[Report Object](#)

[Report Data Type](#)

Report Object

3/31/2019 • 3 minutes to read

Reports are used to print or display information from a database. You can use a report to structure and summarize information, and to print documents, such as sales quotes and invoices.

Creating a report consists of two primary tasks; the first task is to create the underlying data model and the next is to define the visual layout that displays the data. The report object defines the underlying data model and specifies which database tables and fields to pull data from. When the report is run, that data is displayed in a specified layout; the visual layout, which determines the content and format of a report when it is viewed and printed.

For more information about defining database tables and fields, see [Defining a Report Dataset](#).

You build the layout of a report by arranging data items and columns, and specifying the general format, such as text font and size. There are two types of report layouts; client report definition, also called RDL layouts and Word layouts. RDL layouts are defined in Visual Studio Report Designer or Microsoft SQL Server Reporting Services Report Builder. Word layouts are created using Word. Word layouts are based on a Word document that includes a custom XML part representing the report dataset.

NOTE

Extension objects can have a name with a maximum length of 30 characters.

Snippet support

Typing the shortcut `treport` will create the basic layout for a report object when using the AL Language extension in Visual Studio Code.

Report syntax

```

report Id MyReport
{
    UsageCategory = Administration;
    ApplicationArea = All;

    dataset
    {
        dataitem(DataItemName; SourceTableName)
        {
            column(ColumnName; SourceFieldName)
            {
            }
        }
    }

    requestpage
    {
        ContextSensitiveHelpPage = 'my-feature';
        layout
        {
            area(Content)
            {
                group(GroupName)
                {
                    field(Name; SourceExpression)
                    {
                        ApplicationArea = All;
                    }
                }
            }
        }

        actions
        {
            area(processing)
            {
                action(ActionName)
                {
                    ApplicationArea = All;
                }
            }
        }
    }

    var
        myInt: Integer;
}

```

Report example

The following example is a report that prints the list of customers. The report object defines a dataset of columns from the Customer table. For more information on creating a Word Layout report, see [Creating a Report](#).

```

report 50103 "Customer List"
{
    CaptionML=ENU='Customer List';
    RDLCLayout = 'Customer List Report.rdlc'; // if Word use WordLayout property
    dataset
    {
        dataitem(Customer;Customer)
        {
            RequestFilterFields="No.". "Search Name". "Customer Posting Group":

```

```

Request: 1100110011001; Search Name ; Customer Posting Group ;
column(COMPANYNAME;COMPANYNAME)
{
}
column(CurrReport_PAGENO;Customer."no.")
{
}
column(Customer_TABLECAPTION_CustFilter;TABLECAPTION + ': ' + CustFilter)
{
}
column(CustFilter;CustFilter)
{
}
column(Customer_No;"No.")
{
}
column(Customer_Customer_Posting_Group;"Customer Posting Group")
{
}
column(Customer_Customer_Disc_Group;"Customer Disc. Group")
{
}
column(Customer_Invoice_Disc_Code;"Invoice Disc. Code")
{
}
column(Customer_Customer_Price_Group;"Customer Price Group")
{
}
column(Customer_Fin_Charge_Terms_Code;"Fin. Charge Terms Code")
{
}
column(Customer_Payment_Terms_Code;"Payment Terms Code")
{
}
column(Customer_Salesperson_Code;"Salesperson Code")
{
}
column(Customer_Currency_Code;"Currency Code")
{
}
column(Customer_Credit_Limit_LCY;"Credit Limit (LCY)")
{
    DecimalPlaces=0;0;
}
column(Customer_Balance_LCY;"Balance (LCY)")
{
}
column(CustAddr_1;CustAddr[1])
{
}
column(CustAddr_2;CustAddr[2])
{
}
column(CustAddr_3;CustAddr[3])
{
}
column(CustAddr_4;CustAddr[4])
{
}
column(CustAddr_5;CustAddr[5])
{
}
column(Customer_Contact;Contact)
{
}
column(Customer_Phone_No;"Phone No.")
{
}
column(CustAddr_6;CustAddr[6])
{
}

```

```

    {
    }
    column(CustAddr_7;CustAddr[7])
    {
    }
    column(Customer_ListCaption;Customer_ListCaptionLbl)
    {
    }
    column(CurrReport_PAGENOCaption;CurrReport_PAGENOCaptionLbl)
    {
    }
    column(Customer_NoCaption;FIELDCAPTION("No."))
    {
    }
    column(Customer_Customer_Posting_GroupCaption;Customer_Customer_Posting_GroupCaptionLbl)
    {
    }
    column(Customer_Customer_Disc_GroupCaption;Customer_Customer_Disc_GroupCaptionLbl)
    {
    }
    column(Customer_Invoice_Disc_CodeCaption;Customer_Invoice_Disc_CodeCaptionLbl)
    {
    }
    column(Customer_Customer_Price_GroupCaption;Customer_Customer_Price_GroupCaptionLbl)
    {
    }
    column(Customer_Fin_Charge_Terms_CodeCaption;FIELDCAPTION("Fin. Charge Terms Code"))
    {
    }
    column(Customer_Payment_Terms_CodeCaption;Customer_Payment_Terms_CodeCaptionLbl)
    {
    }
    column(Customer_Salesperson_CodeCaption;FIELDCAPTION("Salesperson Code"))
    {
    }
    column(Customer_Currency_CodeCaption;Customer_Currency_CodeCaptionLbl)
    {
    }
    column(Customer_Credit_Limit_LCYCaption;FIELDCAPTION("Credit Limit (LCY)"))
    {
    }
    column(Customer_Balance_LCYCaption;FIELDCAPTION("Balance (LCY)"))
    {
    }
    column(Customer_ContactCaption;FIELDCAPTION(Contact))
    {
    }
    column(Customer_Phone_NoCaption;FIELDCAPTION("Phone No."))
    {
    }
    column(Total_LCY_Caption;Total_LCY_CaptionLbl)
    {
    }

    trigger OnAfterGetRecord();
    begin
        CALCFIELDS("Balance (LCY)");
        FormatAddr.FormatAddr(
            CustAddr,Name,"Name 2",'',Address,"Address 2",
            City,"Post Code",County,"Country/Region Code");
    end;

}
}

requestpage
{
    SaveValues=true;
    ContextSensitiveHelpPage = 'my-feature';
}

```

```

    layout
    {
    }

    actions
    {
    }
}

labels
{
    LabelName = 'LabelText', Comment = 'Foo', MaxLength = 999, Locked = true;
}

trigger OnPreReport();
var
    CaptionManagement : Codeunit 42;
begin
    CustFilter := CaptionManagement.GetRecordFiltersWithCaptions(Customer);
end;

var
    FormatAddr : Codeunit 365;
    CustFilter : Text;
    CustAddr : ARRAY [8] OF Text[50];
    Customer_ListCaptionLbl : TextConst ENU='Customer - List';
    CurrReport_PAGENOCaptionLbl : TextConst ENU='Page';
    Customer_Customer_Posting_GroupCaptionLbl : TextConst ENU='Customer Posting Group';
    Customer_Customer_Disc_GroupCaptionLbl : TextConst ENU='Cust./Item Disc. Gr.';
    Customer_Invoice_Disc_CodeCaptionLbl : TextConst ENU='Invoice Disc. Code';
    Customer_Customer_Price_GroupCaptionLbl : TextConst ENU='Price Group Code';
    Customer_Payment_Terms_CodeCaptionLbl : TextConst ENU='Payment Terms Code';
    Customer_Currency_CodeCaptionLbl : TextConst ENU='Currency Code';
    Total_LCY_CaptionLbl : TextConst ENU='Total (LCY)';
}

```

See Also

[Creating an RDL Layout Report](#)

[Creating a Word Layout Report](#)

[Adding Help Links from Pages, Reports, and XMLports](#)

[Page Extension Object](#)

[Page Properties](#)

[Developing Extensions](#)

[AL Development Environment](#)

Defining a Report Dataset

3/31/2019 • 2 minutes to read

You use a report object in the AL Language development environment to define the data model, or dataset, of a report. The dataset determines the data that is extracted or calculated from the Dynamics 365 Business Central database tables that can be used in a report. For more information, see [Report Object](#).

You build the report dataset from data items and columns. A data item is a table. A column can be:

- A field in a table
- A variable
- An expression
- A text constant

Typically, data items and columns correspond to fields in a table. When the report is run, each data item is iterated for all records in the underlying table. Filters are applied and the dataset is created. When a report is based on more than one table, you must set relations between the data items so that you can retrieve and organize the data.

Snippet support

Typing the shortcut `treport` will create the basic layout for a report object when using the AL Language extension in Visual Studio Code.

Example

The following example adds the `Customer` table as the data item and the `CustomerName` and `CompanyName` as fields of a column to the report. For more information on creating a report, see [Creating a Report](#).

```
dataset
{
    dataitem(Customer; Customer)
    {
        column(CustomerName; CustomerName)
        {
        }
        column(CompanyName; CompanyName)
        {
        }
    }
}
```

See Also

[Report Object](#)

[Reports Overview](#)

[Report Design Overview](#)

Request Pages

3/31/2019 • 2 minutes to read

A request page is a page that is run before the report starts to execute. Request pages enable end users to specify options and filters for a report. Request pages are defined as part of designing a [report object](#). The syntax is shown further down in this topic. You design the filters on request pages by using the following report properties:

PROPERTY	DESCRIPTION
RequestFilterHeading Property	Sets a caption for the request page tab that is related to this data item.
RequestFilterHeadingML Property	Sets the text used as a RequestFilterHeading Property for a request page tab.
RequestFilterFields Property	Sets which fields are automatically included on the tab of the request page that is related to this data item. The user can set filters on these fields.

In addition to specifying options and filters, users can choose from the following actions on a request page:

- Print
- Preview
- Cancel
- Help

Filtering on request pages

The fields that you define as `RequestFilterFields` are shown on the request page. In addition, an end user can add more fields on which to filter to the request page. Defining the `RequestFilterFields` property in `dataitem()` part of the report code is done as illustrated in the example below:

```
report 50103 "Customer List"
{
    CaptionML = ENU = 'Customer List';
    RDLCLayout = 'Customer List Report.rdlc'; // if Word use WordLayout property
    dataset
    {
        dataitem(Customer; Customer)
        {
            RequestFilterFields = "No.", "Search Name", "Customer Posting Group";
        }
    }
    ...
}
```

For more information about the report object, see [Report Object](#).

Set the [SaveValues](#) property to `true` in order to save the values that the end user enters on the request page.

NOTE

We recommend that you add fields that the end users of the report will frequently set filters on.

By default, for every data item in the report, a FastTab for defining filters and sorting is created on the request

page. To remove a FastTab from a request page, do not define any `RequestFilterFields` for the data item and set the `DataItemTableView` property to define sorting. The request page is displayed, but there is no tab for this data item.

If a `DataItemTableView` is not defined, then end users can select a sort field and sort order at runtime.

If you set the property `UseRequestPage` to `No`, then the report will start to print as soon as it is run. In this case, end users cannot cancel the report run. It is still possible to cancel the report, but some pages may print.

In a complex report that uses data from several tables, the functionality may depend on a specific key and sort order. Design your reports so that end users cannot change the sort order in a way that affects the functionality of the report.

See Also

[Report Object](#)

[Reports Overview](#)

[Report Design Overview](#)

[RequestFilterHeading Property](#)

[RequestFilterHeadingML Property](#)

[RequestFilterFields Property](#)

[DataItemTableView](#)

Adding Pages and Reports to Tell me

3/31/2019 • 3 minutes to read

The Business Central client includes the **Tell me** feature that lets users find objects and online help articles by entering search terms. When you have added a page or a report in your extension, you most likely want it to be discoverable to users in **Tell me**. In AL, you make a page or report searchable from **Tell me** by setting the [UsageCategory property](#) in code. The **UsageCategory** setting will make the page or report searchable, and the value chosen for the setting will further sub categorize the item.

TELL ME WHAT YOU WANT TO DO

cust

On current page (Business Manager)

Customer

Register a new customer.

Register Customer Payments

Process your customer payments by matching amounts received on your bank account wi...

Go to Pages and Tasks

Show all (46)

> Customers

Lists

> Customer Disc. Groups

Administration

> Customer Order Status

Tasks

Go to Reports and Analysis

Show all (50)

Customer Labels

Reports and Analysis

Customer Listing

Reports and Analysis

Customer Register

Reports and Analysis

Documentation

Show all (20)

?

Transferring and Posting Cost Entries

Before you define cost allocations, you must understand where cost entries come from.

?

Manually Adjust the Costs of Items

You can adjust the inventory valuation of an item using the FIFO or Average costing meth...

?

Managing Inventory Costs

Cost management, also referred to as "costing", is concerned with recording and reportin...

Get from Microsoft AppSource

Show all (48)

dynamic commerce Shipping Costs

Manage shipping cost directly in Dynamics 365 Business Central.

Advanced Inventory Count

Simplify inventory counts with comprehensive data entry, reconciliation, posting and anal...

Cash Basis Accounting

Cash Basis Accounting

Tell me finds pages and reports by searching the captions that are specified on page and report objects by the [CaptionML property](#).

Working with the UsageCategory property

When you create a [Page](#) or a [Report](#), you add the [UsageCategory Property](#). If the **UsageCategory** is set to **None**, or if you do not specify **UsageCategory**, the page or report will not show up when you search in Dynamics 365 Business Central.

UsageCategory property values

The values for the **UsageCategory** property are listed below. The sub category will help the user navigate through the search results and it is a best practice to be consistent when categorizing the pages and the reports that you add. A consistent approach will help guiding the user and improve productivity.

VALUE	DESCRIPTION
None	The page or report is not included in search.
Lists	The page or report is listed as Lists under the Pages and Tasks category.
Tasks	The page or report is listed as Tasks under the Pages and Tasks category.
ReportsAndAnalysis	The page or report is listed as Reports and Analysis under the Reports and Analysis category.
Documents	The page or report is listed as Documents under the Reports and Analysis category.
History	The page or report is listed as Archive under the Reports and Analysis category.
Administration	The page or report is listed as Administration under the Pages and Tasks category.

Adding additional search terms

You can specify other words or phrases that can help users find a page or report by using the [AdditionalSearchTerms](#) and [AdditionalSearchTermsML](#) properties. If the page or report is searchable by **Tell me** (that is, the **UsageCategory** property is set a value other than `None`), the search terms specified by these properties are used in addition to the caption of the page or report. These properties are useful when the caption does not always reflect what users will look for. A good example of this in Business Central is pages and reports associated with **Item**. Users unfamiliar with Business Central might look for 'product' or 'merchandise' instead of 'item'.

NOTE

For Business Central on-premises, the Business Central Web Server configuration file (navsettings.json) includes a setting called `UseAdditionalSearchTerms` that enables or disables the use of additional search terms by the **Tell me**. For more information, see [Configuring Business Central Web Server Instances](#).

Example

The following example creates a `SimpleItemList` page and sets a `UsageCategory` property to the page, so that the `SimpleItemList` page is discoverable through search using the **Tell me** feature. Also, the example sets the `AdditionalSearchTerms` property to add two search terms for the page.

```
page 50210 SimpleItemList
{
    PageType = List;
    SourceTable = Item;
    UsageCategory = Lists;
    AccessByPermission = page SimpleItemList = X;
    ApplicationArea = All;
    AdditionalSearchTerms = 'product, merchandise';

    layout
    {
        area(content)
        {
            group(General)
            {
                field("No.;" "No.") {}
                field(Name;Name) {}
                field>Description;Description) {}
            }
        }
    }
}
```

Optional settings

In addition to making a page or report searchable, you can control the access of an object by providing **Read**, **Insert**, **Modify**, **Delete**, and **Execute** (RIMDX) permissions by adding the [AccessByPermission property](#). Likewise, control the application area access on the specified object by adding the [ApplicationArea Property](#).

The **AccessByPermission** property and **ApplicationArea** property are the optional settings, which can be applied with the **UsageCategory** property. These settings are used to set restrictions on an object when you enable the Search functionality.

Working in the Dynamics NAV Development Environment

If you are using the Dynamics NAV Development Environment, you can also set `UsageCategory`, `AdditionalSearchTerms`, `AccessByPermission`, and `ApplicationArea` properties on pages and reports to control their search.

After you change these properties by using the Dynamics NAV Development Environment, before the changes take effect in the client, you must run **Build Object Search Index** from the **Tools** menu.

See Also

[Adding Menus to the Navigation Pane](#)

[UsageCategory Property](#)

[Page Object](#)

[Report Object](#)

[AL Development Environment](#)

Testing Reports

3/31/2019 • 2 minutes to read

Testing your report requires you to run it and to verify the data output. This practice helps you ensure that your customers are presented with complete and accurate data.

Before extensions, the output of a report was saved to a file, but extensions deployed to Dynamics 365 Business Central cannot access the file system and therefore must save the output of a report to a stream. Codeunit 131007 `Library - Report Dataset` offers a high-level API for running and testing the output of reports that does not require direct access to the file system.

Example

The following example shows how to initialize the codeunit 131007 `Library - Report Dataset` by using the `RunReportAndLoad` method. This method is preferred as it will run the report and initialize the `Library - Report Dataset` codeunit. To verify the output, call either the `AssertElementWithValueExists` or the `AssertElementWithValueNotExist` method. The other methods in the library should work as well if they do not contain "Tag" in the name. `RUNREQUESTPAGE` and `[RequestPageHandler]` are optional and you can use them when you want to open the request page.

TIP

If you want to run the report separately and load the data from the input stream manually, you can use the `LoadDataFromInstream` method.

```
codeunit 50105 MyReportTesting
{
    Subtype = Test;

    procedure TestingReports();
    var
        XmlParameters: Text;
        LibraryReportDataset: Codeunit "Library - Report Dataset";
        GenJournalLine: Record "Gen. Journal Line";
    begin
        // Run the Report Remittance Advice - Journal.
        XmlParameters := REPORT.RUNREQUESTPAGE(REPORT::"Remittance Advice - Journal");
        LibraryReportDataset.RunReportAndLoad(REPORT::"Remittance Advice - Journal", GenJournalLine,
        XmlParameters);

        // Verifying Total Amount on Report.
        LibraryReportDataset.AssertElementWithValueExists('Amt_GenJournalLine', GenJournalLine.Amount);
    end;

    [RequestPageHandler]
    procedure RemittanceAdviceJournalRequestPageHandler(var RemittanceAdviceJournal: TestRequestPage 399);
    begin
        // Empty handler used to close the request page. We use default settings.
    end;
}
```

Any changes done in the handler above will result in the `XmlParameters` being changed and applied automatically when the report runs. Examples of the implementation in the existing tests are in `Codeunit 133770` and

Remarks

[TestRequestPage.SaveAsXML](#) uses a different format than [REPORT.SAVEASXML](#) or [REPORT.SaveAs](#) by serializing the output of **Report Previewer**. This is a component that will be deprecated in the future and replaced with the new methods that can be used for the new tests. Another difference is that [TestRequestPage.SaveAsXML](#) requires files to be saved to disk and loaded, while other methods work in memory, making them more efficient.

NOTE

The existing tests still need support and the codeunit solves this problem by supporting both formats for now.

[TestRequestPage.SaveAsXML](#) uses Tags for values, while the new format uses attributes. This means that you cannot use any public method that contains "Tag" in the name to test the reports generated in the memory.

See Also

[Reports Overview](#)

Creating a Word Layout Report

3/31/2019 • 2 minutes to read

When you create a new report, there are two things you have to think about; defining the report dataset of data items and columns, and then designing the report layout. These steps will show how to create a very simple report based on a Word layout. For more information about the report object, see [Report Object](#).

Create a Word layout report

To facilitate testing your report layout, the following example extends the Customer List page with a trigger that runs the report as soon as the Customer List page is opened.

1. Create a new extension to the **Customer List** page that contains code to run the report, as well as a simple report object by adding the following lines of code:

```
pageextension 50100 MyExtension extends "Customer List"
{
    trigger OnOpenPage();
    begin
        report.Run(Report::MyWordReport);
    end;
}

report 50124 MyWordReport
{
    DefaultLayout = Word;
    WordLayout = 'MyWordReport.docx';
}
```

2. Build the extension (Ctrl+Shift+B) to generate the MyWordReport.docx file.
3. Add the **Customer** table as the data item and the **Name** field as a column to the report by adding the following lines of code. For more information about defining a dataset, see [Report Dataset](#).

```
dataset
{
    dataitem(Customer; Customer)
    {
        column(Name; Name)
        {
        }
    }
}
```

4. Build the extension (Ctrl+Shift+B).
5. Open the generated report layout file in Word.
6. In Word, edit the layout using the **XML Mapping Pane** on the **Developer** tab.

NOTE

If you do not see the Developer tab, go to **Options**, then **Customize Ribbon**, and in the **Main tabs** section, select the **Developer** check box.

7. In Word, in the **Custom XML part**, locate the report, and then open the layout.
8. Right-click on the **Customer** table and select **Repeating** from **Insert Content Control** to add the repeater data item.
9. Right-click on the **Name** field and select **Plain Text** from **Insert Content Control** to add the column as a text box.
10. Save the report layout when you are done and close it.
11. Back in Visual Studio Code, press Shift+F5 to compile and run the report.

You will now see the generated report in preview mode.

See Also

[Report Object](#)

[Creating an RDL Layout Report](#)

Creating an RDL Layout Report

5/21/2019 • 2 minutes to read

When you create a new report for Dynamics 365 Business Central, there are two things you have to think about; defining the report dataset of data items and columns, and then designing the report layout. These steps will show you how to create a very simple report based on an RDL layout. For more information about the report object, see [Report Object](#).

IMPORTANT

RDL layouts can result in slower performance with document reports, regarding actions that are related to the user interface (for example, like sending emails) compared to Word layouts. When developing layouts for document reports, we recommend that you design Word layouts instead of RDL. With Word layouts, reports are not impacted by the security constraints on sandbox appdomains like they are with RDL layouts. From a service perspective, RDL layouts are not trusted, so they will run in a sandbox appdomain that only lives for the current report invocation.

To create and modify RDL report layouts, you use SQL Server Report Builder or Visual Studio Report Designer. For information about required versions of these tools, see [System Requirements](#).

Create an RDL layout report

To facilitate testing your report layout, the following example extends the Customer List page with a trigger that runs the report as soon as the Customer List page is opened.

1. Create a new extension to the Customer List page that contains code to run the report, as well as a simple report object by adding the following lines of code:

```
pageextension 50123 MyExtension extends "Customer List"
{
    trigger OnOpenPage();
    begin
        report.Run(Report::MyRdlReport);
    end;
}

report 50123 MyRdlReport
{
    DefaultLayout = RDLC;
    RDCLLayout = 'MyRDLCReport.rdl';
}
```

2. Build the extension (Ctrl+Shift+B) to generate the MyRDLCReport.rdl file.
3. Add the **Customer** table as the data item and the **Name** field as a column to the report by adding the following lines of code:

```
dataset
{
    dataitem(Customer; Customer)
    {
        column(Name; Name)
        {
        }
    }
}
```

4. Build the extension (Ctrl+Shift+B). The file will be created in the root of the current project.
5. Open the generated report layout file in Microsoft SQL Server Report Builder.
6. Edit the layout by inserting a table.
7. Add the **Name** column from the datasets folder into the table and save the file.
8. Back in Visual Studio Code, press Shift+F5 to compile and run the report.

You will now see the generated report in preview mode.

See Also

[Report Object](#)

[Creating a Word Layout Report](#)

Web Client URL

3/31/2019 • 8 minutes to read

There are several parameters that you can add to the Web client URL to manipulate what is displayed in the client, such as opening a specific company, or targeting a specific page, report, or table. For example, the following URL displays page **9305 Sales Order List** for the CRONUS International Ltd. company:

```
https://businesscentral.dynamics.com/?company=CRONUS%20International%20Ltd.&page=9305
```

The following URL opens report **5 Receivables – Payables** for the same company:

```
http://businesscentral.dynamics.com/?company=CRONUS%20International%20Ltd.&report=5
```

This article describe how you can constuct URLs, which can be useful for including in other sources, such as emails or Word documents, or sending as hyperlinks to other users.

IMPORTANT

Certain data in the URL, such as filters, could be considered sensitive information. Use discretion if you distribute URLs that contain filters, or if it is possible, exclude this information from the address.

URL Syntax

The Web client URL has the following syntax:

```
https://<hostname>[/<aadtenantid>[/sandbox]]/?[company=<companyname>]&[page|report|table=<ID>]&[tenant=<tenantID>]&[mode=<View|Edit|Create>]&[profile=<profileID>]&[bookmark=<bookmark>]&[captionhelpdisabled=<0|1>]&[showribbon=<0|1>]&[shownavigation=<0|1>]&[showuiparts=<0|1>]&[redirect=<0|1>]
```

The URL consists of two parts, the hostname part and the query string. The hostname part includes the protocol (https) and the hostname. The query string part includes everything after `<hostname>`. The query string determines what content to target.

Syntax Key

The following table describes the notation that is used to indicate the syntax.

NOTATION	DESCRIPTION
Text without brackets	Parameters that you must type as shown.
<>	A placeholder for values that you must supply. Do not include the brackets in the address.
[]	Optional parameters. Do not include the brackets in the address.
	A set of values from which to choose. Use one of the options and do not include <code> </code> in the address.

Building the URL

Use the following guidelines to write URL syntax and create a URL:

- You can place parameters in any order after `/?`. For example, the following URLs will yield the same results.

```
https://businesscentral.dynamics.com/?company=CRONUS%20International%20Ltd.&page=9305&mode=View
```

```
https://businesscentral.dynamics.com/?page=9305&mode=View&company=CRONUS%20International%20Ltd.
```

- Separate parameters after `/?` with the ampersand symbol (`&`).
- Use `%20` for any spaces in values, or similar escape sequences for other characters which cannot be used in URLs.
- Enclose values in single quotation marks (`' '`) if they are unescaped.

URL Parameters

The following table describes the parameters of the URL for displaying a page.

PARAMETER	DESCRIPTION
<code>https</code>	Specifies the Internet protocol to use. Only <code>https</code> is supported.
<code>hostname</code>	Specifies the hostname for Dynamics 365, for example, <code>businesscentral.dynamics.com</code> .
<code>aadtenantid</code>	Specifies the unique identifier for an Azure Active Directory (AAD) tenant. The value can be formatted as a GUID or domain name. This is useful to those who work across multiple AAD organizations, such as delegated administrators, support personnel or external accountants, because it allows explicitly targeting an AAD tenant. If this is omitted, you will be directed to the primary AAD tenant or the same AAD tenant that you are currently signed in to.
<code>sandbox</code>	Specifies that the URL should target the Dynamics 365 Business Central sandbox environment instead of a production environment.
<code>company</code>	The name of the company in Dynamics 365 which you want to target. If you only have one company, then you can omit this parameter.
<code>page</code>	Opens a page object.
<code>report</code>	Opens a report object.
<code>table</code>	Opens a table object. This requires special permissions. For more information about opening a table, see Viewing Table Data .

PARAMETER	DESCRIPTION
ID	The ID of the page, report, or table to open.
tenant	(on-premises only) Specifies the ID of the tenant to connect to. You must provide this parameter when Web client is deployed in multitenant architecture. The tenant that you specify must be mounted on the Dynamics 365 Business Central service instance that the Web client connects to. For more information, see Multitenant Deployment Architecture .
mode	<p>Specifies the mode in which to display the page.</p> <ul style="list-style-type: none"> - View The page can only be viewed. The user cannot change data on the page. Note: Worksheet page types only display in the edit mode, even if the value is set to View. - Edit The user can change data on the page. Note: To use the edit mode, the Editable Property of the page in Page Designer must be set to Yes. This mode is not supported for pages of the type List, RoleCenter, and CardPart. If you set the value to Edit, pages of these types still display in the view mode. For List type pages, the user can modify the list by choosing Edit List on the page. - Create Opens a blank page that enables the user to create a new item. Note: The Create mode is not supported for pages of the type CardPart, List, ListPart, RoleCenter, and Worksheet. For pages of the type CardPart, List, and ListPart, the page displays in the view mode. Do not use this mode for Worksheet pages; otherwise you will get an error when you try to open the page.
profile	Specifies the ID of the profile to open.
bookmark	<p>Specifies a record in the underlying table of the page. The value of a bookmark is an alphanumeric string of characters, for example, 27%3bEgAAAAJ7CDAAMQA5ADAANQA4ADkAMw%3d%3 .</p> <p>For the page types Card, CardPart, and Document, the bookmark specifies the record that is shown in the page. For page types List, ListPart, and Worksheet, the bookmark specifies the record that is selected in the list on the page.</p> <p>Important: Bookmarks are generated automatically. You can only determine a value for the bookmark by displaying the page in the Web client and looking at its address. Therefore, a bookmark is only relevant when the address you are working with has been copied from another instance of the page in the Web client.</p>

PARAMETER	DESCRIPTION
captionhelpdisabled	<p>Specifies that the ability to look up Help by selecting a field caption is disabled.</p> <p>If you want the Help look up from the field captions, either omit this parameter or set its value to <code>0</code>, such as <code>captionhelpdisabled=0</code>.</p> <p>If you do not want the Help lookup from field captions, set the value to <code>1</code>, such as <code>captionhelpdisabled=1</code>.</p> <p>Note: The parameter needs to be added at the first request when the user logs on to take effect, adding the parameter on an existing session has no effect.</p>
showribbon	<p>Specifies whether to show the Action bar on the specified page when it opens.</p> <p>If you want the Action bar, either omit this parameter or set its value to <code>1</code>, such as <code>showribbon=1</code>.</p> <p>If you do not want the Action bar, set the value to <code>0</code>, such as <code>showribbon=0</code>.</p> <p>Note: This parameter only works for pages of the list page type.</p>
shownavigation	<p>Specifies whether to show the navigation bar when the specified page opens.</p> <p>If you want the navigation bar, either omit this parameter or set its value to <code>1</code>, such as <code>shownavigation=1</code>.</p> <p>If you do not want the navigation bar, set the value to <code>0</code>, such as <code>shownavigation=0</code>.</p> <p>Note: This parameter only works for pages of the list page type.</p>
showuiparts	<p>Specifies whether to show UI parts when the specified page opens. The default value, if the parameter is not specified, is <code>1</code> which displays the UI parts. Use the value <code>0</code> to not show UI parts.</p> <p>If you want the UI parts, either omit this parameter or set its value to <code>1</code>, such as <code>showuiparts=1</code>.</p> <p>If you do not want the UI parts, set the value to <code>0</code>, such as <code>showuiparts=0</code>.</p> <p>Note: This parameter only works for pages of the list page type.</p>

PARAMETER	DESCRIPTION
<code>redirect</code>	<p>Specifies whether users are presented with an option to download the Business Central App when they open the Web client in a mobile browser in order to improve the user experience.</p> <p>If you do not want to give users this option, set the value to <code>0</code>, such as <code>redirect=0</code>.</p>
<code>extension</code>	<p>Specifies the unique identifier (ID) of an extension that is deployed on the tenant. This parameter is mainly used during the development of the specified extension in a non-production environment. When this parameter is set, only the specified extension is available in the client; all other extensions are ignored and not visible. This enables you to isolate and focus on the behavior of the specified extension only.</p> <p>An extension ID is a 32-digit GUID, such as <code>72CC5E27-BD97-4271-AF55-F77E4471E493</code>. You set this parameter using the format <code>extension={GUID}</code>, for example:</p> <pre>&extension={72CC5E27-BD97-4271-AF55-F77E4471E493}</pre> <p>You can determine an extension ID by opening the extension in Visual Studio Code and looking in the app.json file, or by running the Get-NAVAppManifest cmdlet on the extension package.</p>

Filtering Data on the Page

You can filter the data that is displayed in the page by using the filter parameter in the address. The filter parameter enables you to display only records from the underlying table of the page that have specific values for one or more fields.

Example

The following address displays data in page 9305 only for the customer who has the **Sell-to Customer No.**=10000 and the **Location Code**=Blue.

```
https://businesscentral.dynamics.com/?company=CRONUS%20International%20Ltd.&page=9305&filter='Sell-to Customer No.' IS '10000' AND 'Location Code' IS 'BLUE'
```

Filter Syntax

The filter has the following syntax.

```
&filter='<field>' IS '<value>' [ AND '<field>' IS '<value>']
```

You must include a space or `%20` before and after the `IS` and `AND` operators. You can add the filter anywhere in the address after `/?`.

Filter Parameters

The following table describes the filter parameters.

PARAMETER	DESCRIPTION
<code>field</code>	The name of the table field on which to filter.
<code>IS</code>	Specifies the equal operator.
<code>value</code>	The value of the table field on which to filter.
<code>AND</code>	<p>Use this parameter to specify more than one filter. It specifies an “and” operator for adding additional filters. Place <code>AND</code> between each additional filter.</p> <p>To be included in the page data, the table record must match values for all fields in the filter.</p>

See Also

[Viewing Table Data](#)

Linking to the Dynamics 365 Business Central App

3/31/2019 • 4 minutes to read

The protocol handler for the Business Central App lets you construct a URL for starting the app on a device, such as a phone or tablet. You can then distribute this URL by e-mail or from a Web page to the users.

The Business Central App URL is based on the *ms-businesscentral* URI scheme, which is registered automatically when the app is installed. Invoking a URL based on this scheme will start the app with the provided parameters.

Constructing the URL

To construct a URL, start with *ms-businesscentral* scheme, and then add additional parameters as needed. Some parameters are required and others are optional.

The structure of a Business Central App link is very similar to links for the Web client, and has the following syntax:

```
ms-businesscentral://[<hostname>][/<aadtenantid>][/<sandbox>][?<parameter>=<value>[&<parameter>=<value>]]
```

[] indicates an optional parameter; all other parameters are required.

<> indicate values that you must supply. Do not include the brackets in the address.

Parameters

The following table describes the parameters for the main part of the URL, which are the parameters up to and including [/sandbox/].

PARAMETER	DESCRIPTION	EXAMPLE
hostname	Domain name for the Dynamics 365 Business Central solution or IP address of the computer/server that hosts it. This is required for an ISV Embed solution. For standard Business Central, you use <code>businesscentral.dynamics.com</code> or you can omit this parameter.	<code>ms-businesscentral://businesscentral.dynamics.com/</code> <code>ms-businesscentral:///</code> <code>ms-businesscentral://businesscentral.mysolution.com/</code>
aadtenantid	The unique identifier for an Azure Active Directory (AAD) tenant. The value can be formatted as a GUID or domain name. This is useful to those who work across multiple AAD organizations, such as delegated administrators, support personnel or external accountants, because it allows explicitly targeting an AAD tenant. If this is omitted, you will be directed to the primary AAD tenant or the same AAD tenant that you are currently signed in to.	<code>ms-businesscentral://businesscentral.mysolution.com/mysolution</code>
sandbox	Specifies that the URL should target the Dynamics 365 Business Central sandbox environment instead of a production environment.	<code>ms-businesscentral://businesscentral.dynamics.com/sandbox/</code> <code>ms-businesscentral://businesscentral.mysolution.com/sandbox/</code>

The following table describes the optional parameters that are indicated by `[?<parameter>=<value>[&<parameter>=<value>]]` in the syntax. These parameters are referred to as the *query parameters*.

PARAMETER	DESCRIPTION	EXAMPLE
page	The ID of the page that you want to open directly.	<code>ms-businesscentral:///?page=21</code> <code>ms-businesscentral://businesscentral.mysolution.com/?page=21</code>

PARAMETER	DESCRIPTION	EXAMPLE
bookmark	<p>The bookmark of the record you want to open. The value of a bookmark is an alphanumeric string of characters, for example, <code>19%3bGwAAAAJ7BDEAMAAwADA%3d</code>.</p> <p>For the page types Card, CardPart, and Document, the bookmark specifies the record that is shown in the page. For page types List, ListPart, and Worksheet, the bookmark specifies the record that is selected in the list on the page.</p> <p>Important: Bookmarks are generated automatically. You can only determine a value for the bookmark by displaying the page in the client and looking at its address. Therefore, a bookmark is only relevant when the address you are working with has been copied from another instance of the page.</p>	<pre>ms-businesscentral:///? bookmark=19%3bGwAAAAJ7BDEAMAAwADA%3d</pre> <pre>ms- businesscentral://businesscentral.mysolution.com/? bookmark=19%3bGwAAAAJ7BDEAMAAwADA%3d</pre>
filter	<p>The filter you want to apply to the page.</p> <p>The filter parameter enables you to display only records from the underlying table of the page that have specific values for one or more fields. For more information about filters, see Filtering Data on the Page.</p>	<pre>ms-businesscentral:///? page9305&filter='No. '%20IS%20'1001'</pre> <pre>ms-businesscentral:///?page9305&filter='Sell-to- Customer-No. '%20IS%20'10000 '%20AND%20'Location- Code '%20IS%20'BLUE'</pre> <pre>ms- businesscentral://businesscentral.mysolution.com/? page9305&filter='No. '%20IS%20'1001'</pre> <pre>ms- businesscentral://businesscentral.mysolution.com/? page9305&filter='Sell-to-Customer- No. '%20IS%20'10000 '%20AND%20'Location- Code '%20IS%20'BLUE'</pre>
profile	<p>The name of the profile that you want to use in the client. This determines the Role Center that is opened. If not provided, the default profile is used. Business Manager</p>	<pre>ms-businesscentral:///? profile=BUSINESS%20MANAGER</pre> <pre>ms- businesscentral://businesscentral.mysolution.com/? profile=BUSINESS%20MANAGER</pre>
company	<p>The company that you want to open in the client. If not provided, the default company is used. CRONUS%20International%20Ltd.</p>	<pre>ms- businesscentral:///?'company=CRONUS%20International%20Ltd.'</pre> <pre>ms- businesscentral://businesscentral.mysolution.com/?'company=</pre>
mode	<p>Whether the page opens in view, edit, or create mode. <code>view</code> only lets you see the data on the page, not modify data. <code>edit</code> lets you to modify data on the page. <code>create</code> lets you to modify data on the page and add new entities.</p>	<pre>ms-businesscentral:///?page=21&mode=create</pre> <pre>ms- businesscentral://businesscentral.mysolution.com/? page=21&mode=create</pre>

The query parameters can be in any order. However, the first parameter must be preceded by the `?` symbol, and any additional parameters must be preceded by the `&` symbol.

See Also

[Web Client URL](#)

Working with Translation Files

5/21/2019 • 3 minutes to read

Dynamics 365 Business Central is multilanguage enabled, which means that you can display the user interface (UI) in different languages. To add a new language to the extension you have built, you must first enable the generation of XLIFF files. The XLIFF file extension is .xlf. The generated XLIFF file contains the strings that are specified in properties such as **Caption** and **Tooltip**.

NOTE

To submit an app to AppSource, you must use .xlf translation files.

IMPORTANT

You can use the .xlf translation files approach only for objects from your extension. For translating the base application you still need to use the .txt files approach. For more information, see the **Translation and Localization apps** section below.

Translation and Localization apps

The .xlf files approach cannot be used for translating the base application. If you are working on a translation or localization app (for example for a [country/region localization](#)), you must take the .txt file containing the base application translation, and place the file in the root folder of your extension. When the extension is compiled, the .txt file is then packaged with the extension.

We recommend that you use only one .txt file per language. There is no enforced naming on the .txt files, but a suggested good practice is to name it `<extensionname>.<language>.txt`.

For more information about importing and exporting .txt files, see [How to: Add Translated Strings By Importing and Exporting Multilanguage Files in Dynamics NAV](#).

Generating the XLIFF file

To enable generation of the translation file, you must add a setting in the manifest. In the app.json file of your extension, add the following line:

```
"features": [ "TranslationFile" ]
```

Now, when you run the build command (Ctrl+Shift+B) in Visual Studio Code, a `\Translations` folder will be generated and populated with the .xlf file that contains all the labels, label properties, and report labels that you are using in the extension. The generated .xlf file can now be translated.

IMPORTANT

Make sure to rename the translated file to avoid that the file is overwritten next time the extension is built.

Label syntax

The label syntax is shown in the example below for the **Caption** property:

```
Caption = 'Developer translation for %1', Comment = '%1 is extension name', locked = false, MaxLength=999;
```

NOTE

The `comment`, `locked`, and `maxLength` attributes are optional and the order is not enforced. For more information, see [Label Data Type](#).

Use the same syntax for report labels:

```
labels
{
    LabelName='LabelText',Comment='Foo',MaxLength=999,Locked=true;
}
```

And the following is the syntax for **Label** data types:

```
var
a:Label'LabelText',Comment='Foo',MaxLength=999,Locked=true;
```

The **ML** versions of properties are **not** included in the .xlf file:

- [CaptionML](#)
- [ConstValueML](#)
- [InstructionalTextML](#)
- [OptionCaptionML](#)
- [PromotedActionCategoriesML](#)
- [ReqFilterHeadingML](#)
- [RequestFilterHeadingML](#)
- [ToolTipML](#)

The [TextConst Data Type](#) is not included in the .xlf file either.

The XLIFF file

In the generated .xlf file, you can see a `<source>` element for each label. For the translation, you will now have to add the `target-language` and a `<target>` element per label. This is illustrated in the example below.

```

<ding="utf-8"?>
<xliff version="1.2" xmlns="urn:oasis:names:tc:xliff:document:1.2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xliff:document:1.2 xliff-core-1.2-transitional.xsd">
  <file datatype="xml" source-language="en-US" target-language="da-DK" original="ALProject16">
    <body>
      <group id="body">
        <trans-unit id="PageExtension 1255613137 - Property 2879900210" maxWidth="999" size-unit="char"
translate="yes" xml:space="preserve">
          <source>Developer translation for %1</source>
          <target>Udvikleroversættelse for %1</target>
          <note from="Developer" annotates="general" priority="2">%1 is extension name</note>
          <note from="Xliff Generator" annotates="general" priority="3">PageExtension - PageExtension</note>
        </trans-unit>
      </group>
    </body>
  </file>
</xliff>

```

NOTE

You can have only one .xlf file per language. If you translate your extension to multiple languages, you must have a translation file per language. There is no enforced naming on the file, but a suggested good practice is to name it

```
<extensionname>.<language>.xlf
```

When the extension is built and published, you change the language of Dynamics 365 Business Central to view the UI in the translated language.

See Also

[How to: Add Translated Strings By Importing and Exporting Multilanguage Files in Dynamics NAV](#)

Getting Started Developing Connect Apps for Dynamics 365 Business Central

5/3/2019 • 7 minutes to read

A Connect app establishes a point-to-point connection between Dynamics 365 Business Central and a 3rd party solution or service and is typically created using standard REST API to interchange data. Any coding language capable of calling REST APIs can be used to develop your Connect app. In the following section you can read about how you get started exploring the available APIs for Dynamics 365 Business Central.

To explore and develop against APIs in Dynamics 365 Business Central, you must first sign up for a trial tenant and then you have to connect and authenticate. To do that, follow the steps below.

1. Sign up for [Dynamics 365 Business Central](#).

When you have your tenant, you can sign into the UI to play with the product, as well as [explore the APIs](#)

2. There are two different ways to connect to and authenticate against the APIs.

- Use Azure Active Directory (AAD) based authentication against the common API endpoint:
<https://api.businesscentral.dynamics.com/v1.0/api/v1.0>
- Use basic authentication with username and password (a so-called web service access key) against the common API endpoint that includes the user domain, for example
<https://api.businesscentral.dynamics.com/v1.0/cronus.com/api/v1.0>.

IMPORTANT

When going into production, you must use Azure Active Directory (AAD)/OAuth v2 authentication and the common endpoint <https://api.businesscentral.dynamics.com/v1.0/api/v1.0>. For exploring and initial development, you can use basic authentication.

In the following sections you can read more about setting up the two types of authentication and using both authentication methods in Postman.

Setting up basic authentication

If you prefer to set up an environment with basic authentication just to explore the APIs, you can skip setting up the AAD based authentication for now and proceed with the steps below. If you, however, want to go into production, you must use AAD/OAuth v2 authentication, see the section [Setting up Azure Active Directory \(AAD\) based authentication](#).

1. To set up basic authentication, log into your tenant, and in the **Search** field, enter **Users** and then select the relevant link.
2. Select the user to add access for, and on the **User Card** page, in the **Web Service Access Key** field, generate a key.
3. Copy the generated key and use it as the password for the username.

Now that we have the username and password, we can connect and authenticate. You can do this from code, or API explorers such as Postman or Fiddler. In the [Exploring the APIs with Postman and basic authentication](#) section we will use Postman.

Setting up Azure Active Directory (AAD) based authentication

Sign in to the [Azure Portal](#) to register Dynamics 365 Business Central as an app and thereby provide access to Dynamics 365 Business Central for users in the directory.

1. Follow the instructions in the [Integrating applications with Azure Active Directory](#) article. The next steps elaborate on some of the specific settings you must enable.
2. On the **API permissions** page for your app, click the **Add a permission** button.
3. Make sure the **Microsoft APIs** tab is selected. In the *Commonly used Microsoft APIs* section, click on the **Dynamics 365 Business Central** and select **Delegated permissions**.
4. Ensure that the right permission is checked: **user_impersonation**. Use the search box if necessary.
5. Click the **Add permissions** button.

NOTE

If **Dynamics 365** does not show up in search, it's because the tenant does not have any knowledge of Dynamics 365. To make it visible, an easy way is to register for a [free trial](#) for Dynamics 365 Business Central with a user from the directory.

6. From the **Certificates & secrets** page, in the **Client secrets** section, choose **New client secret**:
 - Type a key description (of instance app secret),
 - Select a key duration of either In 1 year, In 2 years, or Never Expires.
 - When you press the Add button, the key value will be displayed, copy, and save the value in a safe location.

NOTE

You'll need this key later to configure the project in Visual Studio. This key value will not be displayed again, nor retrievable by any other means, so record it as soon as it is visible from the Azure portal.

You have now set up the AAD based authentication. Next, you can go exploring the APIs, see the [Exploring the APIs with Postman and AAD authentication](#) section below.

Exploring the APIs with Postman and basic authentication

In this `Hello World` example, we are going over the basic steps required to retrieve the list of customers in our trial tenant. This example is based on running with basic authentication.

1. First, in Postman, set up a `GET` call to the base API URL.
 - When you call the base API URL, you will get a list of all the available APIs. You can append `$metadata` to the URL to also get information about the fields in the APIs. The list of supported APIs and fields information can also be found in the API documentation.
 - Since we are using basic authentication, we need to include the users domain in the URL, for example, call `GET https://api.businesscentral.dynamics.com/v1.0/<your tenant domain>/api/v1.0`.

NOTE

The parameter `<your tenant domain>` is your default Azure Active Directory GUID.

2. On the **Authorization** tab in Postman select **Basic Auth** in the **Type** and provide the Username and **Web Service Access Key** from above as password.

3. Choose **Send** in Postman to execute the call, and inspect the returned body, which should include a list of the APIs.

Exploring the APIs with Postman and AAD authentication

In this `Hello World` example, we are going over the basic steps required to retrieve the list of customers in our trial tenant. This example is based on running with AAD authentication.

1. First, in Postman, set up a `GET` call to the base API URL.
 - When you call the base API URL, you will get a list of all the available APIs. You can append `$metadata` to the URL to also get information about the fields in the APIs. The list of supported APIs and fields information can also be found in the API documentation, for example, call
`GET https://api.businesscentral.dynamics.com/v1.0/api/v1.0`
2. On the **Authorization** tab in Postman select **OAuth 2.0** in the **Type** and then choose **Get New Access Token**.
3. In the **GET NEW ACCESS TOKEN** window, enter the following information as specified below:
 - In the **Token name** field, choose a descriptive name.
 - In the **Grant type** field, choose **Authorization Code**.
 - In the **Callback URL** field, specify the URL specified as the sign-on URL/Reply URL in the Azure Portal.
 - In the **Auth URL** field, specify a URL such as
`https://login.windows.net/<your tenant domain>/oauth2/authorize?resource=https://api.businesscentral.dynamics.com`
 -
 - In the **Access Token URL** field, specify a URL such as
`https://login.windows.net/<your tenant domain>/oauth2/token?resource=https://api.businesscentral.dynamics.com`
 -
 - In the **Client ID** field, enter the Application ID from the registered app in Azure Portal.
 - In the **Client Secret** field, enter the key generated under **Keys** that you copied in step 6 in the [Setting up Azure Active Directory \(AAD\) based authentication](#).
 - In the **Client Authentication** field, choose the **Send client credentials in body** option.
4. Choose the **Request token** button. The first time you log in, you will get prompted for consent.
5. Scroll down and choose **Use token** button.

An Authorization request header is now added containing the Bearer token.
6. Choose **Send** in Postman to execute the call, and inspect the returned body, which should include a list of the APIs.

NOTE

For OAuth for testing purposes, a multi-tenant AAD app has been created. Admin consent is needed before the ADD app can be used. Information is as follows:

- Grant Type: Implicit
- Callback URL: `https://localhost`
- Auth URL: `https://login.windows.net/common/oauth2/authorize?resource=https://api.businesscentral.dynamics.com`
- Client ID: `060af3ac-70c3-4c14-92bb-8a88230f3f38`

Calling the API

Each resource is uniquely identified through an ID, see the following example of calling `GET <endpoint>/companies`:

```
{
  "@odata.context": "<endpoint>/$metadata#companies",
  "value": [
    {
      "id": "bb6d48b6-c7b2-4a38-9a93-ad5506407f12",
      "systemVersion": "18453",
      "name": "CRONUS USA, Inc.",
      "displayName": "CRONUS USA, Inc.",
      "businessProfileId": ""
    }
  ]
}
```

The resource ID must be provided in the URL when trying to read or modify a resource or any of its children. The ID is provided in parenthesis () after the API endpoint. For example, to GET the "CRONUS USA, Inc." company details, you must call `<endpoint>/companies(bb6d48b6-c7b2-4a38-9a93-ad5506407f12)/`.

All resources, such as customers, invoices etc., live in the context of a parent company, of which there can be more than one in the Dynamics 365 Business Central tenant. Therefore, it is a requirement to provide the company ID in the URL for all resource API calls. To GET all customers in the "CRONUS USA, Inc." company, we must call a GET on the URL `<endpoint>/companies(bb6d48b6-c7b2-4a38-9a93-ad5506407f12)/customers`.

See Also

[Using Deltas With APIs](#)

[Using Filtering With APIs](#)

[Tips for Working with APIs](#)

Instrumenting an Application for Telemetry

6/25/2019 • 3 minutes to read

This article describes how you can implement custom telemetry trace events in your application for collecting telemetry data. This data can then be collected and visualized for analyzing the application against the desired business goals, troubleshooting, and more.

Telemetry overview

One aspect of event logging is collecting data about how the application and your deployment infrastructure is working in order to diagnose conditions and troubleshoot problems that affect operation and performance. For example, this type of event logging includes Dynamics NAV Server events and trace events like SQL and AL method (function) traces.

Another aspect of event logging is *telemetry*, which is collecting data about how your application functions and how it is being used in production. Telemetry can tell you about specific activities that users perform within the application in the production environment. Telemetry is also a useful tool for troubleshooting, especially instances where you are not able to reproduce the conditions experienced by the user or have no access to the user's environment. Telemetry can be divided into different levels or categories, like: telemetry for engineering, telemetry about the business, telemetry for customers.

By default, the Business Central application is instrumented to emit several system telemetry trace events that are recorded in the event log. Custom telemetry trace events enable you to send telemetry data from anywhere in the application code.

Creating custom telemetry events

To create a custom telemetry event, you use the [SENDTRACETAG method](#) in code. You can use the SENDTRACETAG method in any object, trigger, or method. The SENDTRACETAG method has the following syntax:

```
SENDTRACETAG(Tag, Category, Verbosity, Message[, DataClassification])
```

You use the parameters to define the information about the telemetry trace event. This information is can be consumed by event logging tools, and presented in different ways.

PARAMETER	DESCRIPTION
Tag	A text string that assigns an identifier to the telemetry trace event. The tag can consist of letters, numbers, and special characters. Business Central system telemetry events use an auto-generated, auto-incremented, 7-character tag that includes numbers and letters, such as 000002Q. and 000013P. Try to make your tags unique from these telemetry event tags by, for example, using at least 8 characters or a prefix, like Cronus-0001 and Cronus-0002.
Category	A text string that assigns the telemetry trace event to a category that you define. For example, you could have a category for upgrading, user activity, or reporting.

PARAMETER	DESCRIPTION
Verbosity	An enumeration that specifies the severity level of the telemetry trace event. The value can be Critical, Error, Warning, Normal, or Verbose. This severity level can be used by Dynamics NAV Server to filter out lower-level telemetry trace events from being emitted. See Viewing and collecting telemetry data .
Message	A text string that specifies the descriptive message for the telemetry trace event.
DataClassification	A DataClassification data type that assigns a classification to the telemetry trace event. For more information, see Data Classifications .

For example, the following code creates simple telemetry trace events for the five different severity levels.

```
SENDTRACETAG('Cronus-0001', 'Action', VERBOSITY::Critical, 'This is a critical message.',
DATACLASSIFICATION::CustomerContent);
SENDTRACETAG('Cronus-0002', 'Action', VERBOSITY::Error, 'This is an error message.',
DATACLASSIFICATION::EndUserIdentifiableInformation);
SENDTRACETAG('Cronus-0003', 'Action', VERBOSITY::Warning, 'This is a warning message.',
DATACLASSIFICATION::AccountData);
SENDTRACETAG('Cronus-0004', 'Action', VERBOSITY::Normal, 'This is an informational message.',
DATACLASSIFICATION::OrganizationIdentifiableInformation);
SENDTRACETAG('Cronus-0005', 'Action', VERBOSITY::Verbose, 'This is a verbose message.',
DATACLASSIFICATION::SystemMetadata);
```

For a simple test of this code, add it to the `OnRun` trigger of a codeunit, and then run the codeunit. Of course, you can also call the code from other objects, triggers or functions as well.

Viewing and collecting telemetry data

Viewing and collecting telemetry data is done the same way as with other trace events emitted by Business Central, for example, by using tools like Event Viewer, Performance Monitor, PerfView, or logman.

- In Event Viewer, telemetry trace events can be viewed from **Applications and Services Logs**, in the **Microsoft > Dynamics365BusinessCentral > Common** folder. The custom telemetry trace events are recorded in the **Admin** folder. You should be aware that only events with severity level of Warning, Error, and Critical will appear.

For more information, see [Monitoring Business Central Server Events Using Event Viewer](#).

- With other tools like Performance Monitor, PerfView, and logman, you can collect telemetry data by using **Microsoft-DynamicsNAV-Common** as the event trace provider.

For more information, see [Get Started Monitoring Events](#).

IMPORTANT

The Dynamics NAV Server instance includes a configuration setting called **Diagnostic Trace Level** (`TraceLevel` in the `customsettings.config` file) that enables you to specify the lowest severity level of telemetry events to be recorded in the event log, or even turn off telemetry event logging altogether. If you do not see the expected events, then verify the Dynamics NAV Server instance configuration with an administrator. For information, see [Configuring Business Central Server](#).

See Also

[Monitoring Business Central Server Events](#)

Getting started with Microsoft .NET Interoperability from AL

3/31/2019 • 2 minutes to read

You can call .NET type members, including methods, properties, and constructors, from AL code. In this article we will guide you through the process of creating an extension that uses .NET types.

IMPORTANT

.NET Interoperability is only available on-premise. If you want to use this functionality, you must set the `"target": "Internal"` in the app.json file. For more information, see [JSON Files](#).

Enabling .NET Interoperability

.NET interoperability can only be used by applications that target on-premise deployments. See [JSON Files](#) for more information on how to set the correct compilation target.

Declaring a .NET package

Using a .NET type in AL is a two-step process. First, you must declare the type in a **dotnet** package, and then reference it from code using the **DotNet** type.

You start by declaring an empty **dotnet** package in your extension. See the example snippet below.

```
dotnet
{
}
}
```

It is recommended to have only one package per extension that contains all the .NET types which you will be using.

You continue by adding a declaration of the assembly that you will be referencing. For this example, we will use the `microsoft.mscorlib` assembly that contains the core .NET types. A **dotnet** package can contain an unlimited number of assembly declarations. The name of the assembly must be the one defined in the assembly's manifest. See the following example snippet.

```
dotnet
{
  assembly(microsoft.mscorlib)
  {
  }
}
```

By default, the compiler only knows about the location of the `microsoft.mscorlib` assembly. You can reference any compatible assembly by providing the compiler with a path to the assembly's containing folder. This can be achieved by adding the path to assembly's containing folder to the `"al.assemblyProbingPaths"` setting.

NOTE

Any update to an assembly's code is not automatically detected by the compiler. If an assembly has changed, then you must restart your development environment.

You continue by adding a reference to a type from the referenced assembly. In this example, we will use `System.DateTime` from `microsoft` and we will give it the alias `MyDateTime`. The type must be referenced using its fully-qualified name. The alias is used for referencing the .NET type from code. If an alias is not provided, the compiler will use the .NET type name. A .NET assembly declaration can contain any number of type declarations. See the example below.

```
dotnet
{
    assembly(microsoft)
    {
        type(System.DateTime; MyDateTime){}
    }
}
```

Using a .NET type from AL code

From this point on, we can reference the .NET type from AL code using its given alias, as shown in the example below.

```
dotnet
{
    assembly(microsoft)
    {
        type(System.DateTime; MyDateTime){}
    }
}

pageextension 50100 CustomerListExt extends "Customer List"
{
    trigger OnOpenPage();
    var
        now: DotNet MyDateTime;
    begin
        now := now.UtcNow();
        Message('Hello, world! It is: ' + now.ToString());
    end;
}
```

Publishing your extension

The AL Language extension, including the AL compiler, and the server to which you publish your code are completely decoupled. When publishing, the server re-compiles your code and tries to resolve all the references to external assemblies. In order for the compilation to succeed, the server must be able to locate and load all the referenced assemblies and types.

The server will search the global assembly cache (GAC), the **Add-ins** folder, and the **Add-in** table. You must manually install any custom assembly in one of these locations.

See Also

[Getting Started with AL](#)

[.NET Control Add-Ins](#)

[Subscribing to Events in a .NET Framework Type](#)

[Serializing .NET Framework Types](#)

.NET Control Add-Ins

3/31/2019 • 3 minutes to read

In Dynamics 365 Business Central on-premises you can use existing .NET and Javascript control add-ins from the AL Language through .NET interoperability. It is recommended that you convert your existing .NET and Javascript add-ins to native AL control add-ins that are supported both on-premises and in the cloud. For more information about native AL control add-ins, see [Control Add-In Object](#).

To declare the usage of a .NET or Javascript add-in in AL, you need three critical pieces of information about the .NET type that represent the interface of the add-in. These are the name of the assembly containing the add-in, the name of the control add-in, and the name of the class that implements the control add-in. We will show how to retrieve this information for the `Microsoft.Dynamics.Nav.Client.PingPong` control add-in that ships with Business Central.

The name of the assembly can be retrieved from the `AssemblyName` element in the `.csproj` file associated with the .NET project that represents the control add-in. In this case the name of the assembly is

```
Microsoft.Dynamics.Nav.Client.PingPong .
```

NOTE

If you do not have access to the `.csproj` file, you can determine the name of the assembly by following the instructions in [How to: Determine an Assembly's Fully Qualified Name](#).

The following code sample contains the stub definition of the `Microsoft.Dynamics.Nav.Client.PingPong` .NET add-in.

```

namespace Microsoft.Dynamics.Nav.Client.PingPong
{

    /// <summary>
    /// Add-in for pinging the server from the client. The client will respond with a pong.
    /// </summary>
    [ControlAddInExport("Microsoft.Dynamics.Nav.Client.PingPong")]
    public class PingPongAddIn : WinFormsControlAddInBase
    {

        /// <summary>
        /// Event will be fired when the AddIn is ready for communication through its API
        /// </summary>
        [ApplicationVisible]
        public event MethodInvoker AddInReady;

        /// <summary>
        /// Event will be fired when the specified time by the ping has elapsed.
        /// </summary>
        [ApplicationVisible]
        public event MethodInvoker Pong;

        /// <summary>
        /// Starts the ping process.
        /// </summary>
        /// <param name="milliseconds">Number of milliseconds before ponging.</param>
        /// <remarks>If a milliseconds are less than the minimum then the MinimumValue is used.</remarks>
        [ApplicationVisible]
        public void Ping(int milliseconds)
        {
            ...
        }
    }
}

```

The next needed piece of information is the namespace-qualified name of the type annotated with the `ControlAddInExport` attribute. This is the type that provides the implementation of the control add-in and which exposes members annotated with the `ApplicationVisible` attribute to the AL runtime. In this example this is `Microsoft.Dynamics.Nav.Client.PingPong.PingPongAddIn`.

The `ControlAddInExport` attribute's constructor takes as an argument the name of the control add-in, as represented in the runtime, and in existing C/AL code. In this example, the name of the control add-in is `Microsoft.Dynamics.Nav.Client.PingPong`. This was the last component needed to construct a declaration for this .NET control add-in in AL. The name of the assembly is used in creating the `assembly` construct, the namespace-qualified name of the type is used as the first element in the `type` declaration, and the name of the control add-in is used as the alias of the type. You complete the declaration by setting the `IsControlAddIn` property to true. This property is used to tell the AL compiler to treat the given type declaration as a .NET control add-in declaration.

```

dotnet
{
    assembly("Microsoft.Dynamics.Nav.Client.PingPong")
    {
        type("Microsoft.Dynamics.Nav.Client.PingPong.PingPongAddIn"; "Microsoft.Dynamics.Nav.Client.PingPong")
        {
            IsControlAddIn = true;
        }
    }
}

```

You can now use the `Microsoft.Dynamics.Nav.Client.PingPong` from AL, just as you use a native control add-in.

```

page 50100 MyPage
{
    layout
    {
        area(Content)
        {
            usercontrol(PingPongControl; PingPongAddIn)
            {
                trigger Pong()
                begin
                    Message('Pong received.');
```

Remarks

Only members of the .NET type implementing the control add-in that are annotated with the `ApplicationVisibleAttribute` will be accessible from AL. Usages of .NET control add-ins in C/AL are automatically converted to AL by the [Txt2Al conversion tool](#), but the code will only compile, if you manually insert the declaration of the control add-in, as outlined above.

If within the same project you have a native AL control add-in and a .NET add-in with the same name, the .NET add-in will be the one used.

See Also

[Getting Started with AL](#)

[Control Add-In Object](#)

[Getting started with Microsoft .NET Interoperability from AL](#)

[Subscribing to Events in a .NET Framework Type](#)

[Serializing .NET Framework Types](#)

[How to: Determine an Assembly's Fully Qualified Name](#)

Subscribing to Events in a .NET Framework Type

3/31/2019 • 2 minutes to read

With .NET Framework interoperability in Dynamics 365 Business Central objects, you can configure a DotNet variable to subscribe to events that are published by a .NET Framework type. Events are handled by triggers in the AL code of the Business Central object.

You start by declaring in AL the usage of two .NET types from the `System` assembly. The first type is `System.Timers.Timer` and it will be used to generate .NET events. The second one is called `System.Timers.ElapsedEventArgs` and it is required for creating a subscriber to the `Elapsed` event emitted by the `Timer` type.

```
dotnet
{
    assembly(System)
    {
        type(System.Timers.Timer; MyTimer) {}
        type(System.Timers.ElapsedEventArgs; MyElapsedEventArgs) {}
    }
}
```

You can only subscribe to events that are emitted by global variables of the .NET type marked with the `WithEvents` attribute. For all the global variables that are marked with this attribute, the compiler will expose the events available on the type as triggers on the variable. The syntax for declaring these triggers is `{VariableName}::{EventName}(...ParameterList)`, but IntelliSense will offer suggestions for the event name and autocomplete the parameter list.

```
pageextension 50101 CustomerListExt extends "Customer List"
{
    var
        [WithEvents]
        timer: DotNet MyTimer;

    trigger OnOpenPage()
    begin
        SetupTimer();
    end;

    procedure SetupTimer()
    begin
        timer := timer.Timer(2000);
        timer.AutoReset := true;
        timer.Enabled := true;
        timer.Start();
    end;

    trigger timer::Elapsed(sender: Variant; e: DotNet MyElapsedEventArgs)
    begin
        // Print a message when this event is published
        Message('%1', e.SignalTime());
        timer.Stop();
    end;
}
```

See Also

[Getting started with Microsoft .NET Interoperability from AL](#)
[.NET Control Add-Ins](#)
[Serializing .NET Framework Types](#)
[Method Attributes](#)

Serializing .NET Framework Types

3/31/2019 • 6 minutes to read

In Microsoft .NET Framework, serialization is the process of converting an object into a format that can be transmitted across a network connection. Microsoft .NET Framework interoperability uses serialization for communication between client-side .NET Framework objects and server-side .NET Framework objects. When you configure DotNet variables in a Dynamics 365 Business Central object, you can specify .NET Framework objects to target either the Business Central Windows client or Business Central Server. In some cases, a client-side object and a server-side object must communicate and share data, such as return values and parameters. The serialization occurs when the following conditions are true:

- When a server-side object is assigned to a client-side object, and vice-versa.
- When a server-side object is passed as a parameter in a method call from the server to a client-side object, and vice-versa.

Serialization requires that the .NET Framework types that are used by the DotNet variables are serializable. Many types in the Microsoft .NET Framework class library are already serializable. If you are using a .NET Framework type that cannot be serialized, then you must modify the type to make it serializable.

IMPORTANT

For the Business Central Web client, you cannot implement Microsoft .NET Framework interoperability objects that target the client.

Making a Type Serializable

There are two ways that you can make a .NET Framework type serializable. You can implement basic serialization by applying the [System.SerializableAttribute](#) attribute to the type or you can implement custom serialization by using [System.Runtime.Serialization.ISerializable](#) interface.

Basic Serialization Using SerializableAttribute

Basic serialization uses the .NET Framework to automatically serialize an object. To implement basic serialization on a type, you decorate the type with the [SerializableAttribute](#) class as shown in the following example.

```
[Serializable]
public class MyObject
{
    code
}
```

This method requires that you have access to the source code of the .NET Framework assembly.

Basic Serialization of Date Fields

You can use basic serialization only if all data fields in the type are serializable. Fields that are calculated at runtime cannot be serialized. If a field cannot be serialized, then at runtime, the serialization process will throw an exception and the AL code execution will fail.

You can exclude fields from the serialization process by decorating the field with the [System.NonSerializedAttribute](#) class.

Custom Serialization Using ISerializable Interface

With custom serialization, you can create an object that controls the serialization of types in another object. This method is useful when you do not have access to the source code of the assembly that contains the .NET Framework types that you are implementing with .NET Framework interoperability. The custom object specifies which types will be serialized and how serialization will be done.

To implement custom serialization, you create a class that implements the [ISerializable](#) interface, and decorate the class with [SerializableAttribute](#). In most cases, you must also implement the [System.Runtime.Serialization.ISerializable.GetObjectData](#) method and a special constructor that is used when the source object is deserialized. You use the **GetObjectData** method to populate the [SerializationInfo](#) the data that is required to serialize the source object at runtime.

The common language runtime calls the constructor during deserialization to construct a replica of the source object. The constructor takes two parameters, a **SerializationInfo** type and a [System.Runtime.Serialization.StreamingContext](#) type. The **StreamingContext** parameter describes the source and destination of a given serialized stream.

Custom Serialization Example

The following code example demonstrates a custom serialization object that implements the basic functionality that is required for compliance with the **ISerializable** interface. In the first procedure of this example, you create a .NET Framework assembly that includes a serializable type. In the second procedure, in the Business Central development environment, you create a codeunit that includes two DotNet variables for the serializable type. You set one variable to target the Business Central Windows client and the other to target the Business Central Server. In AL code, you add code that transfers the value for the DotNet variable on the Business Central Server to the Business Central Windows client. You will also add code that verifies that the data transfer is successful.

To create the custom serialization object

1. In Microsoft Visual Studio, create a C# Class Library project called *SerializationSample*.
2. Add the following code.

```

using System;
using System.Runtime.Serialization;

[Serializable]
public class SerializeWithInterface : ISerializable
{
    // Defines a field that will not be serialized.
    [NonSerialized]
    private string notSerializedField;
    // Defines two fields that will be serialized.
    private int serializedIntField;
    private string serializedStringField;
    // Specifies literal field names.
    private const string serializedIntFieldName = "serializedIntField";
    private const string serializedStringFieldName = "serializeStringField";

    // Defines a default constructor that initializes the object with default values.
    public SerializeWithInterface()
    {
        this.serializedIntField = 1;
        this.notSerializedField = string.Empty;
        this.serializedStringField = string.Empty;
    }

    // Defines the protected constructor that is required by the ISerializable interface.
    // Data is stored in the SerializationInfo argument and is extracted using the GetValue method.
    protected SerializeWithInterface(SerializationInfo si, StreamingContext context)
    {
        this.serializedStringField = (string)si.GetValue(serializedStringFieldName, typeof(string));
        this.serializedIntField = (int)si.GetValue(serializedIntFieldName, typeof(int));
    }

    // Fills the SerializationInfo object with data that must to be sent to the replicated object.
    // Data is stored in the dictionary using the AddValue method.
    // The SerializationInfo object is a key/value dictionary.
    // The name you use to store the value must match the name used in the constructor.
    // For this reason, a string constant is used.
    public void GetObjectData(SerializationInfo info, StreamingContext context)
    {
        info.AddValue(serializedStringFieldName, this.serializedStringField);
        info.AddValue(serializedIntFieldName, this.serializedIntField);
    }

    // Remaining class implementation is irrelevant for the serialization process.
    // The SerializedStringField property is used by AL code to verify that the contained data is
    transferred between the server and client.
    // SerializedStringField gets or sets the internal serialized string field.
    public string SerializedStringField
    {
        get { return this.serializedStringField; }
        set { this.serializedStringField = value; }
    }
}

```

3. Build the project.

4. Copy the SerializationSample.dll to the **Add-ins** folder of the Business Central Windows client and Business Central Server installation folders.

By default, the path of the Business Central Windows client installation folder is C:\Program Files (x86)\Microsoft Dynamics 365 Business Central\140\RoleTailored Client\Add-ins.

By default, the path of the Business Central Server installation folder is C:\Program Files\Microsoft Dynamics 365 Business Central\130\Service\Add-ins.

1. In the AL Development Environment, add the following code.

```
dotnet
{
    assembly(SerializationSample)
    {
        type(SerializationSample.SerializeWithInterface; SerializeWithInterface){}
    }
}

codeunit 50101 SerializationSample
{
    procedure Testing()
    var
        ServerObject: DotNet SerializeWithInterface;
        ClientObject: DotNet SerializeWithInterface;

    begin
        // Constructor that instantiates the ServerObject object on the server.
        ServerObject := ServerObject.SerializeWithInterface();
        // Constructor that instantiates the ClientObject object on the server.
        ClientObject := ClientObject.SerializeWithInterface();
        // Assign unique values to the data members in the two objects.
        ServerObject.SerializedStringField := 'ServerSide';
        ClientObject.SerializedStringField := 'ClientSide';
        // Transfer the server object to the client object using serialization.
        ClientObject := ServerObject;
        // Test if the objects contain the same data.
        If ClientObject.SerializedStringField <> ServerObject.SerializedStringField then
            Error('Client object does not match the server object.');
```

```
        Message('Server data has been serialized to the client object.');
```

```
    end;
}
```

The line that contains assignment of the **ServerObject** to the **ClientObject** causes the serialization process to run. When completed, the message **Server data has been serialized to the client object** appears, which verifies that the server object has been transferred to the client object.

See Also

[Getting Started with AL](#)

[Getting started with Microsoft .NET Interoperability from AL](#)

[.NET Control Add-Ins](#)

[Subscribing to Events in a .NET Framework Type](#)

[Using Designer](#)

Exporting Permission Sets

3/31/2019 • 2 minutes to read

Permission sets that exist in Dynamics 365 Business Central can be exported and packaged for your extension directly from the client, instead of defining XML by hand.

To export permission sets from Dynamics 365 Business Central

1. In Dynamics 365 Business Central, search for **Permission Sets**, and then choose the relevant link.
2. On the **Permission Sets** page, choose the permissions that you want to export, and then choose **Export Selected Permissions**.
3. In the **Export Permission Sets** dialog, choose to export permission sets only for the application, only for the tenant, or for both.
4. Save the file to your extension folder.
5. Delete the permission sets from Dynamics 365 Business Central.

You can generate a permission set file which contains permissions to all the files in your extension. This will make it easier to start setting up permissions for your app. You can do this by simply creating an extension with some objects.

To export permission sets using Visual Studio Code

1. In Visual Studio Code, open your extension.
2. Create extension with some objects like Page, Report, Table, Query, Codeunit, or XmlPort.
3. Open the command palette using the `Ctrl+Shift+P` keys and select the **AL: Generate permission set containing current extension objects** command.

NOTE

If you do this repeatedly, Visual Studio Code will probe for overwriting the file, there is no support for merging manual corrections into newly generated content.

4. Publish the app.

Now, you have the XML file with default permissions to all your objects.

See Also

[Permissions on Database Objects](#)

[Permissions Property](#)

[TestPermissions Property](#)

Creating and Interacting with an OData V4 Bound Action

3/31/2019 • 2 minutes to read

This topic provides an overview of how to expose a procedure as an OData V4 web service action and how to verify that the service is working as expected.

Declaring the OData bound action

The following example shows you how you can declare an OData bound action on a page exposed as a web service. For that, you need to add a procedure to the `SalesInvoiceCopy` page, expose the procedure using the `[ServiceEnabled]` attribute, and use the `WebServiceActionContext` and `WebServiceActionResultCode` AL types to set the result of the function.

NOTE

Bound actions cannot be added by extending an existing page that has been exposed as a web service.

```

page 50110 SalesInvoiceCopy
{
    ODataKeyFields = "Id";
    SourceTable = "Sales Header";

    layout
    {
        area(Content)
        {
            group(GroupName)
            {
                field(Id; Id)
                {
                    ApplicationArea = All;
                }

                field("No."; "No.")
                {
                    ApplicationArea = All;
                }

                field("Sell-to Customer No."; "Sell-to Customer No.")
                {
                    ApplicationArea = All;
                }
            }
        }
    }

    trigger OnOpenPage()
    begin
        SetRange("Document Type", "Document Type"::Invoice);
    end;

    [ServiceEnabled]
    procedure Copy(var actionContext: WebServiceActionContext)
    var
        FromSalesHeader: Record "Sales Header";
        ToSalesHeader: Record "Sales Header";
        SalesSetup: Record "Sales & Receivables Setup";
        CopyDocMgt: Codeunit "Copy Document Mgt.";
        DocType: Option Quote,"Blanket Order",Order,Invoice,"Return Order","Credit Memo","Posted
Shipment","Posted Invoice","Posted Return Receipt","Posted Credit Memo";
    begin
        SalesSetup.Get;
        CopyDocMgt.SetProperties(true, false, false, false, false, SalesSetup."Exact Cost Reversing Mandatory",
false);

        FromSalesHeader.Get("Document Type", "No.");
        ToSalesHeader."Document Type" := FromSalesHeader."Document Type";
        ToSalesHeader.Insert(true);

        CopyDocMgt.CopySalesDoc(DocType::Invoice, FromSalesHeader."No.", ToSalesHeader);

        actionContext.SetObjectType(ObjectType::Page);
        actionContext.SetObjectId(Page::SalesInvoiceCopy);
        actionContext.AddEntityKey(Rec.FIELDNO(Id), ToSalesHeader.Id);

        // Set the result code to inform the caller that an item was created.
        actionContext.SetResultCode(WebServiceActionResultCode::Created);
    end;
}

```

Registering and publishing the page as a web service

1. Open the Business Central Web Client.
2. In the **Search** box, enter **Web Services**, and choose the related link.
3. In the **Web Services** page, on the **Home** tab, choose **New**.
4. In the **Object Type** column, select **Page**. In the **Object ID** column, enter 43, and in the **Service Name** column, enter `SalesInvoiceCopy`.
5. Select the check box in the **Published** column.
6. When you publish the web service, in the **OData URL** and **SOAP URL** fields, you can see the URLs that are generated for the web service.

Verifying the web service availability

HTTP request

```
POST /ODataV4/Company({companyName})/SalesInvoiceCopy({id})/NAV.Copy
```

Request headers

HEADER	VALUE
Authorization	Bearer {token}. Required.

Example

```
{baseurl}/ODataV4/Company('CRONUS%20USA%2C%20Inc. ')/SalesInvoiceCopy('S-ORD101001')/NAV.Copy
```

See Also

[AL Development Environment](#)

[Getting started with Microsoft .NET Interoperability from AL](#)

[Developing for Multiple Platform Versions](#)

[Exporting Permission Sets](#)

[Discover Events Using the Event Recorder](#)

AL Development Environment

3/31/2019 • 2 minutes to read

This section describes all of the objects that are available with the AL Language development environment for Dynamics 365 Business Central.

TIP

If you are looking for the C/SIDE documentation, visit our [Dynamics NAV library](#).

Defining the AL data model

TO	SEE
Learn about how to define new table objects for your extension.	Table Object
Learn about how to modify and extend existing table objects.	Table Extension Object

Presenting the AL data

TO	SEE
Learn about how to create new page objects for your extension.	Page Object
Learn about how to modify and extend existing page objects.	Page Extension Object
Learn about how to create page customization objects.	Page Customization Object
Learn about how to create profile objects.	Profile Object
Learn about how to create report objects.	Report Object
Learn about how to create xmlport objects.	XmlPort Object
Learn about how to create query objects.	Query Object
Learn about how to create control add-in objects.	Control Add-In Object

Writing AL code

TO	SEE
Learn about writing codeunits for your extension.	Codeunit Object

API for HTTP, JSON, TextBuilder, and XML

For information about the HTTP, JSON, TextBuilder, and XML classes, see [HTTP, JSON, TextBuilder, and XML API Overview](#).

See Also

[Developing Extensions](#)

[Getting Started with AL](#)

[FAQ for Developing in AL](#)

Programming in AL

3/31/2019 • 2 minutes to read

AL is the programming language that is used for manipulating data (such as retrieving, inserting, and modifying records) in a Dynamics 365 Business Central database, and controlling the execution of the various application objects, such as pages, reports, or codeunits.

With AL, you can create business rules to ensure that the data which is stored in the database is meaningful and consistent with the way customers do business. Through AL programming, you can:

- Add new data or transfer data from one table to another, for example, from a journal table to a ledger table.
- Combine data from multiple tables into one report or display it on one form or page.

Where to write AL code

Almost every object in Dynamics 365 Business Central contains triggers where you can add your AL code. Triggers exist for the following objects:

- Tables
- Table fields
- Pages
- Reports
- Data items
- XMLports
- Queries

You can initiate the execution of your AL code from the following:

- Actions
- Any object that has an instantiation of the object that contains AL code. An example of an instantiation is a variable declaration.

NOTE

If the AL code is in a `local` method, then you cannot run it from another object.

Guidelines for placing AL code

We recommend the following guidelines for AL code:

- In general, put the code in codeunits instead of on the object on which it operates. This promotes a clean design and provides the ability to reuse code. It also helps enforce security. For example, typically users do not have direct access to tables that contain sensitive data, such as the **General Ledger Entry** table, nor do they have permission to modify objects. If you put the code that operates on the general ledger in a codeunit, give the codeunit access to the table, and give the user permission to execute the codeunit, then you will not compromise the security of the table and the user will be able to access the table.

- If you must put code on an object instead of in a codeunit, then put the code as close as possible to the object on which it operates. For example, put code that modifies records in the triggers of the table fields.

Reusing code

Reusing code makes developing applications both faster and easier. More importantly, if you organize your AL code as suggested, your applications will be less prone to errors. By centralizing the code, you will not unintentionally create inconsistencies by performing the same calculation in many places, for example, in several triggers that have the same table field as their source expression. If you have to change the code, you could either forget about some of these triggers or make a mistake when you modify one of them.

See Also

[Simple Statements](#)

[Control Statements](#)

[Methods](#)

[System-Defined Variables](#)

[Developing Extensions](#)

[Getting Started with AL](#)

AL Simple Statements

3/31/2019 • 4 minutes to read

AL simple statements are single-line statements that are executed sequentially and do not alter the flow of execution of code. This article explains some of the simple statements in AL.

Assignment statements

Assignment statements assign a value to a variable. The value that you assign to the variable is an AL expression. It can be a constant or a variable, or it can consist of multiple elements of AL expressions. If you use a method call as the value to assign to a variable in an assignment statement, then the value that is assigned is the return value of the method.

You use the ":= " operator for assignment statements.

Example

The following example assigns a constant integer value to an integer variable that you have defined.

```
Count := 1;
```

Example

The following example assigns a value that consists of a constant, an operator, and a variable.

```
Amount := 2 * Price;
```

Example

The following example assigns the return value of the [Open Method \(File\)](#) to a Boolean variable that you have defined.

```
OK := TestFile.Open('C:\temp\simple.xml');
```

The return value of the `Open` method is optional. If you do not handle the return value in your code, then a run-time error occurs when a method returns **false**. The following example causes a run-time error if the file `C:\temp\simple.xml` cannot be opened.

```
TestFile.Open('C:\temp\simple.xml');
```

You can handle the return value by using an if-then statement.

```
if TestFile.Open('C:\temp\simple.xml') then begin
    // continue running
else
    Error(Text001);
```

Method statements

You use method statements to execute either built-in system methods or user-defined (custom) methods. Method

calls may include parameters, which are passed to the method. For more information, see [Calling Methods](#).

AssertError statements

You use AssertError statements in test methods to test how your application behaves under failing conditions. The AssertError keyword specifies that an error is expected at run time in the statement that follows the AssertError keyword.

If a simple or compound statement that follows the AssertError keyword causes an error, then execution successfully continues to the next statement in the test method. You can get the error text of the statement by using the [GetLastErrorText method](#).

If a statement that follows the AssertError keyword does not cause an error, then the AssertError statement causes the following error and the test method that is running produces a FAILURE result:

```
TestAssertErrorFail: FAILURE
```

```
An error was expected inside an AssertError statement.
```

Example

To create a test method to test the result of a failure of a `CheckDate` method that you have defined, you can use the following code. This example requires that you create a method called `CheckDate` to check whether the date is valid for the customized application.

```
InvalidDate := 19000101D;  
InvalidDateErrorMessage := Text001;  
AssertError CheckDate(InvalidDate);  
  
IF GetLastErrorText <> InvalidDateErrorMessage then  
    Error('Unexpected error: %1', GetLastErrorText);
```

This example requires the following variables.

```
var  
    InvalidDate : Date;  
    InvalidDateErrorMessage : Text;  
    Text001 : TextConst 'The date is outside the valid date range.';
```

With statements

The following syntax shows a with-do statement.

```
with <Record> do  
    <Statement>
```

When you work with records, addressing is created as record name, dot (period), and field name:

`<Record>.<Field>`

If you work continuously with the same record, then you can use `with` statements. When you use a `with` statement, you can only specify the record name one time.

Within the scope of `<Statement>`, fields in `Record>` can be addressed without having to specify the record name.

You can nest several `with` statements. If you have identical names, then the inner `with` statement overrules the

outer `with` statement.

Example

This example shows two ways to write the same code that creates a record variable that you can commit later.

```
CustomerRec."No." := '1234';
CustomerRec.Name := 'Windy City Solutions';
CustomerRec."Phone No." := '555-444-333';
CustomerRec.Address := '1241 Druid Avenue';
CustomerRec.City := 'Windy City';
Message('A variable has been created for this customer.');
```

This example requires the following variables.

```
var
    CustomerRec : Record Customer;
```

The following example shows another way to create a record variable that you can commit later:

```
with CustomerRec do begin
    "No." := '1234';
    Name := 'Windy City Solutions';
    "Phone No." := '555-444-333';
    Address := '1241 Druid Avenue';
    City := 'Windy City';
    Message('A variable has been created for this customer.');
```

```
end;
```

Programming conventions

Within `with-do` blocks, do not repeat the name of the object by using the member variable or method.

If you nest a `with-do` block within another explicit or implicit `with-do` block, then the `with-do` block that you create within another `with-do` block must always be attached to a variable of the same type as the variable that is attached to the surrounding `with-do` block. Otherwise, it can be difficult to see what variable that a member variable or method refers to. For example, implicit `with-do` blocks occur in table objects and in pages that have been attached to a record.

Example

The following example demonstrates nested `with-do` blocks. Both `with-do` blocks are attached to a Customer Ledger Entry record variable.

```
with CustLedgEntry do begin
    Insert;
    ...;
    with CustLedgEntry2 do begin
        Insert;
        ...;
    end;
end;
```

Incorrect example

The following example demonstrates incorrect code in which you cannot directly tell which record variable that the `MyField` field refers to.

```
with CustLedgEntry do begin
  ...;
  with VendLedgEntry do begin
    MyField := <Some Value>;
    ...;
  end;
end;
```

See Also

[Control Statements](#)

[Methods](#)

FAQ for Developing in AL

5/24/2019 • 2 minutes to read

This topic contains a number of frequently asked questions and answers to these questions.

How do I get started?

For an overview of developing apps for Dynamics 365 Business Central, see aka.ms/GetStartedWithApps

Next, follow the [Getting Started with AL](#) to set up the tools.

Which version of the AL Language extension should I use?

1. For Dynamics 365 Business Central cloud sandboxes you must use the AL Language extension available in the [Visual Studio Code Marketplace](#).
2. For the latest Developer Preview releases you must use the AL Language extension that is available on the Docker images.

How do I enable the debugger?

To read about enabling the Visual Studio Code Debugger, see here [Debugging](#)

Can I create something similar to Menusuites?

In the AL Language extension, the concept of Menusuites is not supported. The two primary purposes of Menusuites are:

- Making pages searchable
- Making pages accessible through a navigation structure

The first purpose can be achieved in Extensions by using the new properties added to Pages and Reports. For more information, see [Adding Pages and Reports to Search](#).

The second purpose can be achieved by extending the Navigation Pane page and/or by adding Actions to other existing pages that can serve as a navigation starting point. For more information, see [Adding Menus to the Navigation Pane](#).

How do I upgrade Extensions V1 to Extensions V2?

For information on upgrading, see the following topics: [Upgrading Extensions v2](#) and [Converting from Extensions v1 to Extensions v2](#).

File APIs are not available in Extensions V2. What do I do?

Code that relies on temporary files must be rewritten to rely on `InStream` and `OutStream` types. Code that relies on permanent files must be rewritten to use another form of permanent storage.

DotNet types are not available in Extensions V2. What now?

For cloud solutions .NET interop is not available due to safety issues in running arbitrary .NET code on cloud servers.

With the AL Language extension, you can find AL types that replace the most typical usages of .NET like HTTP, JSON, XML, StringBuilder, Dictionaries and Lists. Many .NET usages can be replaced directly by the AL types resulting in much cleaner code. For more information, see [HTTP, JSON, TextBuilder, and XML API Overview](#).

For things that are not possible to achieve in AL code, the recommendation is to use Azure Functions to host the DLL or C# code previously embedded and call that service from AL.

See Also

[Getting Started with AL](#)

[Keyboard Shortcuts](#)

[AL Development Environment](#)

Working with multiple AL project folders within one workspace

3/31/2019 • 2 minutes to read

Visual Studio Code offers the multi-root workspace feature which enables grouping different project folders into one workspace. The AL Language extension also supports the multi-root functionality and allows you to work with multiple AL folders including roots and projects within one workspace.

Working with multiple project folders

Go through the following steps to work simultaneously on several related projects.

1. On the **File** tab of Visual Studio Code, select **Add Folder to Workspace...** .
2. Save the workspace file if you plan to open it again.
This will create a `code-workspace` file that contains an array of folders with either absolute or relative paths. If you want to share your workspace files, choose the relative paths.
3. Modify the settings of your files in the **Settings** editor. You can change your user settings, global workspace settings, or individual folder settings.

For more information about multi-root workspaces in Visual Studio Code, see [Multi-root Workspaces](#).

Grouping a set of disparate project folders into one workspace

It is not mandatory to use only AL-based roots. Different kinds of projects can be mixed, and each AL project will have its configuration values for the following settings:

- `al.packageCachePath`
- `al.enableCodeAnalysis`

The `al.packageCachePath` setting allows you to specify the path to a folder that will act as the cache for the symbol files used by your project. It can be specified in the **User Settings**, **Workspace Settings**, or **Project Settings**. The `al.enableCodeAnalysis` setting allows you to enable the execution of code analyzers on your project. It can likewise be specified in the **User Settings**, **Workspace Settings**, or **Project Settings**.

See also

[Development in AL](#)

[Best Practices for AL](#)

Using the Code Analysis Tool

6/25/2019 • 2 minutes to read

This topic shows how you can use static code analysis tool on an AL project from within Visual Studio Code.

Enabling code analysis

First, follow the steps below to create a simple project in AL.

1. Press **Alt + A, Alt + L** to create a new project.
2. Open the Command Palette **Ctrl+Shift+P** and choose either **User Settings** or **Workspace Settings**.
3. Copy the setting `al.enableCodeAnalysis` to the settings file and set it to `true`: `"al.enableCodeAnalysis": true`.
4. Copy the setting `al.codeanalyzers` to the settings file and then use **Ctrl+Space** to pick from the available code analyzers. Separate the list of code analyzers with commas. For more information about the available analyzers, see [AppSourceCop](#), [CodeCop](#), [PerTenantExtensionCop](#), and [UICop](#).

At this point, the selected analyzers will be run on your project. Next, add some code to the project that will, in the following example, be used to demonstrate a violation of the AA0001 **"There must be exactly one space character on each side of a binary operator such as := + - AND OR =."** code analysis rule.

Adding your own code to the project

In the Visual Studio Code Explorer, open the `HelloWorld.al` file and replace the existing code with the following:

```
pageextension 50100 CustomerListExt extends "Customer List"
{
    trigger OnOpenPage();
    var
        result: Integer;
    begin
        // The following line will trigger the warning
        // AA0001 "There must be exactly one space character on each side
        // of a binary operator such as := + - AND OR =."
        result := 2+2;
        Message('2 + 2 = ' + Format(result));
    end;
}
```

Viewing the results of the code analysis

The code analysis tools will run in the background. You will see the faulty expression underlined and the warning **"There must be exactly one space character on each side of '+'."** will be displayed if you mouse over the underlined code. You can also view the list of issues by selecting the **View** tab of Visual Studio Code and choosing the **Problems** option.

Using the **Ctrl+Shift+B** shortcut to build your project will run the code analysis tools on the entire project and the detected issues will be displayed in the **Output** window of Visual Studio Code. For more information about AL keyboard shortcuts, see [Keyboard shortcuts](#).

Code analyzers

A code analyzer is a library that builds on the compiler's functionality to offer enhanced analysis of the syntax and

semantics of your code at build time. The AL Language extension for Visual Studio Code contains four analyzers:

- **CodeCop** is an analyzer that enforces the official AL Coding Guidelines. For more information about the CodeCop rules, see [CodeCop Analyzer Rules](#).
- **PerTenantExtensionCop** is an analyzer that enforces rules that must be respected by extensions meant to be installed for individual tenants. For more information about the PerTenantExtensionCop rules, see [PerTenantExtensionCop Analyzer Rules](#).
- **AppSourceCop** is an analyzer that enforces rules that must be respected by extensions meant to be published to Microsoft AppSource. For more information about the AppSourceCop rules, see [AppSourceCop Analyzer Rules](#).
- **UICop** is an analyzer that enforces rules that must be respected by extensions meant to customize a user interface. For more information about the UICop rules, see [UICop Analyzer Rules](#).

See also

[Using the Code Analysis Tools with the Ruleset](#)

[Ruleset for the Code Analysis Tool](#)

[Development in AL](#)

[Debugging in AL](#)

Ruleset for the Code Analysis Tool

3/31/2019 • 2 minutes to read

In an AL project, you can use a custom ruleset file to specify how code analysis will report the issues it encounters. Different settings can affect how rules are applied and each ruleset file name must follow the pattern

`<name>.ruleset.json` to benefit from IntelliSense in Visual Studio Code.

NOTE

Use the `ruleset` and `rule` snippets provided by the AL Language extension to create your ruleset.

The following table describes the schema of a ruleset object:

SETTING	MANDATORY	TYPE	VALUE
name	Yes	String	The name of the ruleset.
description	No	String	The description of the ruleset. You can use this to document the purpose of the ruleset.
generalAction	No	Error Warning Info Hidden	The action to apply to all the diagnostics that have rules defined in this file or in other files that have a Default action specified and to all the diagnostics generated by the current set of analyzers that do not have a rule defined. If an included file has a stricter generalAction , that one will be used.
includedRuleSets	No	Array of IncludedRuleSet	List of external ruleset files to include in the current ruleset. The order in which the files are processed is undefined.
rules	No	Array of Rule	Collection of rules to apply to diagnostics generated by analyzers.

An **IncludedRuleSet** is a complex JSON object that defines the inclusion of an external ruleset file in the current ruleset, and has the following properties:

SETTING	MANDATORY	TYPE	VALUE
---------	-----------	------	-------

SETTING	MANDATORY	TYPE	VALUE
path	Yes	String	The path to the included file. For includes specified in the file to which the al.ruleSetPath is set, the path can be absolute or relative to the project folder. For files included from the root ruleset file, the path is relative to the file.
action	Yes	Error Warning Info Hidden None Default	The action to apply for all the diagnostics that have an action specified in the included ruleset that is different from None and Hidden .

A **Rule** is a complex JSON object that specifies how you can process a specific diagnostic. A **Rule** object has the following properties:

SETTING	MANDATORY	TYPE	VALUE
id	Yes	String	The string that uniquely identifies a diagnostic.
action	Yes	Error Warning Info Hidden None	The action to apply if the diagnostic is emitted. There cannot be two rules with the same id and different actions in the same rule file.

Examples

The following example shows a ruleset that sets the severity of rule **AA0001 : There must be exactly one space character on each side of a binary operator such as := + - AND OR =**, provided by the **CodeCop** analyzer to **Error**.

```
{
  "name": "Company ruleset",
  "description": "These rules must be respected by all the AL code written within the company.",
  "rules": [
    {
      "id": "AA0001",
      "action": "Error",
      "justification": "This diagnostic helps to improve readability. It must be respected in all cases."
    }
  ]
}
```

The following example shows a project specific ruleset that extends a company wide ruleset contained in the file **company.ruleset.json** and sets the severity of the rule **AA0005 : Only use BEGIN..END to enclose compound statements**, provided by the **CodeCop** analyzer to **Info**.

```
{
  "name": "Personal Project ruleset",
  "description": "A list of project specific rules",
  "includedRuleSets": [
    {
      "action": "Default",
      "path": "./company.ruleset.json"
    }
  ],
  "rules": [
    {
      "id": "AA0005",
      "action": "Info",
      "justification": "For this specific project, this diagnostic should be informational."
    }
  ]
}
```

See Also

[Using the Code Analysis Tools](#)

[Using the Code Analysis Tools with the ruleset](#)

[AL Development Environment](#)

Using the Code Analysis Tools with the Ruleset

6/25/2019 • 3 minutes to read

This topic shows how you can use a custom ruleset to customize the severity of diagnostics produced by the code analysis tools that are part of the AL Language extension for Visual Studio Code.

Using rulesets with code analysis

First, create a simple project in AL.

1. Press **Alt + A, Alt + L** to create a new project.
2. Open the Command Palette by using the **Ctrl+Shift+P** shortcut and choose either User Settings or Workspace Settings.
3. Copy the setting `al.enableCodeAnalysis` to the settings.json file and set it to `true`:
`"al.enableCodeAnalysis": true`.
4. Copy the setting `al.codeanalyzers` to the settings file and then use **Ctrl+Space** to pick from the available code analyzers. Separate the list of code analyzers with commas. For more information about the available analyzers, see [AppSourceCop](#), [CodeCop](#), [PerTenantExtensionCop](#), and [UICop](#).

At this point, the selected analyzers will be run on your project. Next, add some code to the project that will, in the following example, be used to demonstrate violations of the AA0001 **"There must be exactly one space character on each side of a binary operator such as := + - AND OR =."** code analysis rule.

Add your own code to the project

In the Visual Studio Code Explorer, open the `HelloWorld.al` file and replace the existing code with the following:

```
pageextension 50100 CustomerListExt extends "Customer List"
{
    trigger OnOpenPage();
    var
        result: Integer;
    begin
        // The following line will trigger the warning
        // AA0001 "There must be exactly one space character on each side
        // of a binary operator such as := + - AND OR =."
        result := 2+2;
        Message('2 + 2 = ' + Format(result));
    end;
}
```

On the **View** tab of Visual Studio Code, select the **Problems** option and you will see a warning with the message **"There must be exactly one space character on each side of '+'."** In this case, the problem can be fixed by running the AL Formatter command. For more information, see [AL Formatter](#).

Creating and customizing a ruleset

To create and customize a ruleset of your own, follow the next steps:

1. On the **File** tab in Visual Studio Code, choose **New File**.
2. Save the empty file with a name, for example `<name>.ruleset.json` and make a note of the file path.

3. Add the following code to the `<name>.ruleset.json` file:

```
{
  "name": "My Custom ruleset",
  "rules": [
    {
      "id": "AA0001",
      "action": "None"
    }
  ]
}
```

4. In your project settings set **al.ruleSetPath** to the path to the `<name>.ruleset.json` file, relative to the project root. For more information about custom rules, see [ruleset for the Code Analysis tool](#).

NOTE

Use the `ruleset` and `rule` snippets provided by the AL Language extension to create your ruleset. The ruleset will be applied to all the analyzers enabled for the current project. For more information about selectively enabling analyzers, see [Using the Code Analysis Tools](#).

Running the code analysis

The code analysis will run in the background and you will see the warning **"There must be exactly one space character on each side of '+'."** disappear from the **Problems** option in Visual Studio Code.

To trigger a new compilation manually, use the **Ctrl+Shift+B** shortcut to build your project. For more information about AL keyboard shortcuts, see [Keyboard shortcuts](#).

Limitations

Changing the contents of the ruleset file will not be detected by the AL Language extension. To see the effects of changing the ruleset file, you can try any of the following:

- Set the **al.incrementalBuild** setting to false and trigger a new compilation by using the **Ctrl+Shift+B** shortcut.
- Reload the window.
- In the project settings, change the **al.ruleSetPath** setting to an invalid path. Save the settings file, change back the setting, and save it.

See also

[Ruleset for the Code Analysis Tool](#)

[Using the Code Analysis Tools](#)

[Development in AL](#)

[Debugging in AL](#)

AppSourceCop Analyzer Rules

3/31/2019 • 4 minutes to read

AppSourceCop is an analyzer that enforces rules that must be respected by extensions meant to be published to Microsoft AppSource.

Rules

ID	TITLE	DESCRIPTION	MESSAGEFORMAT	CATEGORY	DEFAULT SEVERITY	ISENABLEDBYDEFAULT
AS0001	Tables cannot be deleted.	Tables cannot be deleted.	Table '{0}' has been deleted.	Upgrade	Error	true
AS0002	Fields cannot be deleted.	Fields cannot be deleted.	Field '{0}' has been deleted from table '{1}'.	Upgrade	Error	true
AS0003	The previous version was not found.	The previous version was not found.	The previous version was not found. Name='{0}', Publisher='{1}', Version'{2}'.	Upgrade	Warning	true
AS0004	Fields cannot change type.	Fields cannot change type.	Field '{0}' has changed type from '{1}' to '{2}'. Type changes are not allowed.	Upgrade	Error	true
AS0005	Fields cannot change name.	Fields cannot change name.	Field '{0}' has changed name to '{1}'. Name changes are not allowed.	Upgrade	Error	true
AS0006	Tables cannot change name.	Tables cannot change name.	Table '{0}' has changed name to '{1}'. Name changes are not allowed.	Upgrade	Error	true
AS0007	Properties cannot change value.	Properties cannot change value.	The property '{0}' has changed value. Value change is not allowed for this property.	Upgrade	Error	true

ID	TITLE	DESCRIPTION	MESSAGEFORMAT	CATEGORY	DEFAULT SEVERITY	ISENABLEDBYDEFAULT
AS0008	Keys cannot change name.	Keys cannot change name.	Key '{0}' has changed name to '{1}'. Name changes are not allowed.	Upgrade	Error	true
AS0009	Key fields cannot be changed.	Key fields cannot be changed.	Key '{0}' has changed the key fields. Changes to the field list are not allowed.	Upgrade	Error	true
AS0010	Keys cannot be deleted.	Keys cannot be deleted.	Key '{0}' has been deleted. Key deletions is not allowed.	Upgrade	Error	true
AS0011	An affix is required.	An affix is required.	The identifier '{0}' must have at least one of the mandatory affixes '{1}'.	Extensibility	Error	true
AS0013	The field identifier must be within the allowed range.	The field identifier must be within the allowed range.	The field identifier '{0}' is not valid. It must be within the allowed range '{1}' - '{2}'.	Extensibility	Error	true
AS0014	The project manifest must contain the allocated identifier range.	The project manifest must contain the allocated identifier range.	The project manifest must contain the allocated identifier range.	Extensibility	Error	true
AS0015	Please enable the TranslationFile feature in the app.json file for your project.	Please enable the TranslationFile feature in the app.json file for your project.	Please enable the TranslationFile feature in the app.json file for your project.	Extensibility	Error	true

ID	TITLE	DESCRIPTION	MESSAGEFORMAT	CATEGORY	DEFAULT SEVERITY	ISENABLEDBYDEFAULT
AS0016	Fields of field class 'Normal' must use the DataClassification property and its value should be different from ToBeClassified.	Fields of field class 'Normal' must use the DataClassification property and its value must be different from ToBeClassified. FlowFields and FlowFilter fields are automatically set to the SystemMetadata data classification.	Field with name '{0}' must use the DataClassification property and its value should be different from ToBeClassified.	Extensibility	Error	true

NOTE

Several rules enforced by the AppSourceCop analyzer are incompatible with rules enforced by the PerTenantExtensionCop. Make sure to enable only one of these at a time.

Configuration

The AppSourceCop analyzer can be further configured by adding a file named `AppSourceCop.json` in the project's root folder. The AL Language extension will offer intellisense for this file.

The following table describes the settings in the `AppSourceCop.json` file:

SETTING	MANDATORY	VALUE
name	No	The name of a previous version of this package with which you want to compare the current package for breaking changes.
publisher	No	The publisher of a previous version of this package with which you want to compare the current package for breaking changes.
version	Yes	The version of a previous version of this package with which you want to compare the current package for breaking changes.
mandatoryAffixes	No	Affixes that must be prepended or appended to the name of all new application objects, extension objects, and fields.

The `name`, `publisher`, `version` properties are used for specifying a previous version of the current package. AppSourceCop will use this information to download the specified package from the server and compare the current package with it. AppSourceCop will report any breaking changes introduced by the current package.

The `mandatoryAffixes` property specifies strings that must be prepended or appended to the names of all new objects, extension objects and fields. By using these affixes, you can prevent clashes between objects added by your extension and objects added by other extensions.

Example

In the following example, we will configure AppSourceCop to validate that all new elements have a name that contains one of the specified affixes.

NOTE

Make sure code analysis is enabled and `${AppSourceCop}` is in the list of enabled code analyzers.

We start by creating the default "Hello world" extension.

```
pageextension 50100 CustomerListExt extends "Customer List"
{
    trigger OnOpenPage();
    begin
        begin
            Message('App published: Hello world');
        end;
    end;
}
```

We continue by adding the configuration file `AppSourceCop.json` in the project's root folder and setting its content to the following.

```
{
  "mandatoryAffixes": [ "Foo", "Bar" ]
}
```

You are immediately greeted by the following error message:

```
AS0011: The identifier 'CustomerListExt' must have at least one of the mandatory affixes 'Foo, Bar'.
```

Prepending **Foo** to the name of the page extension object will fix this error and prevent clashes between this page extension and page extensions added by other developers.

NOTE

It is still possible to use the `mandatoryPrefix` and `mandatorySuffix` properties in the `AppSourceCop.json`, for more information see [AS0011](#).

See Also

[Using the Code Analysis Tool](#)

[Ruleset for the Code Analysis Tool](#)

[Using the Code Analysis Tools with the Ruleset](#)

CodeCop Analyzer Rules

3/31/2019 • 4 minutes to read

CodeCop is an analyzer that enforces the official AL Coding Guidelines.

Rules

ID	TITLE	DESCRIPTION	MESSAGEFORMAT	CATEGORY	DEFAULT SEVERITY	ISENABLEDBYDEFAULT
AA0001	There must be exactly one space character on each side of a binary operator such as := + - AND OR =.	There must be exactly one space character on each side of a binary operator such as := + - AND OR =. The parameter comma operator however, should have no spaces.	There must be exactly one space character on each side of '{0}'.	Readability	Warning	true
AA0002	There must be no space character.	There must be no space character between a unary operator and its argument.	There must be no space character after '{0}'.	Readability	Warning	true
AA0003	There must be exactly one space character between the NOT operator and its argument.	There must be exactly one space character between the NOT operator and its argument.	There must be exactly one space character after '{0}'.	Readability	Warning	true
AA0005	Only use BEGIN..END to enclose compound statements.	Only use BEGIN..END to enclose compound statements.	Only use BEGIN..END to enclose compound statements.	Readability	Warning	true
AA0008	Function calls should have parenthesis even if they do not have any parameters.	Use parenthesis in a function call even if the function does not have any parameters.	You must specify open and close parenthesis after '{0}'.	Readability	Warning	true

ID	TITLE	DESCRIPTION	MESSAGEFORM AT	CATEGORY	DEFAULT SEVERITY	ISENABLEDBYD EFAULT
AA0013	When BEGIN follows THEN, ELSE, DO, it should be on the same line, preceded by one space character.	When BEGIN follows THEN, ELSE, DO, it should be on the same line, preceded by one space character.	When BEGIN follows THEN, ELSE, DO, it should be on the same line, preceded by one space character.	Readability	Warning	true
AA0018	The END, IF, REPEAT, UNTIL, FOR, WHILE and CASE statement should always start a line.	The END, IF, REPEAT, UNTIL, FOR, WHILE and CASE statement should always start a line.	The '{0}' keyword should always start a line.	Readability	Warning	true
AA0021	Variable declarations should be ordered by type.	Variable declarations should be ordered by type. In general, object and complex variable types are listed first followed by simple variables.	Variable declarations should be ordered by type.	Readability	Warning	true
AA0022	Substitute the IF THEN ELSE structure with a CASE.	An IF followed by two or more ELSE IF should be replaced with a CASE.	Substitute the IF THEN ELSE structure with a CASE.	Readability	Warning	true
AA0040	Avoid using nested WITH statements.	It can be difficult to see what variable that a member variable or function refers to, when nesting WITH statements of variables with different types.	This WITH statement is nested inside another WITH statement at '{0}'.	Readability	Warning	true

ID	TITLE	DESCRIPTION	MESSAGEFORMAT	CATEGORY	DEFAULT SEVERITY	ISENABLEDBYDEFAULT
AA0074	TextConst and Label variable names should have an approved suffix.	TextConst and Label variable names should have a suffix (an approved three-letter suffix: Msg, Tok, Err, Qst, Lbl, Txt) describing usage.	TextConst and Label variable '{0}' must have a suffix from this list: Msg, Tok, Err, Qst, Lbl, Txt.	Readability	Warning	true
AA0100	Do not have identifiers with quotes in the name.	Do not have identifiers with quotes in the name.	Do not have identifiers with quotes in the name.	Design	Warning	true
AA0101	Use camel case captions in pages of type API.	For pages of the type API and all their field controls, the Caption property value should be camel-cased to follow the Microsoft REST API Guidelines.	For pages of the type API and all their field controls, the Caption property value should be camel-cased.	Design	Warning	true
AA0102	Use camel case name for field controls in pages of type API.	Field controls in pages of type API should have a camel case name in order to follow the Microsoft REST API Guidelines.	Field controls in pages of type API should have a camel case name.	Design	Warning	true
AA0136	Do not write code that will never be hit.	Do not write code that will never be hit.	Unreachable code detected.	Design	Warning	true
AA0137	Do not declare variables that are unused.	Do not declare variables that are unused.	Variable '{0}' is unused in '{1}'.	Design	Warning	true
AA0139	Do not assign a text to a target with smaller size.	Do not assign a text to a target with smaller size.	Possible overflow assigning '{0}' to '{1}'.	Design	Warning	true

ID	TITLE	DESCRIPTION	MESSAGEFORMAT	CATEGORY	DEFAULT SEVERITY	ISENABLEDBYDEFAULT
AA0161	Only use ASSERTERROR in Test Codeunits.	Only use ASSERTERROR in Test Codeunits.	Only use ASSERTERROR in Test Codeunits.	Design	Warning	true
AA0194	Only write actions that have an effect.	Remember to specify either the 'OnAction' trigger or 'RunObject' property on an action.	Remember to specify either the 'OnAction' trigger or 'RunObject' property on an action.	Design	Warning	true

See Also

[Using the Code Analysis Tool](#)

[Ruleset for the Code Analysis Tool](#)

[Using the Code Analysis Tools with the Ruleset](#)

PerTenantExtensionCop Analyzer Rules

3/31/2019 • 2 minutes to read

PerTenantExtensionCop is an analyzer that enforces rules that must be respected by extensions meant to be installed for individual tenants.

Rules

ID	TITLE	DESCRIPTION	MESSAGEFORMAT	CATEGORY	DEFAULT SEVERITY	ISENABLEDBYDEFAULT
PTE0001	Object ID must be in free range.	Object ID must be in free range.	{0} '{1}' has an ID of [{2}]. It must be between {3} and {4}.	ObjectValidation	Error	true
PTE0002	Field ID must be in free range.	Field ID must be in free range.	Field '{0}' has an ID of [{1}]. It must be between {2} and {3}.	ObjectValidation	Error	true
PTE0003	Functions must not subscribe to CompanyOpen events.	Functions must not subscribe to CompanyOpen events.	Function {0} subscribes to {1}.	ObjectValidation	Error	true
PTE0004	Table definitions must have a matching permission set.	Table definitions must have a matching permission set.	Table '{0}' is missing a matching permission set.	ObjectValidation	Error	true
PTE0005	Property 'target' has invalid value.	'Internal' is a reserved usage for the 'target' property.	App.json 'target' property must not be set to 'Internal'.	PackageValidation	Error	true
PTE0006	Encryption key functions must not be invoked.	Encryption key functions must not be invoked.	Encryption key function '{0}' is not allowed.	PackageValidation	Error	true
PTE0007	Test assertion functions are not allowed in a non-test context.	Test assertion functions are not allowed in a non-test context.	Assertion function '{0}' must not be invoked.	PackageValidation	Error	true

ID	TITLE	DESCRIPTION	MESSAGEFORMAT	CATEGORY	DEFAULT SEVERITY	ISENABLEDBYDEFAULT
PTE0008	Fields must use ApplicationArea property.	Fields must use ApplicationArea property.	Field with name '{0}' must have a value for the ApplicationArea property.	PackageValidation	Error	true
PTE0009	This app.json property must not be used for per-tenant extensions.	The properties 'HelpBaseUrl' and 'SupportedLocales' are reserved for translation apps.	The app.json '{0}' property must not be used for per-tenant extensions.	PackageValidation	Error	true

See Also

[Using the Code Analysis Tool](#)

[Ruleset for the Code Analysis Tool](#)

[Using the Code Analysis Tools with the Ruleset](#)

UICop Analyzer Rules

3/31/2019 • 2 minutes to read

UICop is an analyzer that enforces rules that must be respected by extensions meant to be installed for individual tenants.

Rules

ID	TITLE	DESCRIPTION	MESSAGEFORMAT	CATEGORY	DEFAULT SEVERITY	ISENABLEDBYDEFAULT
AW0001	The Web client does not support displaying the Request page of XMLPorts.	The Web client does not support displaying the Request page of XMLPorts.	The Web client does not support displaying the Request page of the XMLPort '{0}'.	WebClient	Warning	true
AW0002	The Web client does not support displaying both Actions and Fields in Cue Groups. Only Fields will be displayed.	The Web client does not support displaying both Actions and Fields in Cue Groups. Only Fields will be displayed.	The Web client does not support displaying both Actions and Fields in the Cue Group '{0}'. Only Fields will be displayed.	WebClient	Warning	true
AW0003	The Web client does not support displaying Repeater controls containing Parts.	The Web client does not support displaying Repeater controls containing Parts.	The Web client does not support displaying Repeater controls containing Parts.	WebClient	Warning	true
AW0004	A Blob cannot be used as a source expression for a page field.	A Blob cannot be used as a source expression for a page field.	A Blob cannot be used as a source expression for a page field.	WebClient	Warning	true
AW0005	Actions should use the Image property.	Actions should use the Image property.	Action with name '{0}' should have a value for the Image property.	WebClient	Info	true

ID	TITLE	DESCRIPTION	MESSAGEFORMAT	CATEGORY	DEFAULT SEVERITY	ISENABLEDBYDEFAULT
AW0006	Pages and reports should use the UsageCategory and ApplicationArea properties to be searchable.	Pages and reports should use the UsageCategory and ApplicationArea properties to be searchable.	The {0} '{1}' should use the UsageCategory and ApplicationArea properties to be searchable.	WebClient	Info	true
AW0007	The Web client does not support displaying Repeater controls that contain FlowFilter fields.	The Web client does not support displaying Repeater controls that contain FlowFilter fields.	The FlowFilter field '{0}' in the Repeater control '{1}' cannot be displayed by the Web client.	WebClient	Error	true

See Also

[Using the Code Analysis Tool](#)

[Ruleset for the Code Analysis Tool](#)

[Using the Code Analysis Tools with the Ruleset](#)

Isolated Storage

3/31/2019 • 2 minutes to read

Isolated Storage is a data storage that provides isolation between extensions, so that you can keep keys/values in one extension from being accessed from other extensions. Keys/values in the Isolated Storage are accessible through an API. The involved option type is `DataScope`.

The methods supported for `IsolatedStorage` are:

METHOD	DESCRIPTION	FOR MORE INFORMATION, SEE
<code>Set(String, String, [DataScope])</code>	Sets the value associated with the specified key within the extension.	Set(String, String, [DataScope]) Method
<code>Get(String, [DataScope], var Text)</code>	Gets the value associated with the specified key within the extension.	Get(String, [DataScope], var Text) Method
<code>Contains(String, [DataScope])</code>	Determines whether the storage contains a value with the specified key within the extension.	Contains(String, [DataScope]) Method
<code>Delete(String, [DataScope])</code>	Deletes the value with the specified key from the isolated storage within the extension.	ISOLATEDSTORAGE.DELETE Method

See Also

[DataScope Option Type](#)

File Handling and Text Encoding

3/31/2019 • 4 minutes to read

There are several AL methods that you can use to open files, import and export files to and from Dynamics 365 Business Central, and more. For a list of methods, see [File Data Type](#).

The following are recommended best practices for working with files:

- Use fully qualified paths to eliminate ambiguity.
- Be aware of operating system file access restrictions when designing applications that use files. Consider which users have access to files and directories and what Access Control List (ACL) that you need to apply to file directories.

Text encoding

Text encoding is the process of transforming bytes of data into readable characters for users of a system or program. When you import a file as text or as a stream, the text encoding format ensures that all the language-specific characters are represented correctly in Dynamics 365 Business Central. When you export a file as text or as a stream, the text encoding format ensures that all the language-specific characters are represented correctly in the system or program that will read the exported file.

Encoding formats

You can specify text encoding for the following objects.

OBJECT OR DATA TYPE	FOR MORE INFORMATION, SEE
XMLports	TextEncoding Property (XMLports)
File	OPEN Method (File)
BLOB	CREATEINSTREAM Method (BLOB) CREATEOUTSTREAM Method (BLOB)

There are several industry text encoding formats and different systems support different formats. Internally, Dynamics 365 Business Central uses Unicode encoding. For exporting and importing data with an XMLport, it supports MS-DOS, UTF-8, UTF-16, and Windows encoding formats.

Data is imported and exported as follows:

- When data is imported from an external file, it is read using the format that is specified by the **TextEncoding** property or parameter, and then converted to Unicode in Dynamics 365 Business Central.
- When data is exported to an external file, it is converted from Unicode in Dynamics 365 Business Central, and then written to the file in the format that is specified by the **TextEncoding** property or parameter.

You should set the text encoding to the encoding format that is compatible with the system or program that you will be exporting to or importing from. The following sections describe the available text encoding formats.

MS-DOS encoding format

MS-DOS encoding, which is also referred to as OEM encoding, is an older format than UTF-8 and UTF-16, but it is

still widely supported.

MS-DOS encoding requires a different character set for each language. When the property is set to MS-DOS, text is encoded by using the system locale language of the computer that is running Dynamics 365 Business Central service instance. So if you use MS-DOS encoding, you should set the system locale language of server instance computer to match the language of the data that is being imported or exported. For example, if an XMLport includes text in Danish, then you should set the system locale language of the server instance computer to Danish before the XMLport is run.

You should choose **MS-DOS** with XMLports that were created in earlier versions of Dynamics 365 Business Central.

UTF-8 encoding format

UTF-8 encoding is a Unicode Transformation Format that uses one byte (8 bits) to encode each character. UTF-8 is based on the Unicode character set, which includes most characters of all languages in a single character set.

Unlike MS-DOS, when you use UTF-8, you do not have to consider the language settings of Dynamics 365 Business Central service instance or the external system or program that will read or write the data.

UTF-8 is compatible with ASCII so that it will understand files written in ASCII format.

UTF-8 is the most common encoding format and the recommended setting if you are not sure of the format that is supported by the system that you are integrating with.

UTF-16 encoding format

UTF-16 encoding resembles UTF-8 except that UTF-16 uses 2 bytes (16 bits) to encode each character. UTF-16 is also based on the Unicode character set, so you do not have to consider the language setting of Dynamics 365 Business Central service instance or the external system or program that reads or writes the data.

UTF-16 includes two encoding schemes which mandate the byte order: UTF-16LE and UTF-16BE. The schemas are supported as follows:

- When exporting, the file is written using UTF-16LE encoding.
- When importing, the file is read using the UTF-16, UTF-16LE, or UTF-16BE, depending on encoding scheme of the file itself.

A UTF-16 encoded file will typically be larger than the same file encoded with UTF-8, except for Eastern language character sets, which will typically be smaller.

UTF-16 is incompatible with ASCII so that it will not understand files written in ASCII format.

Windows format

Windows encoding is also referred to as ANSI encoding. If you set the text encoding to **Windows**, you can import and export text files that are based on the Windows codepage on the user's computer. As a result, you do not have to consider the language setting of Dynamics 365 Business Central service instance computer or the external system or program that reads or writes the data.

For example, if an XMLport can import bank files from a foreign bank in addition to a domestic bank, use Windows encoding instead of MS-DOS encoding to avoid changing the language of Dynamics 365 Business Central service instance computer.

Windows encoding is compatible with ASCII so that it will understand files written in ASCII format.

See Also

[TextEncoding Property \(XMLports\)](#)

[File Data Type](#)

FlowFields

3/31/2019 • 2 minutes to read

FlowFields display the result of the calculation described in the [CalcFormula Property](#). For example, the **Account Balance** field in the General Ledger Account table shows the balance of the account and is calculated as the sum of the NetAmount fields for all General Journal entries in the account.

FlowFields increase performance in activities such as calculating the balance of your customers. In traditional database systems, this involves a series of accesses and calculations before a result is available. By using FlowFields, the result is immediately available.

FlowFields are not physical fields that are stored in the database. They are a description of a calculation and a location for the result to be displayed. Because the information in FlowFields exists only at run time, values in FlowFields are automatically initialized to 0 (zero). To update a FlowField, use the [CalcFields Method \(Record\)](#). If a FlowField is the direct source expression of a control on a page, then the FlowField is automatically calculated when the page is displayed.

FlowField types

There are seven types of FlowFields. Each is described in the following table.

FLOWFIELD TYPE	FIELD TYPE	DESCRIPTION
Sum	Decimal, Integer, BigInteger, or Duration	The sum of a specified set in a column in a table.
Average	Decimal, Integer, BigInteger, or Duration	The average value of a specified set in a column in a table.
Exist	Boolean	Indicates whether any records exist in a specified set in a table.
Count	Integer	The number of records in a specified set in a table.
Min	Any	The minimum value in a column in a specified set in a table.
Max	Any	The maximum value in a column in a specified set in a table.
Lookup	Any	Looks up a value in a column in another table.

Example

Consider the Customer table in the following illustration. This table contains two FlowFields. The field named **Any Entries** is a FlowField of the Exist type, and the **Balance** field is a FlowField of the Sum type.

Customer (Table data)

Customer	Name	Country/Region Code	Balance (FlowField)	Any Entries (FlowField)
10000	Windy City Solutions	US	60	Yes
10010	Modern Cars Inc.	US	90	Yes
10020		FR	210	Yes
10030		UK	0	No
10040	La Cuisine Française	FR	200	Yes

Customer Entry (Table data)

Customer	Date	Comment	Amount
10000			10
10000			20
10000			30
10010			40
10010			50
10020			60
10020			70
10020			80
10040			90
10040			110

Virtual part of the table data

The figure shows that the value in the Balance FlowField for customer number 10000 (Windy City Solutions) is retrieved from the Amount column in the Customer Entry table. The value is the sum of the amount fields for the entries that have the customer number 10000.

$$\text{Sum} = 10 + 20 + 30 = 60.$$

The values shown in the **Balance** column in the Customer table for customers 10010, 10020, and 10040 are found in the same way. For customer number 10030 the value is 0 (zero), as there are no entries in the Customer Entry table that have a Customer No. that equals 10030.

In this example, the Balance FlowField in the Customer table reflects the sum of a specific subset of the Amount fields in the Customer Entry table. How the calculation of a FlowField is to be made, is defined in a calculation formula. The calculation formula for the **Balance** field is:

```
Sum("Customer Entries".Amount WHERE(CustNo=FIELD(CustNo)))
```

Correspondingly, the **Any Entries** field, which indicates whether any entries exist, has the following definition:

```
Exist("Customer Entries" WHERE(CustNo=FIELD(CustNo)))
```

See Also

[CalcFields Method \(Record\)](#)

[Create FlowFields and FlowFilters](#)

Create FlowFields and FlowFilters

3/31/2019 • 2 minutes to read

This topic describes the procedure and the properties used to create FlowFields and FlowFilters.

A FlowField performs a set of calculations and displays the results immediately. A FlowFilter displays the results based on the user input to calculate the filtered values that will affect the calculation of a FlowField. The FlowFields and FlowFilters are not physical fields; these fields act as a virtual field which does not actually exist in the database. They are a description of a calculation and a location for the result to be displayed which is typically derived in the [CalcFormula Property](#).

For more information about the FlowField type, see [FlowFields](#), and for more information about the FlowFilter type, see [FlowFilter Overview](#).

Classifying the field type

In order to create FlowFields and FlowFilters, you must first classify the field type by using the FieldClass Property. For more information, see [FieldClass Property](#). By classifying the field as a FlowField or a FlowFilter type, you enable the fields to act as a virtual field whose value can be dynamically derived based on the calculation formula.

Calculation formula

A FlowField type is always associated with a calculation formula that determines how the FlowField is calculated. Likewise, the FlowFilter type is associated with the calculation formula. To perform the calculations by using the FlowField and FlowFilter type, you must derive those fields in the calculation formula which you classify in the table. You can choose from several methods of calculations including sum (total), average, maximum value, minimum value, record count, lookup, and more, by using the CalcFormula Property. For more information about the syntax and formulas, see [Calculation Formulas and the CalcFormula Property](#).

Example

In the following example, `MyTable` sets the `Global Dimension 1 Filter` and `Global Dimension 2 Filter` fields whose values are based only on the dimension values included in the filter. `Total Amount` is classified as a FlowField type and here you specify the calculations. Also, this field formulates the currency filter to one single currency because you do not store the filter value on the entries, hence you define the `Currency Filter` as a FlowFilter type. Lastly, the `Total Amount` displays the results immediately based on the filters you apply in the user interface.

```

table 50123 MyTable
{
    fields
    {
        field(1;MyField; Decimal)
        {
            Description='New field';
        }
        field(3;"no."; Text[20])
        {
            Description='Serial number of the service';
        }
        field(4;"Global Dimension 1 Filter"; Code[20])
        {
            FieldClass = FlowFilter;
        }
        field(5;"Global Dimension 2 Filter"; Code[20])
        {
            FieldClass = FlowFilter;
        }
        field(6;"Currency Filter"; Code[10])
        {
            FieldClass = FlowFilter;
        }
        field(2; "Total Amount"; Decimal)
        {
            FieldClass = FlowField;
            CalcFormula = Sum("Detailed Cust. Ledg. Entry"."Amount (LCY)"
                WHERE ("Customer No."=FIELD("No."),
                    "Initial Entry Global Dim. 1"=FIELD("Global Dimension 1 Filter"),
                    "Initial Entry Global Dim. 2"=FIELD("Global Dimension 2 Filter"),
                    "Currency Code"=FIELD("Currency Filter")
                ) );
        }
    }
}

```

See Also

[FlowFields](#)

[FlowFilter Overview](#)

[Calculation Formulas and the CalcFormula Property](#)

Extensible Enums

3/31/2019 • 2 minutes to read

An enumeration type, also known as an enum in programming, is a keyword used to declare a type that consists of a set of named constants. The list of named constants is called the enumeration list. Enums can be used as table fields, local and global variables, and parameters.

To declare an `enum` in AL you must specify an ID and a name. The enumeration list consists of values and each of the values are declared with an ID and a value. The value ID is the ordinal value on the enumeration list and must be unique. The following example shows the declaration of an enum, which can be extended, and has the four values; **None**, **Bronze**, **Silver**, and **Gold**.

```
enum 50121 Loyalty
{
    Extensible = true;

    value(0; None) { }
    value(1; Bronze) { }
    value(2; Silver) { }
    value(3; Gold)
    {
        Caption = 'Gold Customer';
    }
}
```

Enums can be extended in order to add more values to the enumeration list in which case the `Extensible` property must be set to `true`. The syntax for an enum extension, which extends the **Loyalty** enum with the value **Diamond**, is shown below.

```
enumextension 50130 LoyaltyWithDiamonds extends Loyalty
{
    value(50130; Diamond)
    {
        Caption = 'Diamond Level';
    }
}
```

Usage

When referencing a defined enum from code, you use the syntax as illustrated below.

```
enum Loyalty
```

If you want to define an enum as a table field type, use the syntax illustrated below:

```
field(50100; Loyal; enum Loyalty) {}
```

Or, as a variable:

```
var
    LoyaltyLevel: enum Loyalty;
```

In code, you address a specific enum value like in the following example:

```
codeunit 50140 EnumUsage
{
    procedure Foo(p: enum Loyalty)
    var
        LoyaltyLevel: enum Loyalty;
    begin
        if p = p::Gold then begin
            LoyaltyLevel := p;
        end;
    end;
}
```

Example

The following example illustrates how to define an enum extension of `TypeEnum`, using this in a table extension `TableWithRelationExt` and displaying this as a control on a new page.

```
enumextension 50133 TypeEnumExt extends TypeEnum
{
    value(10; Resource) { }
}

tableextension 50135 TableWithRelationExt extends TableWithRelation
{
    fields
    {
        modify(Relation)
        {
            TableRelation = if (Type = const (Resource)) Resource;
        }
    }
}

page 50133 PageOnRelationTable
{
    SourceTable = TableWithRelation;
    SourceTableView = where (Type = const (Resource));
    PageType = List;

    layout
    {
        area(Content)
        {
            repeater(MyRep)
            {
                field(Id; Id)
                {
                    ApplicationArea = All;
                }
                field(Type; Type)
                {
                    ApplicationArea = All;
                }
                field(Relation; Relation)
                {
                    ApplicationArea = All;
                }
            }
        }
    }
}
```

TIP

For another example of how to extend the usage of the `TableRelation` property in connection with enums, see [TableRelation Property](#).

Business Central On-Premises

If you want to extend an existing Dynamics 365 on-premises enum, it is possible to mark a table field in C/SIDE as extensible. To enable running C/SIDE and AL side-by-side, see [Running C/SIDE and AL Side-by-Side](#).

Table field options in C/SIDE have three properties to enable enum support:

PROPERTY NAME	DATA TYPE
Extensible	Boolean, default value is No .
EnumTypeId	Integer
EnumTypeName	Text

Some table fields share options that are semantically identical. In those cases the **EnumTypeId** and **EnumTypeName** must be the same across all the fields. There is no design or runtime check for collision of IDs, but loading generated symbols, see [Running C/SIDE and AL Side-by-Side](#), into the compiler will show collision errors.

Conversions

Conversion to and from `enum` is more strict than for `Options` in C/SIDE.

- An enum can be assigned/compared to an enum of the same type.
- To be backwards compatible we support conversion to/from any `Option` for now.

See Also

[AL Data Types](#)

[TableRelation Property](#)

Table Object

3/31/2019 • 2 minutes to read

Tables are the core objects used to store data in Dynamics 365 Business Central. Regardless of how data is registered in the product - from a web service to a finger swipe on the phone app, the results of that transaction will be recorded in a table.

The structure of a table has four sections. The first block contains metadata for the overall table; the table type. The fields section describes the data elements that make up the table; their name and the type of data they can store. The keys section contains the definitions of the keys that the table needs to support. The final section details the triggers and code that can run on the table.

NOTE

Extension objects can have a name with a maximum length of 30 characters.

IMPORTANT

System and virtual tables cannot be extended. System tables are created in the ID range of 2.000.000.000 and above. For more information about object ranges, see [Object Ranges](#).

Snippet support

Typing the shortcut `ttable` will create the basic layout for a table object when using the AL Language extension in Visual Studio Code.

Table syntax

```

table id MyTable
{
    DataClassification = ToBeClassified;

    fields
    {
        field(1;MyField; Integer)
        {
            DataClassification = ToBeClassified;

        }
    }

    keys
    {
        key(PK; MyField)
        {
            Clustered = true;
        }
    }

    var
        myInt: Integer;

    trigger OnInsert()
    begin

    end;

    trigger OnModify()
    begin

    end;

    trigger OnDelete()
    begin

    end;

    trigger OnRename()
    begin

    end;

}

```

Table example

This table stores address information and has four fields; Address, Locality, Town/City, and County.

```

table 50104 Address
{
    caption = 'Sample table';
    DataPerCompany = true;

    fields
    {
        field(1; Address; Text[50])
        {
            Description = 'Address retrieved by Service';
        }
        field(2; Locality; Text[30])
        {
            Description = 'Locality retrieved by Service';
        }
        field(3; "Town/City"; Text[30])
        {
            Description = 'Town/City retrieved by Service';
        }
        field(4; County; Text[30])
        {
            Description = 'County retrieved by Service';

            trigger OnValidate();
            begin
                ValidateCounty(County);
            end;

        }
    }
    keys
    {
        key(PrimaryKey; Address)
        {
            Clustered = TRUE;
        }
    }

    var
        Msg: TextConst = 'Hello from my method';

    trigger OnInsert();
    begin

    end;

    procedure MyMethod();
    begin
        Message(Msg);
    end;
}

```

See Also

[AL Development Environment](#)

[Table Overview](#)

[Table Extension Object](#)

[Table Keys](#)

[Table Properties](#)

Table Extension Object

3/31/2019 • 2 minutes to read

The table extension object allows you to add additional fields or to change some properties on a table provided by the Dynamics 365 Business Central service. In this way, you can add data to the same table and treat it as a single table. For example, you may want to create a table extension for a retail winter sports store. In your solution you want to have `ShoeSize` as an additional field on the customer table. Adding this as an extension allows you to write code for the customer record and also include values for the `ShoeSize`.

Along with defining other fields, the table extension is where you write trigger code for your additional fields.

When developing a solution for Dynamics 365 Business Central, you will follow the code layout for a table extension as shown in the example below.

NOTE

Extension objects can have a name with a maximum length of 30 characters.

IMPORTANT

System and virtual tables cannot be extended. System tables are created in the ID range of 2,000,000,000 and above. For more information about object ranges, see [Object Ranges](#).

IMPORTANT

Extending tables from Dynamics 365 for Sales is currently not supported.

Snippet support

Typing the shortcut `tableext` will create the basic layout for a table extension object when using the AL Language extension in Visual Studio Code.

Properties

Using a table extension allows you to overwrite some properties on fields in the base table. For a list of Table properties, see [Table and Table Extension Properties](#).

Table extension syntax

```
tableextension Id MyExtension extends MyTargetTable
{
    fields
    {
        // Add changes to table fields here
    }

    var
        myInt: Integer;
}
```

Table extension example

This table extension object extends the Customer table object by adding a field `ShoeSize`, with ID 50116 and the data type `Integer`. It also contains a procedure to check if the `ShoeSize` field is filled in.

```
tableextension 50115 RetailWinterSportsStore extends Customer
{
    fields
    {
        field(50116;ShoeSize;Integer)
        {
            trigger OnValidate();
            begin
                if (rec.ShoeSize < 0) then
                    begin
                        message('Shoe size not valid: %1', rec.ShoeSize);
                    end;
                end;
            end;
        }
    }

    procedure HasShoeSize() : Boolean;
    begin
        exit(ShoeSize <> 0);
    end;

    trigger OnBeforeInsert();
    begin
        if not HasShoeSize then
            ShoeSize := Random(42);
        end;
    end;
}
```

Applies to

Tables

See Also

[AL Development Environment](#)

[Table Overview](#)

[Table Object](#)

[Table and Table Extension Properties](#)

[Table Keys](#)

Table Keys

3/31/2019 • 7 minutes to read

The database management system, which is SQL Server, keeps track of data in a table using a primary key. The primary key is composed of up to 16 fields in a record. The combination of values in fields in the primary key makes it possible to uniquely identify each record.

Primary keys

The primary key determines the logical order in which records are stored, regardless of their physical placement.

Logically, records are stored sequentially in ascending order and are sorted by the primary key. Before adding a new record to a table, SQL Server checks if the information in the record's primary key fields is unique and only then inserts the record into the correct logical position. Records are sorted dynamically so the database is always structurally correct. This allows for fast data manipulation and retrieval.

A table description contains a list of keys. A key is a sequence of one or more field IDs from the table. Up to 40 keys can be associated to a table. The first key in the list is the primary key.

The primary key is always active, and SQL Server keeps the table sorted in primary key order and rejects records with duplicate values in primary key fields. Therefore, the values in the primary key must always be unique. Note that it is not the value in each field in the primary key that must be unique, but it is the combination of values in all fields that make up the primary key.

NOTE

In the development environment, it is technically possible to create a primary key based on up to 20 fields. However, because of SQL Server limitations, only the first 16 are used.

Secondary keys

A secondary key is implemented on SQL Server using an additional structure that is called an *index*. This is like an index that is used in textbooks. A textbook index alphabetically lists important terms at the end of a book. Next to each term are page numbers. You can quickly search the index to find a list of page numbers (addresses), and you can locate the term by searching the specified pages. The index is an exact indicator that shows where each term occurs in the textbook.

When you define a secondary key and mark it as enabled, an index is automatically maintained on SQL Server. The index reflects the sorting order that is defined by the key. Several secondary keys can be active at the same time.

A secondary key can be changed to be disabled, which does not occupy database space, and does not use time during updates to maintain its index. Disabled keys can be re-enabled, although this can be time-consuming because SQL Server must scan the whole table to rebuild the index.

The fields that make up the secondary keys do not always contain unique data, and SQL Server does not reject records with duplicate data in secondary key fields. If two or more records contain identical information in the secondary key, then SQL Server uses the primary key for the table to resolve this conflict.

Unique secondary keys

A key definition includes the [Unique](#) property that you can use to create a unique constraint on the table in SQL

Server. A unique key ensures that records in a table do not have identical field values. With a unique key, when table is validated, the key value is checked for uniqueness. If the table includes records with duplicate values, the validation fails. Another benefit of unique indexes is providing information to the query optimizer that helps produce more efficient execution plans.

Like primary keys, you can create unique secondary keys that are comprised of multiple fields. In this case, it's the combination of the values in the secondary key that must be unique. For example, if you have a **Customer** table, you could create a unique key for the **Name**, **Address**, and **City** fields to make sure that there are no customers that have the same combination of values for these fields.

Unlike primary keys, it is possible to define multiple unique secondary keys on a table.

NOTE

The `Unique` property is not supported in table extension objects.

Clustered and non-clustered keys

A key definition includes the `Clustered` property that you use to create a clustered index. A clustered index determines the physical order in which records are stored in the table. Based on the key value, records are sorted in ascending order. Using a clustered key can speed up the retrieval of records.

There can be only one clustered index per table. By default the primary is configured as a clustered key.

NOTE

The `Clustered` property is not supported in table extension objects.

Sort orders and secondary keys

The following example shows how the primary key influences the sort order when a secondary key is active. The Customer table includes four entries (records), and the records in the Customer table have two fields: Customer Number and Customer Name.

The following is the key list for the Customer table.

KEY	KEY TYPE	DEFINITION
1	Primary	Customer Number
2	Secondary	Customer Name

When you sort by the primary key, the Customer table resembles the following table.

CUSTOMER NUMBER	CUSTOMER NAME
001	Customer C
002	Customer A
003	Customer B
004	Customer C

If you select the secondary key for sorting, then the order is based on the contents of the Customer Name field. Because the contents of these fields are not unique, the records must be sub-sorted according to the primary key.

CUSTOMER NAME	CUSTOMER NUMBER
Customer A	002
Customer B	003
Customer C	001
Customer C	004

NOTE

The two records that have the same Customer Name value are sorted by Customer Number.

How keys affect performance

Searching for specific data is easier if several keys have been defined and maintained for the table that holds the desired data. The indexes for each key provide specific views that enable quick, flexible searches. There are advantages and disadvantages to using many keys, as demonstrated in the following table.

IF YOU	PERFORMANCE IMPROVES WHEN YOU	PERFORMANCE SLOWS WHEN YOU
Increase the number of secondary keys that are marked as active.	Retrieve data in several different sorting sequences because the data is already sorted.	Enter data because indexes for each secondary key must be maintained.
Decide to use only a few keys.	Enter data because a minimal number of indexes are maintained.	Retrieve data. You may have to define or reactivate the secondary keys to get the appropriate sorting. Depending on the size of the database, this can take some time, because the index must be rebuilt.

The decision whether to use a few or many keys is not easy. The choice of appropriate keys and the number of active keys to use should be the best compromise between maximizing the speed of data retrieval and maximizing the speed of data updates (operations that insert, delete, or modify data). In general, it may be worthwhile to deactivate complex keys if they are rarely used.

The overall speed depends on the following factors:

- Size of the database.
- Number of active keys.
- Complexity of the keys.
- Number of records in your tables.
- Speed of your computer and its hard disk.

Defining new keys

You define keys in AL code of a table object. To define keys, add the `keys` keyword after the `fields` definition,

and then add a `key` keyword for each key:

```
keys
{
    key(Name; Fields)
    {
    }
    key(Name; Fields)
    {
    }
}
```

Replace `Name` with descriptive text that you want to use to identify the key. Replace `Field` with the name of a field that you want to use as the key. If you want to include multiple fields in a single key, separate each field with a comma.

The first `key` keyword defines the primary key. Subsequent `key` keywords define secondary keys.

Key properties

There are several properties that configure the behavior of a key, such as the [Enabled](#), [Clustered](#), and [Unique](#) properties:

```
keys
{
    key(PrimaryKey; ID)
    {
        Clustered = true;
    }
    key(CustomerInfo; Name,Address,City)
    {
        Unique = true;
    }
    key(Currency; Currency Code)
    {
        Enabled = false;
    }
}
```

For a more information about the different key properties, see [Key Properties](#).

Restrictions on key modifications

When developing a new version of an extension, be aware of the following restrictions to avoid schema synchronization errors that prevent you from publishing the new version:

- Do not delete existing keys.
- Do not add or remove fields, change the order of fields, or change properties of existing keys.
- Do not add additional unique keys.

See Also

[Tables Overview](#)

[Table Object](#)

[Table Extension Object](#)

Page Object

3/31/2019 • 2 minutes to read

Pages are the main way to display and organize visual data in Dynamics 365 Business Central. They are the primary object that a user will interact with and have a different behavior based on the type that you choose. Pages are designed independently of the device they are to be rendered on, and in this way the same page can be reused across phone, tablet, and web clients.

The structure of a page is hierarchical and breaks down into three sections. The first block contains metadata for the overall page; the type of the page and the source table it is showing data from. The next section; the layout, describes the visual parts on the page. The final section details the actions that are published on the page.

When developing a solution for Dynamics 365 Business Central, you will follow the code layout for a page as shown in the page example below, but for more details on the individual controls and properties that are available, see [Page Property Overview](#).

NOTE

Extension objects can have a name with a maximum length of 30 characters.

Snippet support

Typing the shortcut `tpage` will create the basic layout for a page object when using the AL Language extension in Visual Studio Code.

Card page syntax

```

page Id MyPage
{
    PageType = Card;
    ApplicationArea = All;
    UsageCategory = Administration;
    SourceTable = TableName;
    ContextSensitiveHelpPage = 'my-feature';

    layout
    {
        area(Content)
        {
            group(GroupName)
            {
                field(Name; NameSource)
                {
                    ApplicationArea = All;
                }
            }
        }
    }

    actions
    {
        area(Processing)
        {
            action(ActionName)
            {
                ApplicationArea = All;

                trigger OnAction()
                begin
                    end;
            }
        }
    }

    var
        myInt: Integer;
}

```

List page syntax

```

page Id PageName
{
    PageType = List;
    ApplicationArea = All;
    SourceTable = TableName;

    layout
    {
        area(Content)
        {
            repeater(Group)
            {
                field(Name; NameSource)
                {
                    ApplicationArea = All;
                }
            }
        }
        area(Factboxes)
        {
        }
    }

    actions
    {
        area(Processing)
        {
            action(ActionName)
            {
                ApplicationArea = All;

                trigger OnAction();
                begin
                end;
            }
        }
    }
}

```

Page example

```

page 50101 SimpleCustomerCard
{
    PageType = Card;
    SourceTable = Customer;
    ContextSensitiveHelpPage = 'my-feature';

    layout
    {
        area(content)
        {
            group(General)
            {
                field("No."; "No.")
                {
                    ApplicationArea = All;
                    CaptionML = ENU = 'Hello';

                    trigger OnValidate()
                    begin
                        if "No." < '' then
                            Message('Number too small')
                        end;
                    }

                field(Name; Name)
                {
                    ApplicationArea = All;
                }
                field(Address; Address)
                {
                    ApplicationArea = All;
                }
            }
        }
    }
    actions
    {
        area(Navigation)
        {
            action(NewAction)
            {
                ApplicationArea = All;
                RunObject = codeunit "Document Totals";
            }
        }
    }
}

```

See Also

[AL Development Environment](#)

[Adding Help Links from Pages, Reports, and XMLports](#)

[Page Extension Object](#)

[Page and Page Extension Properties Overview](#)

[Page Properties](#)

[Developing Extensions](#)

[Configure Context-Sensitive Help](#)

Page Extension Object

3/31/2019 • 2 minutes to read

The page extension object extends a Dynamics 365 Business Central page object and adds or overrides the functionality.

The structure of a page is hierarchical and breaks down into three sections. The first block contains metadata for the overall page; the type of the page and the source table it is showing data from. The next section; the layout, describes the visual parts on the page. The final section details the actions that are published on the page.

For more information about the Page and Page Extension objects, see [Pages Overview](#).

NOTE

Extension objects can have a name with a maximum length of 30 characters.

IMPORTANT

The API page type should not be extended by creating a page extension object. Instead, create a new API by adding a [page object](#).

Snippet support

Typing the shortcut `tpageext` will create the basic layout for a table object when using the AL Language extension in Visual Studio Code.

Page extension syntax

```
pageextension Id MyExtension extends MyTargetPage
{
    layout
    {
        // Add changes to page layout here
    }

    actions
    {
        // Add changes to page actions here
    }

    var
        myInt: Integer;
}
```

Page extension examples

The following page extension object extends the Customer Card page object by adding a field control `ShoeSize` to the `General` group on the page. The field control is added as the last control in the group using the `addlast` method. In the actions area, you can see what the syntax looks like for actions that execute triggers and actions that run objects.

```

pageextension 50110 CustomerCardExtension extends "Customer Card"
{
    layout
    {
        addlast(General)
        {
            field("Shoe Size"; ShoeSize)
            {
                ApplicationArea = All;
                Caption = 'ShoeSize';

                trigger OnValidate();
                begin
                    if ShoeSize < 10 then
                        Error('Feet too small');
                    end;
                end;
            }
        }

        modify("Address 2")
        {
            Caption = 'New Address 2';
        }
    }

    actions
    {
        addlast(Creation)
        {
            group(MyActionGroup)
            {
                Action(MyAction1)
                {
                    ApplicationArea = All;
                    Caption = 'Hello!';

                    trigger OnAction();
                    begin
                        Message('My message');
                    end;
                }

                Action(MyAction2)
                {
                    ApplicationArea = All;
                    RunObject = codeunit "Activities Mgt.";
                }
            }
        }
    }

    var
        Msg: TextConst = 'Hello from my method';

    trigger OnOpenPage();
    begin
        Message(Msg);
    end;
}

```

You can reference Report and XMLPort objects and use these objects in the **RunObject** property, as well as, declare variables of the types **Report** and **XMLPort** and call AL methods on them. This page extension object extends the Customer List page object by adding two actions; the first action calls the **Customer - List** report, the second action calls the **Export Contact** xmlport.

```

pageextension 50114 AddCustomerReport extends "Customer List"
{
    actions
    {
        AddLast("&Customer")
        {
            action("Customer List Report")
            {
                trigger OnAction();
                var
                    rep : Report "Customer - List";
                begin
                    rep.Run;
                end;
            }

            action("Export Contact List")
            {
                trigger OnAction();
                var
                    xml : XmlPort "Export Contact";
                begin
                    xml.Run;
                end;
            }
        }
    }
}

```

Applies To

Pages

See Also

[Page Object](#)

[Page and Page Extension Properties](#)

[Developing Extensions](#)

[AL Development Environment](#)

Page Customization Object

3/31/2019 • 2 minutes to read

The page customization object in Dynamics 365 Business Central allows you to add changes to the page layout and actions. The page customization object has more restrictions than the [page extension object](#); when you define a new page customization object, you cannot add variables, procedures, or triggers.

NOTE

Extension objects can have a name with a maximum length of 30 characters.

Snippet support

Typing the shortcut `tpagecust` will create the basic layout for a page customization object when using the AL Language extension in Visual Studio Code.

Page customization syntax

```
pagecustomization MyCustomization customizes MyTargetPage
{
    layout
    {
        // Add changes to page layout here
    }

    actions
    {
        // Add changes to page actions here
    }

    //Variables, procedures and triggers are not allowed on Page Customizations
}
```

Page customization example

The following page customization example `MyCustomization` is initialized to perform changes to **Customer List**. By using the `moveafter` method, `Blanket Orders` is moved next to the `Aged Accounts Receivable` action item. And the `modify` method is used to hide the `NewSalesBlanketOrder` action item.

```
pagecustomization MyCustomization customizes "Customer List"
{
    actions
    {
        moveafter("Blanket Orders"; "Aged Accounts Receivable")

        modify(NewSalesBlanketOrder)
        {
            Visible = false;
        }
    }
}
```

See Also

[Developing Extensions](#)

[AL Development Environment](#)

[Page Object](#)

[Page Extension Object](#)

[Page Extension Properties](#)

Report Object

3/31/2019 • 3 minutes to read

Reports are used to print or display information from a database. You can use a report to structure and summarize information, and to print documents, such as sales quotes and invoices.

Creating a report consists of two primary tasks; the first task is to create the underlying data model and the next is to define the visual layout that displays the data. The report object defines the underlying data model and specifies which database tables and fields to pull data from. When the report is run, that data is displayed in a specified layout; the visual layout, which determines the content and format of a report when it is viewed and printed.

For more information about defining database tables and fields, see [Defining a Report Dataset](#).

You build the layout of a report by arranging data items and columns, and specifying the general format, such as text font and size. There are two types of report layouts; client report definition, also called RDL layouts and Word layouts. RDL layouts are defined in Visual Studio Report Designer or Microsoft SQL Server Reporting Services Report Builder. Word layouts are created using Word. Word layouts are based on a Word document that includes a custom XML part representing the report dataset.

NOTE

Extension objects can have a name with a maximum length of 30 characters.

Snippet support

Typing the shortcut `treport` will create the basic layout for a report object when using the AL Language extension in Visual Studio Code.

Report syntax

```

report Id MyReport
{
    UsageCategory = Administration;
    ApplicationArea = All;

    dataset
    {
        dataitem(DataItemName; SourceTableName)
        {
            column(ColumnName; SourceFieldName)
            {
            }
        }
    }

    requestpage
    {
        ContextSensitiveHelpPage = 'my-feature';
        layout
        {
            area(Content)
            {
                group(GroupName)
                {
                    field(Name; SourceExpression)
                    {
                        ApplicationArea = All;
                    }
                }
            }
        }

        actions
        {
            area(processing)
            {
                action(ActionName)
                {
                    ApplicationArea = All;
                }
            }
        }
    }

    var
        myInt: Integer;
}

```

Report example

The following example is a report that prints the list of customers. The report object defines a dataset of columns from the Customer table. For more information on creating a Word Layout report, see [Creating a Report](#).

```

report 50103 "Customer List"
{
    CaptionML=ENU='Customer List';
    RDLCLayout = 'Customer List Report.rdlc'; // if Word use WordLayout property
    dataset
    {
        dataitem(Customer;Customer)
    }
}

```

```

{
    RequestFilterFields="No.", "Search Name", "Customer Posting Group";
    column(COMPANYNAME;COMPANYNAME)
    {
    }
    column(CurrReport_PAGENO;Customer."no.")
    {
    }
    column(Customer_TABLECAPTION_CustFilter;TABLECAPTION + ': ' + CustFilter)
    {
    }
    column(CustFilter;CustFilter)
    {
    }
    column(Customer_No;"No.")
    {
    }
    column(Customer_Customer_Posting_Group;"Customer Posting Group")
    {
    }
    column(Customer_Customer_Disc_Group;"Customer Disc. Group")
    {
    }
    column(Customer_Invoice_Disc_Code;"Invoice Disc. Code")
    {
    }
    column(Customer_Customer_Price_Group;"Customer Price Group")
    {
    }
    column(Customer_Fin_Charge_Terms_Code;"Fin. Charge Terms Code")
    {
    }
    column(Customer_Payment_Terms_Code;"Payment Terms Code")
    {
    }
    column(Customer_Salesperson_Code;"Salesperson Code")
    {
    }
    column(Customer_Currency_Code;"Currency Code")
    {
    }
    column(Customer_Credit_Limit_LCY;"Credit Limit (LCY)")
    {
        DecimalPlaces=0:0;
    }
    column(Customer_Balance_LCY;"Balance (LCY)")
    {
    }
    column(CustAddr_1;CustAddr[1])
    {
    }
    column(CustAddr_2;CustAddr[2])
    {
    }
    column(CustAddr_3;CustAddr[3])
    {
    }
    column(CustAddr_4;CustAddr[4])
    {
    }
    column(CustAddr_5;CustAddr[5])
    {
    }
    column(Customer_Contact;Contact)
    {
    }
    column(Customer_Phone_No;"Phone No.")
    {
    }
}

```

```

column(CustAddr_6;CustAddr[6])
{
}
column(CustAddr_7;CustAddr[7])
{
}
column(Customer_ListCaption;Customer_ListCaptionLbl)
{
}
column(CurrReport_PAGENOCaption;CurrReport_PAGENOCaptionLbl)
{
}
column(Customer_NoCaption;FIELDCAPTION("No."))
{
}
column(Customer_Customer_Posting_GroupCaption;Customer_Customer_Posting_GroupCaptionLbl)
{
}
column(Customer_Customer_Disc_GroupCaption;Customer_Customer_Disc_GroupCaptionLbl)
{
}
column(Customer_Invoice_Disc_CodeCaption;Customer_Invoice_Disc_CodeCaptionLbl)
{
}
column(Customer_Customer_Price_GroupCaption;Customer_Customer_Price_GroupCaptionLbl)
{
}
column(Customer_Fin_Charge_Terms_CodeCaption;FIELDCAPTION("Fin. Charge Terms Code"))
{
}
column(Customer_Payment_Terms_CodeCaption;Customer_Payment_Terms_CodeCaptionLbl)
{
}
column(Customer_Salesperson_CodeCaption;FIELDCAPTION("Salesperson Code"))
{
}
column(Customer_Currency_CodeCaption;Customer_Currency_CodeCaptionLbl)
{
}
column(Customer_Credit_Limit_LCYCaption;FIELDCAPTION("Credit Limit (LCY)"))
{
}
column(Customer_Balance_LCYCaption;FIELDCAPTION("Balance (LCY)"))
{
}
column(Customer_ContactCaption;FIELDCAPTION(Contact))
{
}
column(Customer_Phone_NoCaption;FIELDCAPTION("Phone No."))
{
}
column(Total_LCY_Caption;Total_LCY_CaptionLbl)
{
}

trigger OnAfterGetRecord();
begin
    CALCFIELDS("Balance (LCY)");
    FormatAddr.FormatAddr(
        CustAddr,Name,"Name 2",'',Address,"Address 2",
        City,"Post Code",County,"Country/Region Code");
end;

}
}

requestpage
{
    SaveValues=true;

```

```

ContextSensitiveHelpPage = 'my-feature';
layout
{
}

actions
{
}
}

labels
{
    LabelName = 'LabelText', Comment = 'Foo', MaxLength = 999, Locked = true;
}

trigger OnPreReport();
var
    CaptionManagement : Codeunit 42;
begin
    CustFilter := CaptionManagement.GetRecordFiltersWithCaptions(Customer);
end;

var
    FormatAddr : Codeunit 365;
    CustFilter : Text;
    CustAddr : ARRAY [8] OF Text[50];
    Customer_ListCaptionLbl : TextConst ENU='Customer - List';
    CurrReport_PAGENOCaptionLbl : TextConst ENU='Page';
    Customer_Customer_Posting_GroupCaptionLbl : TextConst ENU='Customer Posting Group';
    Customer_Customer_Disc_GroupCaptionLbl : TextConst ENU='Cust./Item Disc. Gr.';
    Customer_Invoice_Disc_CodeCaptionLbl : TextConst ENU='Invoice Disc. Code';
    Customer_Customer_Price_GroupCaptionLbl : TextConst ENU='Price Group Code';
    Customer_Payment_Terms_CodeCaptionLbl : TextConst ENU='Payment Terms Code';
    Customer_Currency_CodeCaptionLbl : TextConst ENU='Currency Code';
    Total_LCY_CaptionLbl : TextConst ENU='Total (LCY)';
}

```

See Also

[Creating an RDL Layout Report](#)

[Creating a Word Layout Report](#)

[Adding Help Links from Pages, Reports, and XMLports](#)

[Page Extension Object](#)

[Page Properties](#)

[Developing Extensions](#)

[AL Development Environment](#)

Profile Object

3/31/2019 • 2 minutes to read

The profile object in Dynamics 365 Business Central allows you to build an individual experience for each user profile. Profile object performs a validation to check whether the specified role center page exists, and [page customization objects](#) exists, when you define a new profile object. You can add changes to the page layout, and actions; but you cannot add variables, procedures, or triggers.

NOTE

Extension objects can have a name with a maximum length of 30 characters.

Snippet support

Typing the shortcut `tprofile` will create the basic layout for a profile object when using the AL Language extension in Visual Studio Code.

Profile syntax

```
profile MyProfile
{
    Description = 'My Description';
    RoleCenter = RoleCenter;
    Customizations = Customizations;
}
```

Profile example

The following profile object example performs a validation to check if the `Business Manager` page of type `RoleCenter` exists, and if `MyCustomization` exists, and if it is a page customization object. Then the page customization modifies the layout of the **Customer List** to make the `Name` field invisible using the `modify` method.

```
profile TheBoss
{
    Description = 'The Boss';
    RoleCenter = "Business Manager";
    Customizations = MyCustomization;
}

pagecustomization MyCustomization customizes "Customer List"
{
    layout
    {
        {
            modify(Name)
            {
                Visible = false;
            }
        }
    }
}
```

See Also

[AL Development Environment](#)

[Developing Extensions](#)

[Pages Overview](#)

[Page Customization Object](#)

Codeunit Object

3/31/2019 • 2 minutes to read

A codeunit is a container for AL code that you can use in many application objects. You typically implement business logic in codeunits and call the codeunit from the object that needs to perform that specific logic.

Snippet support

Typing the shortcut `tcodeunit` will create the basic layout for a codeunit object when using the AL Language extension in Visual Studio Code.

Codeunit example

This codeunit example checks whether a given customer has registered a shoe size. If not, the customer is assigned a shoe size of 42.

```
codeunit 50113 CreateCustomer
{
    trigger OnRun();
    var
        r: record Customer;
    begin
        if not r.HasShoeSize() then
            r.ShoeSize := 42;
        end;
    }
}
```

See Also

[Developing Extensions](#)

[Table Extension Object](#)

[Page Extension Object](#)

[AL Development Environment](#)

Query Object

5/21/2019 • 2 minutes to read

A query describes a dataset of Dynamics 365 Business Central. You can query to retrieve fields from a single table or multiple tables. You can specify how to join tables in the query and filter the result data, and you can specify totaling methods on fields, such as sums and averages. Queries retrieve records from one or more tables and combine the records into rows and columns in a single dataset. You create a query by adding a Query object file to your project. In the Query object, you define dataitem and column elements in the elements section. The dataitem element specifies the table to retrieve records from. The column element specifies a field of the table to include in the resulting dataset of a query.

When you have specified the dataitem and column elements, you create links between the dataitem elements. A dataitem link determines which records to include in the dataset based on a common field between two dataitems.

For information about creating a query of the type API, see [API Query Type](#).

NOTE

Extension objects can have a name with a maximum length of 30 characters.

Snippet support

Typing the shortcut `tquery` will create the basic layout for a Query object when using the AL Language extension in Visual Studio Code.

Query syntax

```
query Id MyQuery
{
    QueryType = Normal;

    elements
    {
        dataitem(DataItemName; SourceTableName)
        {
            column(ColumnName; SourceFieldName)
            {
            }
            filter(FilterName; SourceFieldName)
            {
            }
        }
    }

    var
        myInt: Integer;

    trigger OnBeforeOpen()
    begin
    end;
}
```

Query example

The following example shows a query that displays a list of customers with sales and profit figures. The query primarily retrieves fields from the **Customer** table, but also displays fields from the **Salesperson Purchaser** and **Country Region** tables.

The query also uses the DataItemLink property to create a link between the **Customer** table, **Salesperson Code** field and the **Salesperson Purchaser** table, **Code** fields and a link between the **Customer** table, **Country/Region Code** field and the **Country/Region** table, **Code** field.

```
query 50102 "Top Customer Overview"
{
    Caption = 'Top Customer Overview';

    elements
    {
        dataitem(Customer; Customer)
        {
            column(Name; Name)
            {
            }
            column(No; "No.")
            {
            }
            column(Sales_LCY; "Sales (LCY)")
            {
            }
            column(Profit_LCY; "Profit (LCY)")
            {
            }
            column(Country_Region_Code; "Country/Region Code")
            {
            }
            column(City; City)
            {
            }
            column(Global_Dimension_1_Code; "Global Dimension 1 Code")
            {
            }
            column(Global_Dimension_2_Code; "Global Dimension 2 Code")
            {
            }
            column(Salesperson_Code; "Salesperson Code")
            {
            }
            dataitem(Salesperson_Purchaser; "Salesperson/Purchaser")
            {
                DataItemLink = Code = Customer."Salesperson Code";
                column(SalesPersonName; Name)
                {
                }
                dataitem(Country_Region; "Country/Region")
                {
                    DataItemLink = Code = Customer."Country/Region Code";
                    column(CountryRegionName; Name)
                    {
                    }
                }
            }
        }
    }
}
```

See Also

[Developing Extensions](#)

[AL Development Environment](#)

[API Query Type](#)

[Page Extension Object](#)

[Report Object](#)

[Page Properties](#)

XMLport Object

3/31/2019 • 2 minutes to read

XMLports are used to export and import data between an external source and Dynamics 365 Business Central. Sharing data between different computer systems is seamless when it is shared in XML format. Working with XML files can be tedious so the details of how the XML file is handled are encapsulated in XMLports.

To use an XMLport to import or export data, you first create an XMLport object. You can run the XMLport from a page or codeunit object.

NOTE

In the Dynamics 365 Business Central Web client, Request pages are not supported. Request pages are dialog boxes that enables the user to set a filter on the data, sort the data, or choose whether to export or import the data. If you try to run an XMLport with a Request page from the Web client, you receive an error that the XMLport page type is not supported.

Snippet support

Typing the shortcut `txmlport` will create the basic layout for an XMLport object when using the AL Language extension in Visual Studio Code.

XMLport syntax

```

xmlport Id MyXmlport
{
    schema
    {
        textelement(NodeName1)
        {
            tableelement(NodeName2; SourceTableName)
            {
                fieldattribute(NodeName3; NodeName2.SourceFieldName)
                {
                }
            }
        }
    }

    requestpage
    {
        layout
        {
            area(content)
            {
                group(GroupName)
                {
                    field(Name; SourceExpression)
                    {
                    }
                }
            }
        }

        actions
        {
            area(processing)
            {
                action(ActionName)
                {
                }
            }
        }
    }

    var
        myInt: Integer;
}

```

XMLport example

The following example shows a page extension of the **Permission Sets** page that adds an action to the specified page calling the XMLport **ExportPermissionSet**. The XMLport exports the permission set data to an XML file.

```

pageextension 50111 PermissionSetExporter extends "Permission Sets"
{
    actions
    {
        addafter(Permissions)
        {
            action(ExportPermissionSet)
            {
                Promoted = true;
                PromotedCategory = New;
                trigger OnAction();
                begin

```

```

begin
    Xmlport.Run(70000124, false, false);
end;
}
}
}

xmlport 50112 ExportPermissionSet
{
    Format = xml;

    schema
    {
        textelement(PermissionSets)
        {
            tableElement(PSet; "Aggregate Permission Set")
            {
                SourceTableView = WHERE ("App Name" = FILTER (<> ''));
                XmlName = 'PermissionSet';
                fieldattribute(RoleID; pset."Role ID") { }
                fieldattribute(RoleName; pset.Name) { }
                tableelement(P; "Tenant Permission")
                {
                    XmlName = 'Permission';
                    LinkTable = pset;
                    LinkFields = "Role ID" = FIELD ("Role ID");

                    textelement(ObjectType)
                    {
                        trigger onbeforePassvariable();
                        var
                            int: Integer;
                        begin
                            int := p."Object Type";
                            ObjectType := format(int);
                        end;
                    }
                    textelement(ObjectID)
                    {
                        trigger onbeforePassvariable();
                        var
                            int: Integer;
                        begin
                            int := p."Object ID";
                            ObjectID := format(int);
                        end;
                    }
                    textelement(ReadPermission)
                    {
                        trigger onbeforePassvariable();
                        var
                            int: Integer;
                        begin
                            int := p."Read Permission";
                            ReadPermission := format(int);
                        end;
                    }
                    textelement(InsertPermission)
                    {
                        trigger onbeforePassvariable();
                        var
                            int: Integer;
                        begin
                            int := p."Insert Permission";
                            InsertPermission := format(int);
                        end;
                    }
                    textelement(ModifyPermission)
                    {

```


Control Add-In Object

3/31/2019 • 4 minutes to read

The control add-in object allows you to add custom functionality to Dynamics 365 Business Central. A control add-in is a custom control, or visual element, for displaying and modifying data within an iframe or a page. For example, a control add-in can display the content of a webpage, visualize data as a chart or on a map, or host a custom web application. Control add-ins can exchange data with the Dynamics 365 server on various data types and can respond to user interaction to raise events that execute additional AL code.

Control add-in properties

In the control add-in definition, you must set the `Scripts` property to include scripts in the control add-in. The scripts could be local files in the package or references to external files using the HTTP or the HTTPS protocol. With the `StartupScript` property, you can call a special script that runs when the page you have implemented the control add-in on, is loaded. These settings initialize the control add-in. With the `Images` and `StyleSheet` properties, you can specify additional styling to the control add-in. For more information about some of the control add-in properties, see:

- [Images](#)
- [Scripts](#)
- [StartupScript](#)
- [StyleSheets](#)
- [RecreateScript](#)
- [RefreshScript](#)

Sizing of the control add-in

Control add-ins can either have fixed dimensions or dynamically adapt to the available space on the screen. By controlling the sizing of an add-in, you can ensure that the add-in and the surrounding content on the page remain optimal on smaller display targets such as the phone or when users resize the browser. The following properties are available for you to specify how the sizing of the control add-in should behave.

- [HorizontalShrink](#)
- [HorizontalStretch](#)
- [MinimumHeight](#)
- [MinimumWidth](#)
- [MaximumHeight](#)
- [MaximumWidth](#)
- [RequestedHeight](#)
- [RequestedWidth](#)
- [VerticalShrink](#)
- [VerticalStretch](#)

Control add-in considerations

Designing control add-ins that provide the best possible experience can require some additional planning, design, and implementation. The following considerations will help you design add-ins that look and feel seamlessly integrated with Dynamics 365 Business Central.

- Respond to touch events so that mobile users or those on devices supporting touch input can also use the add-in.
- Design content that is responsive and is able to flow, resize, or reorganize naturally based on the available space.
- Consider the accessibility needs of users, for example by implementing keyboard access and support for screen readers.
- Use the Style guidelines to apply a choice of colors, typefaces, and font sizes that match that of Dynamics 365 Business Central. For more information, see [Control Add-in Style Guide](#).
- Provide language translation and other localizations that match the current user language in Dynamics 365 Business Central.

Control add-in syntax example

The following control add-in example syntax defines a chart that can show how customers are represented per country on a map. The control add-in is implemented as a `usercontrol1` on a page called **CustomersMapPage**.

```

// The controladdin type declares the new add-in.

controladdin CustomersPerCountryChart
{
    // The Scripts property can reference both external and local scripts.
    Scripts = 'https://code.jquery.com/jquery-2.1.0.min.js',
              'js/main.js';

    // The StartupScript is a special script that the webclient calls once the page is loaded.
    StartupScript = 'js/chart.js';

    // Images and StyleSheets can be referenced in a similar fashion.

    // The layout properties define how control add-in are displayed on the page.
    VerticalShrink = true;

    // The procedure declarations specify what JavaScript methods could be called from AL.
    // In JavaScript code, there should be a global function LoadData(data) {}
    procedure LoadData(Data : JsonArray);

    // The event declarations specify what callbacks could be raised from JavaScript by using the webclient
    API:
    // Microsoft.Dynamics.NAV.InvokeExtensibilityMethod('CountryClicked', [{country: 'M'}])
    event CountryClicked(Country: JsonObject);
}

page 50100 CustomersMapPage
{
    layout
    {
        area(Content)
        {
            // The control add-in can be placed on the page using usercontrol keyword.

            usercontrol(ControlName; CustomersPerCountryChart)
            {
                // The control add-in events can be handled by defining a trigger with a corresponding name.

                trigger CountryClicked(Country : JsonObject)
                var Data : JsonArray;
                begin

                    // The control add-in methods can be invoked via a reference to the usercontrol.

                    CurrPage.ControlName.LoadData(Data);
                end;
            }
        }
    }
}

```

See Also

[AL Development Environment](#)
[Developing Extensions](#)
[Asynchronous Considerations for Control Add-ins](#)
[InvokeExtensibility Method](#)
[GetImageResource Method](#)
[GetEnvironment Method](#)
[Pages Overview](#)
[Page Extension Object](#)
[Page Customization Object](#)

Data Types and Methods in AL

6/25/2019 • 9 minutes to read

The following data types are available as part of the AL Language. Each data type has various methods that support it. For more information about a data type and its methods, select a link in the table.

TYPE	DESCRIPTION
BigInteger	Stores very large whole numbers that range from -9,223,372,036,854,775,807 to 9,223,372,036,854,775,807.
BigText	Handles large text documents.
Blob	Is a complex data type. Variables of this data type differ from normal numeric and string variables in that BLOBs have a variable length. The maximum size of a BLOB(binary large object) is 2 GB.
Boolean	Indicates true or false.
Byte	Stores a single, 8-bit character as a value in the range 0 to 255. You can easily convert this data type from a number to a character and vice versa. This means you can use mathematical operators on Byte variables.
Char	Stores a single, 16-bit character as a value in the range 0 to 65535. You can convert this data type from a number to a character and vice versa. This means you can use mathematical operators on Char variables.
Code	Denotes a special type of string that is converted to uppercase and removes any trailing or leading spaces.
Codeunit	Is a container for AL code that you can use from other application objects.
CodeunitInstance	Is a container for AL code that you can use from other application objects.
CompanyProperty	Provides language support for company properties.
Database	Provides access to common database functionality.
Date	Denotes a date ranging from January 1, 1753 to December 31, 9999.
DateFormula	Represents a date formula that has the same capabilities as an ordinary input string for the CALCDATE Method (Date). The DateFormula data type is used to provide multilanguage capabilities to the CALCDATE Method (Date).

TYPE	DESCRIPTION
DateTime	Denotes a date and time ranging from January 1, 1753, 00:00:00.000 to December 31, 9999, 23:59:59.999. An undefined or blank DateTime is specified by ODT.
Debugger	Enables communication with a debugger.
Decimal	Denotes decimal numbers ranging from -999,999,999,999,999.99 to +999,999,999,999,999.99.
Dialog	Represents a dialog window.
Dictionary	Represents a collection of keys and values.
DotNet	Represents an unspecified .NET type.
Duration	Represents the difference between two DateTimes. This value can be negative. It is stored as a 64-bit integer. The integer value is the number of milliseconds during the duration.
ErrorInfo	Provides a structure for grouping information about an error.
FieldRef	Identifies a field in a table and gives you access to this field.
File	Represents a file.
FilterPageBuilder	Stores filter configurations for a filter page. A filter page is a dynamic page type that contains one or more filter controls that enables users to set filters on fields of the underlying tables.
Guid	Represents a 16 byte binary data type. This data type is used for the global identification of objects, programs, records, and so on. The important property of a GUID is that each value is globally unique. The value is generated by an algorithm, developed by Microsoft, which assures this uniqueness.
HttpClient	Provides a data type for sending HTTP requests and receiving HTTP responses from a resource identified by a URI.
HttpContent	Represents an HTTP entity body and content headers.
HttpHeaders	Is a collection of headers and their values.
HttpRequestMessage	Represents an HTTP request message.
HttpResponseMessage	Represents a HTTP response message including the status code and data.
InStream	Is a generic stream object that you can use to read from or write to files and BLOBs. You can define the internal structure of a stream as a flat stream of bytes. You can assign one stream to another. Reading from and writing to a stream occurs sequentially.

TYPE	DESCRIPTION
Integer	Stores whole numbers with values that range from - 2,147,483,647 to 2,147,483,647.
IsolatedStorage	Provides data isolation for extensions.
Any	This data type can be substituted by any other data type.
JsonArray	Is a container for any well-formed JSON array. A default JsonArray contains an empty JSON array.
JsonObject	Is a container for any well-formed JSON object. A default JsonObject contains an empty JSON object.
JsonToken	Is a container for any well-formed JSON data. A default JsonToken object contains the JSON value of NULL.
JsonValue	Is a container for any well-formed fundamental JSON value. A default JsonValue is set to the JSON value of NULL.
KeyRef	Identifies a key in a table and the fields in this key.
Label	Denotes a string constant that can be optionally translated into multiple languages.
List	Represents a strongly typed list of objects that can be accessed by index.
Media	Encapsulates media files, such as image .jpg and .png files, in application database tables. The Media data type can be used as a table field data type, but cannot be used as a variable or parameter. The Media data type enables you to import a media file to the application database and reference the file from records, making it possible to display the media file in the client user interface. You can also export media from the database to files and streams.
MediaSet	Encapsulates media, such as images, in application database tables.
ModuleDependencyInfo	Provides information about a dependent module.
ModuleInfo	Represents information about an application consumable from AL.
NavApp	Provides information about a NavApp.
None	Is used implicitly when a method does not return a value.
Notification	Provides a programmatic way to send non-intrusive information to the user interface (UI) in the Business Central Web client.
NumberSequence	Is a complex data type for creating and managing number sequences in the database.

TYPE	DESCRIPTION
Option	Denotes an option value. In the code snippet below, you can see how the Option data type is declared.
OutStream	Is a generic stream object that you can use to write to files and BLOBs.
Page	Contains a number of simpler elements called controls. Controls are used to display information to the user or receive information from the user.
ProductName	An application can have a full name, marketing name, and short name. The PRODUCTNAME functions enable you to retrieve these name variations.
Query	Enables you to retrieve data from multiple tables and combine the data in single dataset.
QueryInstance	Enables you to retrieve data from multiple tables and combine the data in single dataset.
RecordId	Contains the table number and the primary key of a table.
RecordRef	References a record in a table.
Report	Is used to display, print, or process information from a database.
ReportInstance	Reports are used to display, print, or process information from a database.
RequestPage	Is a page that is run before the report starts to execute. Request pages enable end-users to specify options and filters for a report.
Session	Represents a Microsoft Dynamics Business Central session.
SessionSettings	Is a complex data type for passing user personalization settings for a client session as an object. The object contains properties that correspond to the fields in the system table 2000000073 User Personalization , including: App ID, Company, Language ID, Locale ID, Profile ID, Scope, and Time Zone. You can use the AL methods of the SessionSettings data type to get, set, and send the user personalization settings for the current client session.
String	Denotes a sequence of characters. It can be represented by a string literal, a text value or a code value.
System	Is a complex data type.
Record	Is a complex data type.
TaskScheduler	Is a complex data type for creating and managing tasks in the task scheduler, which runs codeunits at scheduled times.

TYPE	DESCRIPTION
TestAction	Represents a test action on a page.
TestField	Represents a testable field on a page.
TestFilter	Represents a test filter on a page.
TestFilterField	Represents the type of a field filter in a test filter on a page or on a request page.
TestPage	Represents a variable type that can be used to test Page Application Objects.
TestPart	Represents a variable type that can be used to test Page Application Objects of type Part.
TestRequestPage	Stores test request pages. A test request page part is a logical representation of a request page on a report. A test request page does not display a user interface (UI). The subtype of a test request page is the report whose request page you want to test.
Text	Denotes a text string.
TextConst	Denotes a multi-language string constant.
TextBuilder	Represents a lightweight wrapper for the .Net implementation of StringBuilder.
Time	Denotes a time ranging from 00:00:00.000 to 23:59:59.999. An undefined or blank time is specified by OT.
Variant	Represents an AL variable object. The AL variant data type can contain many AL data types.
Version	Represents a version matching the format: Major.Minor.Build.Revision .
WebServiceActionContext	Represents an AL WebServiceActionContext.
XmlAttribute	Represents an XML attribute.
XmlAttributeCollection	Represents a collection of XML attributes.
XmlCDATA	Represents a CDATA section.
XmlComment	Represents an XML comment.
XmlDeclaration	Represents an XML declaration.
XmlDocument	Represents an XML document.
XmlDocumentType	Represents an XML document type.

TYPE	DESCRIPTION
XmlElement	Represents an XML element.
XmlNamespaceManager	Represents a namespace manager that can be used to resolve, add and remove namespaces to a collection. It also provides scope management for these namespaces.
XmlNameTable	Represents a table of atomized string objects.
XmlNode	Represents a XML node which can either be for instance an XML attribute, an XML element or a XML document.
XmlNodeList	Represents a collection of XML nodes.
Xmlport	XmlPorts are used to export or import data between an external source and a Microsoft Dynamics Business Central database.
XmlportInstance	Represents an instance of an XmlPort.
XmlProcessingInstruction	Represents a processing instruction, which XML defines to keep processor-specific information in the text of the document.
XmlReadOptions	Represents the options configuring how XML is loaded from a data source.
XmlText	Represents the text content of an element or attribute.
XmlWriteOptions	Represents the options configuring how XML is saved.
Action	Represents the action that the user took on the page.
ClientType	Represents the type of the client executing the operation.
DataClassification	Sets the classification of the data in the table or field.
DataScope	Identifies the scope of stored data in the isolated storage.
DefaultLayout	The default layout to be used by a report.
ErrorType	Represents the type of error.
ExecutionContext	Represents the context in which a session is running. In certain scenarios, for example during upgrade, the system will run a session in a special context for a limited time.
ExecutionMode	The execution mode of the current session.
FieldClass	Represents the type of a field class.
FieldType	Represents the type of a table field.

TYPE	DESCRIPTION
NotificationScope	Specifies the context in which the notification appears in the client.
ObjectType	The different types of objects.
PageBackgroundTaskErrorLevel	Specifies how an error in the page background task appears in the client.
ReportFormat	Specifies the format of the report.
SecurityFilter	Specifies how security filters are applied to the record.
TableConnectionType	Use variables of this data type to specify the type of connection to an external database.
TestPermissions	Specifies a value that can be used to determine which permission sets are used on tests that are run by test codunits or test functions.
TextEncoding	Represents a file encoding.
TransactionModel	Represents a test transaction model.
TransactionType	Represents a transaction type.
Verbosity	Represents the security level of events.
WebServiceActionResultCode	Represents a web service action status code.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

Array Methods

3/31/2019 • 2 minutes to read

An array is a data structure that contains a number of variables which are accessed through computed indices. The variables contained in an array, also called the elements of the array, are all of the same type, and this type is called the element type of the array.

An array has a rank which determines the number of indices associated with each array element. The rank of an array is also referred to as the dimensions of the array. An array with a rank of one is called a single-dimensional array. An array with a rank greater than one is called a multi-dimensional array. Specific sized multi-dimensional arrays are often referred to as two-dimensional arrays, three-dimensional arrays, and so on. Each dimension of an array has an associated length which is an integral number greater than or equal to zero. The maximum number of dimensions is 10 and the total number of elements in all dimensions is 1,000,000.

The length of a dimension determines the valid range of indices for that dimension. For a dimension of length N, indices can range from **1 to N** inclusive. The total number of elements in an array is the product of the lengths of each dimension in the array. If one or more of the dimensions of an array have a length of zero, the array is considered to be empty.

Syntax

The syntax for declaring an array of a specific type is the following:

```
Array [Dimension] of Type;
```

The `Dimension` is a comma-delimited list of integer literals greater than 0, where each integer defines the number of elements in that dimension.

The `Type` is the element type of the array.

Code Example

The following code sample shows the declaration of an array with a simple element type.

```
ArrayOfInteger: Array [10] of Integer;
```

The following code sample shows the declaration of an array with an element type of a fixed length.

```
ArrayOfCode: Array [10] of Code[20];  
ArrayOfText: Array [10] of Text[20];
```

The following code sample shows the declaration of an array with a complex element type.

```
ArrayOfCodeunits: Array [10] of Codeunit 10;  
ArrayOfQueryes: Array [10] of Query "My Query";  
ArrayOfTemporaryRecords: Array [10] of Record 10 Temporary;  
ArrayOfDotNetVariables: Array [10] of DotNet String;
```

Methods

The following AL methods for arrays are available:

[ARRAYLEN Method](#)

[COMPRESSARRAY Method](#)

[COPYARRAY Method](#)

See Also

[AL Method Reference](#)

Method Attributes

4/10/2019 • 2 minutes to read

An attribute is modifier on a method declaration that specifies information that controls the method's use and behavior. For example, decorating a method with the Integration attribute sets the method to be an event publisher. An attribute can have one or more arguments that set properties for the method instance.

In AL, attributes are placed before the method, and have the following syntax:

```
[Attribute_Name(ArgumentName : data_type, ArgumentName : data_type)]
```

For example, the Integration attribute has two arguments, and the syntax is:

```
[Integration(IncludeSender : Boolean, GlobalVarAccess : Boolean)]
```

Attributes

The following method attributes are available:

- [Integration Attribute](#)
- [Business Attribute](#)
- [EventSubscriber Attribute](#)
- [NonDebuggable Attribute](#)

See Also

[AL Method Reference](#)

Procedure overload

4/29/2019 • 2 minutes to read

Procedure overload enables developers to create multiple procedures with the same name, but with different signatures, on the same application object. Conceptually, overloaded procedures are used to execute the same task on a different set of arguments. When an overloaded procedure is called, a specific implementation of that procedure, appropriate to the context of the call, will be run.

Reasons for using procedure overload

Overloaded procedures give programmers the flexibility to call a procedure with similar semantics for different types of data. At the same time, overloaded procedures remove the need for abusing the Variant data type for the purpose of processing different types of data a similar manner and allows the developer to write strongly-typed code and rely on the compiler for validation.

Remarks

Overload resolution is performed by using procedure signatures to find the best match. The signature of a procedure is represented by its name and the type, order, and number of parameters. The return type of a procedure is not part of the procedure's signature.

Example

The following example shows how a **ToString** method can be implemented with and without using procedure overloads.

In the first code snippet, a **ToString** procedure is implemented. This takes a Variant value and inspects the type of the value to delegate to different implementations. If the caller passes a value of a different type than Integer, Date and Text, an empty string will be returned. This can lead to bugs that will only show up at runtime.

```

codeunit 10 Stringifier
{
    local procedure TextToString(value : Text) : Text;
    begin
        Exit(value);
    end;

    local procedure DateToString(value : Date) : Text;
    begin
        Exit(Format(value));
    end;

    local procedure IntegerToString(value : Integer) : Text;
    begin
        Exit(Format(value));
    end;

    procedure ToString(value: Variant) : Text;
    begin
        if value.IsInteger then
            Exit(IntegerToString(value))
        else if value.IsDate then
            Exit(DateToString(value))
        else if value.IsText then
            Exit(TextToString(value))
        else
            Exit('');
    end;
}

```

In the second code snippet, we overload the ToString procedure for Text, Date and Integer. At this point, it is not possible for a caller to call a ToString method with a different type other than Integer, Date, or Text. This will catch the bug above at compile time.

```

codeunit 10 StringifierWithOverloads
{
    procedure ToString(value : Text) : Text;
    begin
        Exit(value);
    end;

    procedure ToString(value : Date) : Text;
    begin
        Exit(Format(value));
    end;

    procedure ToString(value : Integer) : Text;
    begin
        Exit(Format(value));
    end;
}

```

See Also

[AL Development Environment](#)

Action Option Type

3/31/2019 • 2 minutes to read

Represents the action that the user took on the page.

Members

MEMBER	DESCRIPTION
None	Represents the result of running a page.
OK	Represents the result of the user closing a page window by performing one of the following actions: <ul style="list-style-type: none">- Chooses the OK button.- Chooses the X button when there was no Cancel button on the window.- Presses the Esc key when there is no Cancel button on the window.
Cancel	Represents the result of the user closing a page window by performing one of the following actions: <ul style="list-style-type: none">- Chooses the Cancel button.- Chooses the X button when there is a Cancel button on the window.- Presses the Esc key when there is a Cancel button on the window
LookupOK	Represents the result of the user closing a lookup window by performing one of the following actions: <ul style="list-style-type: none">- Chooses the OK button.- Chooses an item in the Lookup window.
LookupCancel	Represents the result of the user closing a lookup window by choosing the Cancel button.
Yes	Represents the result of the user closing a confirmation window by choosing the Yes button.
No	Represents the result of the user closing a confirmation window by performing one of the following actions: <ul style="list-style-type: none">- Chooses the No button.- Chooses the X button.- Presses the Esc key.
RunObject	Represents the result of the user selecting an option that ran another object.
RunSystem	Represents the result of the user selecting an option that ran an external program.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Any Data Type

3/31/2019 • 2 minutes to read

This data type can be substituted by any other data type.

NOTE

The Any Data type cannot be used for declaring constructs in AL. Any is a type that is used for the parameters or return type of methods in the platform.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

BigInteger Data Type

3/31/2019 • 2 minutes to read

Stores very large whole numbers that range from -9,223,372,036,854,775,807 to 9,223,372,036,854,775,807.

Remarks

This data type is a 64-bit integer.

You must add an L to the constant definition to inform AL that the integer must be interpreted and treated as a BigInteger.

If you assign -9,223,372,036,854,775,808 directly to a BigInteger variable, then you get an error when you try to compile the code. However, you can indirectly assign -9,223,372,036,854,775,808 to a BigInteger variable by using the following code.

```
BigIntegerVar := -9223372036854775807L;  
BigIntegerVar := BigIntegerVar - 1;
```

If you try to indirectly assign a value that is smaller than -9,223,372,036,854,775,808, or larger than 9,223,372,036,854,775,807, then you get a run-time error.

Example

```
BI := 1L;  
BI := 455500000000L;
```

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

BigText Data Type

3/31/2019 • 2 minutes to read

Handles large text documents.

The following methods are available on instances of the BigText data type.

METHOD NAME	DESCRIPTION
Length()	Retrieves the length of the text stored in this BigText instance.
AddText(String, [Integer])	Adds a text string to a BigText variable.
AddText(BigText, [Integer])	Adds a text string to a BigText variable.
GetSubText(var Text, Integer, [Integer])	Gets part of a BigText variable.
GetSubText(var BigText, Integer, [Integer])	Gets part of a BigText variable.
TextPos(String)	Gets the position at which a specific string first occurs in this BigText instance.
Write(OutputStream)	Streams a BigText object to a BLOB field in a table.
Read(InputStream)	Streams a BigText object that is stored as a BLOB in a table to a BigText variable.

Remarks

This data type cannot be shown in a message window or be seen in the Debugger. The maximum length of a BigText variable is 2,147,483,647 characters and this corresponds to 2 GB. You can use the BigText methods to manipulate a BigText variable, for example to extract part of a BigText variable or to add a text string to a BigText variable. The normal string methods cannot be used with a BigText variable.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Blob Data Type

3/31/2019 • 2 minutes to read

Is a complex data type. Variables of this data type differ from normal numeric and string variables in that BLOBs have a variable length. The maximum size of a BLOB(binary large object) is 2 GB.

The following methods are available on instances of the Blob data type.

METHOD NAME	DESCRIPTION
HasValue()	Determines whether a binary large object (BLOB) has a value.
Length()	Returns the number of bytes in the binary large object (BLOB).
Import(String)	Imports a binary large object (BLOB) from a file.
Export(String)	Exports a binary large object (BLOB) to a file.
CreateInStream(InStream, [TextEncoding])	Creates an InStream object for a binary large object (BLOB). This enables you to read data from the BLOB.
CreateOutStream(OutStream, [TextEncoding])	Creates an OutStream object for a binary large object (BLOB). This enables you to write data to the BLOB.

Remarks

Use BLOBs to store memos (text), pictures (bitmaps), or user-defined types.

NOTE

You cannot view text that is stored in BLOBs from the development environment.

You can read from and write to BLOBs by creating input and output streams, respectively. To do so, use [CREATEINSTREAM method \(BLOB\)](#) and [CREATEOUTSTREAM method \(BLOB\)](#).

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Boolean Data Type

3/31/2019 • 2 minutes to read

Indicates true or false.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Byte Data Type

3/31/2019 • 2 minutes to read

Stores a single, 8-bit character as a value in the range 0 to 255. You can easily convert this data type from a number to a character and vice versa. This means you can use mathematical operators on Byte variables.

Example

The following example assumes that you have a Byte variable named B and a Text variable named S.

You can assign a constant string of the length 1 to a Byte variable, as shown in the first line of the following code example. You can assign a single character in a Text or Code variable to a Byte variable, as shown in the second line of the following code example. You can assign a numeric value to a Byte variable, as shown in the third line of the following code example. This causes the Byte variable to contain the character from the ASCII character set that corresponds to the numeric ASCII code.

```
B := 'A';  
B := S[2];  
B := 65;
```

You cannot assign a character to a position greater than the position of the null terminator. For example, if the value of the text variable *MyText* is 'abc', then the null terminator is at position 4 and the following assignment causes a run-time error to occur.

```
MyText[5] := 'e';
```

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Char Data Type

3/31/2019 • 2 minutes to read

Stores a single, 16-bit character as a value in the range 0 to 65535. You can convert this data type from a number to a character and vice versa. This means you can use mathematical operators on Char variables.

Example

The following example assumes that you have a Char variable named C and a Text or Code variable named S.

You can assign a constant string of the length 1 to a Char variable, as shown in the first line of the following code example. You can assign a single Char in a Text or Code variable to a Char variable, as shown in the second line of the following code example. You can assign a numeric value to a Char variable, as shown in the third line of the following code example.

```
C := 'A';  
C := S[2];  
C := 65;
```

You cannot assign a Char to a position greater than the position of the null terminator. For example, if the value of the Text variable *MyText* is 'abc', then the null terminator is at position 4 and the following assignment causes a run-time error to occur.

```
MyText[5] := 'e';
```

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

ClientType Option Type

3/31/2019 • 2 minutes to read

Represents the type of the client executing the operation.

Members

MEMBER	DESCRIPTION
Background	A background client.
Desktop	A desktop client.
Management	A management client.
NAS	A NAS client.
OData	A NAS client.
Phone	Microsoft Dynamics Business Central Phone client.
SOAP	A SOAP client.
Tablet	Microsoft Dynamics Business Central Tablet client.
Web	Microsoft Dynamics Business Central Web client.
Windows	Microsoft Dynamics Business Central Windows client.
Current	Microsoft Dynamics Business Central Windows client.
Default	The default client.
ODataV4	A ODataV4 client.
Api	An API client.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Code Data Type

3/31/2019 • 2 minutes to read

Denotes a special type of string that is converted to uppercase and removes any trailing or leading spaces.

Remarks

The length of a Code variable equals the number of characters in the text without leading or trailing spaces.

You must specify the length of a Code variable or field. The maximum length of a Code variable is 1024 characters. The maximum length of a Code field in a table is 250 characters. A Code variable cannot be null. The Code data type supports Unicode.

You can index any character position in a string, such as A[65]. The resulting value will be a [Char Data Type](#). You cannot assign a char to a position in the code variable greater than the current length of the variable + 1.

Fields that contain a date formula must not have data type Code. Instead, use the [DateFormula Data Type](#). All fields that contain a date formula with data type Code must be converted into data type DateFormula.

Example

This example shows some typical examples of code string assignments. In these examples, assume that the variable c is a code variable with a maximum length of 4.

```
c := 'ABC';  
// Results in variable c, which contains 'ABC'  
// and is 3 characters in length.  
c := '1';  
// Results in variable c, which contains '1'  
// and is 1 character in length.  
c := '';  
// Results in variable c, which contains '' (empty string)  
// and is zero (0) characters in length.  
c := ' 2 ';  
// Results in variable c, which contains '2'  
// and is 1 character in length.
```

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Codeunit Data Type

3/31/2019 • 2 minutes to read

Is a container for AL code that you can use from other application objects.

The following methods are available on the Codeunit data type.

METHOD NAME	DESCRIPTION
Run(Integer, [var Record])	Loads and runs the unit of AL code you specify. To use this method, you can specify a table associated with the codeunit when you defined the codeunit properties. This allows you to pass a variable with the method. The transaction that the codeunit contains is always committed due to the Boolean return value.

The following methods are available on instances of the Codeunit data type.

METHOD NAME	DESCRIPTION
Run(var Record)	Loads and executes the unit of C/AL code that you specify.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

CodeunitInstance Data Type

3/31/2019 • 2 minutes to read

A container for AL code that you can use from other application objects.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

CompanyProperty Data Type

3/31/2019 • 2 minutes to read

Provides language support for company properties.

The following methods are available on the CompanyProperty data type.

METHOD NAME	DESCRIPTION
DisplayName()	Gets the current company display name.
UrlName()	Gets the string that represents the company name in a URL.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Database Data Type

3/31/2019 • 2 minutes to read

Provides access to common database functionality.

The following methods are available on the Database data type.

METHOD NAME	DESCRIPTION
UserId()	Gets the user name of the user account that is logged on to the current session.
CompanyName()	Gets the current company name.
Commit()	Ends the current write transaction.
SerialNumber()	Gets a string that contains the serial number of the license file for your system.
CheckLicenseFile(Integer)	Checks a key in the license file of the system.
SelectLatestVersion()	Forces the latest version of the database to be used.
CurrentTransactionType([TransactionType])	Gets the current transaction type and sets a new type to be assigned.
LockTimeout([Boolean])	Determines whether the lock time-out setting is set to On. You can also use this method to override the default setting.
SID([String])	Retrieves the security identifier (SID) of a Windows user account.
UserSecurityId()	Gets the unique identifier of the user that is logged on to the current session.
SetUserPassword(Guid, String)	Sets a password for the user with the given user security ID. If the given password is blank, an empty string will be stored instead of a password hash. This will prevent the user from logging in using a password. Only SUPER can call this method. Passwords cannot be set for the empty GUID or for the default Super ID.
ChangeUserPassword(String, String)	Changes the password for the current user.

METHOD NAME	DESCRIPTION
TenantId()	Gets the ID of the tenant that has started the current session. Use this method when your code must be specific about which tenant database to access in a multitenant deployment. For example, if your code imports data into a cache, you can make a cache tenant-specific by using the tenant ID as a key. Also, if you want to write code that saves documents, you can include the tenant ID in the file name or location, for example. In those cases, you can use the <code>TENANTID</code> method in combination with the <code>COMPANYNAME</code> method to identify the company and the tenant database.
SessionId()	Gets the ID of the current session.
ServiceInstanceId()	Gets the ID of the service instance.
CopyCompany(String, String)	Creates a new company and copies all data from an existing company in the same database.
ImportData(Boolean, var Text, [Boolean], [Boolean], [Record])	Imports data from a file that has been exported from a database.
ExportData(Boolean, var Text, [String], [Boolean], [Boolean], [Boolean], [Record])	Exports data from the database to a file. The data is not deleted from the database.
DataFileInfo(Boolean, var Text, var Text, var Boolean, var Boolean, var Boolean, var Text, var DateTime, var Record)	Specifies data from a file that has been exported from a database.
RegisterTableConnection(TableConnectionType, String, String)	Registers a table connection to an external database.
UnregisterTableConnection(TableConnectionType, String)	Unregisters a table connection to an external database.
SetDefaultTableConnection(TableConnectionType, String, [Boolean])	Establishes a connection to an external database based on a previously registered connection of the specified type.
GetDefaultTableConnection(TableConnectionType)	Gets the default table connection based on the specified connection type. You must already have registered a table connection of this type.
HasTableConnection(TableConnectionType, String)	Verifies if a connection to an external database exists based on the specified name.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

DataClassification Option Type

3/31/2019 • 2 minutes to read

Sets the classification of the data in the table or field.

Members

MEMBER	DESCRIPTION
CustomerContent	Content directly provided/created by admins and users.
ToBeClassified	Content that has not yet been given a classification. This is the initial value when table or field is created.
EndUserIdentifiableInformation	(EUII) Data that identifies or could be used to identify the user of a Microsoft service. EUII does not contain Customer content.
AccountData	Customer billing information and payment instrument information, including administrator contact information, such as tenant administrator's name, address, or phone number.
EndUserPseudonymousIdentifiers	(EUPI) An identifier created by Microsoft tied to the user of a Microsoft service. When EUPI is combined with other information, such as a mapping table, it identifies the end user. EUPI does not contain information uploaded or created by the customer (Customer content or EUII).
OrganizationIdentifiableInformation	(OII) Data that can be used to identify a tenant, generally config or usage data. This data is not linkable to a user and does not contain Customer content.
SystemMetadata	Data generated while running the service or program that is not linkable to a user or tenant.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

DataScope Option Type

3/31/2019 • 2 minutes to read

Identifies the scope of stored data in the isolated storage.

Members

MEMBER	DESCRIPTION
Module	Indicates that the record is available in the scope of the app(extension) context.
Company	Indicates that the record is available in the scope of the company within the app context.
User	Indicates that the record is available for a user within the app context.
CompanyAndUser	Indicates that the record is available for a user and specific company within the app context.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Date Data Type

3/31/2019 • 2 minutes to read

Denotes a date ranging from January 1, 1753 to December 31, 9999.

The displayed text format of the date is determined by your Region and Language Format setting in Windows.

Undefined dates

An undefined or blank date is specified by 0D. The undefined date is considered to be before all other dates.

Normal dates and closing dates

All normal dates have a corresponding closing date. The closing date for a given date is defined as a period of time that follows a given normal date and precedes the next normal date.

Syntax

The syntax for defining DateTime format follows the [ISO standard](#).

- The syntax for defining Date format is `yyyymmddD`, where `D` is a mandatory letter. For example, `20180325D`, read as the 26th of March, 2018.

To assign a normal date to a variable, use the following format: `yyyymmddD`.

Storing dates in the SQL server database

SQL Server stores information about both date and time in columns of the DATETIME types. For date fields, Dynamics 365 Business Central uses only the date and uses a constant value for the time. For a normal date, this constant value contains 00:00:00:000. For a closing date, it contains 23:59:59:000.

The Dynamics 365 Business Central undefined date is represented by the earliest valid date in SQL Server. The earliest valid date in SQL Server for a DATETIME is 01-01-1753 00:00:00:000.

If you store a date in the database that is outside the valid range for a SQL DATETIME, a run-time error occurs.

Example

This example shows a valid assignment of date. It requires that you define the following variable.

VARIABLE	DATATYPE
Date1	Date

This example is compiled and run on a computer with the regional format set to English (United States).

```
Date1 := 20180612D;  
MESSAGE(FORMAT(Date1));
```

The message window displays the following:

06/12/2018

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

DateFormula Data Type

4/24/2019 • 2 minutes to read

Represents a date formula that has the same capabilities as an ordinary input string for the [CALCDATE Method \(Date\)](#). The DateFormula data type is used to provide multilanguage capabilities to the [CALCDATE Method \(Date\)](#).

Remarks

When a date calculation formula is stored in a DateFormula field, it is converted to a generic, non-language dependent format. When a date calculation formula is retrieved from a DateFormula field, it is converted to a valid date conversion string for the currently selected language.

To assign a value to a DateFormula data type, whether it is a field or a variable, you must use the [EVALUATE Method](#).

Example

This example requires that you create a DateFormulaVariable variable that is a DateFormula data type.

```
IF FORMAT(DateFormulaVariable) = ' ' THEN  
    EVALUATE(DateFormulaVariable, '1W');
```

You must use the [FORMAT Method](#) to make a comparison with a text string. If you do not use this method, then the IF statement will fail because you cannot compare a DateFormula data type with a Text data type.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

DateTime Data Type

3/31/2019 • 2 minutes to read

Denotes a date and time ranging from January 1, 1753, 00:00:00.000 to December 31, 9999, 23:59:59.999. An undefined or blank DateTime is specified by ODT.

The displayed text format of a DateTime is determined by your Regional and Language Options in Windows.

Remarks

A DateTime is stored in the database as Coordinated Universal Time (UTC). UTC is the international time standard (formerly Greenwich Mean Time, or GMT). Zero hours UTC is midnight at 0 degrees longitude.

The DateTime is always displayed as local time in Dynamics 365 Business Central. Local time is determined by the time zone regional settings used by your computer. You must always enter DateTimes as local time. When you enter a DateTime as local time, it is converted to UTC using the current settings for the time zone and daylight savings time.

The DateTime data type does not support closing dates.

By default, DateTimes are displayed using the standard display format. When you use the standard display format, seconds and milliseconds are not displayed until you select the DateTime field. Furthermore, if you export your data using an XMLport or by writing it to a file, the seconds and milliseconds are not exported unless you specify that DateTime fields use another format and display this information. For more information about how DateTime objects are displayed and the formats that are available, see [Format Property](#).

The only constant available when you use the DateTime data type is the undefined DateTime, ODT. To assign a constant value to a DateTime variable you must use the [CREATEDATETIME method](#).

If you use a date that is outside the valid date range, a run-time error occurs.

Syntax

The syntax for defining DateTime format follows the [ISO standard](#).

- The syntax for defining Date format is `yyyymmddD`, where `D` is a mandatory letter. For example, `20180325D`, read as the 26th of March, 2018.
- The syntax for defining Time format is `hhmmssT`, where `T` is the time designator. For example, `093125H`, read as 9:13:25.

SQL Server

In SQL Server, the earliest permitted DateTime is January 1, 1753, 00:00:00.000. The latest permitted DateTime is December 31, 9999, 23:59:59.999. If you store a date in the database that is outside the valid range for a SQL DATETIME, a run-time error run-time occurs.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

Debugger Data Type

3/31/2019 • 2 minutes to read

Enables communication with a debugger.

NOTE

This data type is supported only in Business Central on-premises.

The following methods are available on the Debugger data type.

METHOD NAME	DESCRIPTION
Activate()	Activates the debugger and attaches the debugger to the next session that is started.
Deactivate()	Deactivates the debugger.
IsActive()	Indicates whether the debugger is active.
Attach(Integer)	Activates the debugger and attaches it to the specified session.
IsAttached()	Specifies if the debugger is attached to a session.
DebuggedSessionID()	Gets the ID of the previous session that the debugger was attached to.
IsBreakpointHit()	Specifies if a breakpoint is hit in a debugging session.
Break()	Breaks code execution of a debugging session.
BreakOnError(Boolean)	Sets whether the debugger breaks on errors.
BreakOnRecordChanges(Boolean)	Breaks execution before a change to a record occurs.
SkipSystemTriggers(Boolean)	Enables the debugger to skip code that is inside system triggers.
Continue()	Executes code until the next breakpoint or until execution ends.
StepInto()	Executes a method call and then stops at the first line of code inside the method.
StepOut()	Enables debugging to return to the calling method after it steps into a method call.
StepOver()	Executes a method call and then stops at the first line outside the method call.

METHOD NAME	DESCRIPTION
Stop()	Stops execution as if the code hits an error.
DebuggingSessionID()	Gets the ID of the session that the debugger is currently attached to.
GetLastErrorText()	Gets the last error that occurred in the debugger.
EnableSqlTrace(Integer, [Boolean])	Enables or verifies SQL tracing. If you enable SQL tracing, then SQL Server events for selected sessions on the server instance are collected.

NOTE

The Dynamics 365 Business Central Debugger is an example of a debugger application that is built using tables, pages, codeunits, and the AL debugger methods.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Decimal Data Type

3/31/2019 • 2 minutes to read

Denotes decimal numbers ranging from -999,999,999,999.99 to +999,999,999,999.99.

Example

The following are examples of decimal values.

```
546.88
3425.57
```

Example

The following is not a decimal, but rather an [Integer Data Type](#).

```
342
```

Limits on Decimal Data Type Variables

The Decimal data type is mapped to the Microsoft .NET Framework common language runtime (CLR) Decimal data type, which controls and the precision and limits for variables.

The following table shows the limits for variables of type `Decimal`.

LIMIT	VALUE
Maximum format value. This is the maximum value that can be: <ul style="list-style-type: none">- Formatted into a TEXT variable by the FORMAT function.- Input from the UI or XMLPorts.- Assigned directly in source code.	+/- 999,999,999,999.99
Maximum field data type value. This is the maximum value that a field variable in a record can hold while not being persisted.	+/- 999,999,999,999.99
Maximum persisted value. This is the maximum value that can be stored in the database.	Can read previous stored values but cannot store values outside the formatting range since field variables cannot be assigned values outside the formatting range.
Maximum calculating value. This is the maximum value that can be calculated by code statements while not assigning to a field variable, storing to the database, or formatting to a text variable.	+/- 79,228,162,514,264,337,593,543,950,335

LIMIT	VALUE
Scaling factor (digits after decimal point) for calculating values	28 For example, 7.9228162514264337593543950335

It is possible to assign to a variable the maximum value that can be formatted and then multiply that variable by a large positive number, thereby generating a greater value. However, we do not recommend doing this. If you do, you will get errors if you attempt to format this variable to a text variable or assign the variable to a field variable in a record.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

DefaultLayout Option Type

3/31/2019 • 2 minutes to read

The default layout to be used by a report.

Members

MEMBER	DESCRIPTION
None	The default layout is not set.
RDLC	The default layout is RDLC.
Word	The default layout is Word.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Dialog Data Type

3/31/2019 • 2 minutes to read

Represents a dialog window.

The following methods are available on the Dialog data type.

METHOD NAME	DESCRIPTION
Error(String, [Any,...])	Displays an error message and ends the execution of AL code.
Error(ErrorInfo)	Displays an error message and ends the execution of AL code.
Message(String, [Any,...])	Displays a text string in a message window.
Confirm(String, [Boolean], [Any,...])	Creates a dialog box that prompts the user for a yes or no answer. The dialog box is centered on the screen.
StrMenu(String, [Integer], [String])	Creates a menu window that displays a series of options.

The following methods are available on instances of the Dialog data type.

METHOD NAME	DESCRIPTION
Open(String, [var Any,...])	Opens a dialog window.
Update([Integer], [Any])	Updates the value of a '#'-or '@' field in the active window.
Close()	Closes a dialog window that has been opened by the OPEN method.
HideSubsequentDialogs([Boolean])	Specifies that subsequent child dialogs are not shown.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Dictionary Data Type

3/31/2019 • 2 minutes to read

Represents a collection of keys and values.

The following methods are available on instances of the Dictionary data type.

METHOD NAME	DESCRIPTION
<code>Count()</code>	Gets the number of key/value pairs contained in the Dictionary.
<code>Keys()</code>	Gets a collection containing the keys in the Dictionary.
<code>Values()</code>	Gets a collection containing the values in the Dictionary.
<code>Get(TKey, var TValue)</code>	Gets the value associated with the specified key.
<code>Get(TKey)</code>	Gets the value associated with the specified key.
<code>Set(TKey, TValue)</code>	Sets the value associated with the specified key.
<code>Set(TKey, TValue, var TValue)</code>	Sets the value associated with the specified key.
<code>Add(TKey, TValue)</code>	Adds the specified key and value to the dictionary.
<code>ContainsKey(TKey)</code>	Determines whether the Dictionary contains the specified key.
<code>Remove(TKey)</code>	Removes the value with the specified key from the Dictionary.

Remarks

Each addition to the dictionary consists of a value, and its associated key. Every key in a Dictionary must be unique. A key cannot be null, but a value can be, only when the value type is a reference type.

Example

In the following example, the variable `counter` represents the Dictionary data type to store a value representing the number of occurrences for each character in the `customerName`. Using the `Get` method, you get the number of occurrences for the character at position `i`. If `i` returns **false**, it means there is no value associated with that character, so you add the value 1. If `i` returns **true**, it means the value already exists, so you add `c + 1` to the value. The `Add` method adds the {key:value} pair to the Dictionary.

```
procedure CountCharactersInCustomerName(customerName: Text; var counter: Dictionary of [Char, Integer]);  
var  
    i : Integer;  
    c : Integer;  
begin  
    for i := 1 to StrLen(customerName) do  
        begin  
            if counter.Get(customerName[i], c) then  
                counter.Set(customerName[i], c + 1)  
            else  
                counter.Add(customerName[i], 1);  
            end;  
        end;  
    end;  
end;
```

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

DotNet Data Type

3/31/2019 • 2 minutes to read

Represents an unspecified .NET type.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Duration Data Type

3/31/2019 • 2 minutes to read

Represents the difference between two `DateTime`s. This value can be negative. It is stored as a 64-bit integer. The integer value is the number of milliseconds during the duration.

The following are examples of durations:

`DATETIME-DATETIME=DURATION`

`DATETIME-DURATION=DATETIME`

`DATETIME+DURATION=DATETIME`

Example

This example shows how to calculate the difference between two `DateTime`s. It requires that you define the following variables.

VARIABLE	DATATYPE
DateTime1	DateTime
Datetime2	DateTime
Duration	Duration

This example is run on a computer with the Current Format in the Regional and Language Options set to English (United States).

```
DateTime1 := CREATEDATETIME(010109D, 080000T); // January 1, 2009 at 08:00:00 AM
DateTime2 := CREATEDATETIME(050509D, 133001T); // May 5, 2009 at 1:30:01 PM
Duration := DateTime2 - DateTime1;
MESSAGE(FORMAT(Duration));
```

The message window displays the following:

124 days 4 hours 30 minutes 1 second

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

ExecutionContext Option Type

3/31/2019 • 2 minutes to read

Represents the context in which a session is running. In certain scenarios, for example during upgrade, the system will run a session in a special context for a limited time.

Members

MEMBER	DESCRIPTION
Normal	The normal execution context.
Install	An application is being installed.
Uninstall	An application is being uninstalled.
Upgrade	An application is being upgraded.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

ExecutionMode Option Type

3/31/2019 • 2 minutes to read

The execution mode of the current session.

Members

MEMBER	DESCRIPTION
Standard	The session is executing in standard mode.
Debug	The session is executing in debug mode.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

FieldClass Option Type

3/31/2019 • 2 minutes to read

Represents the type of a field class.

Members

MEMBER	DESCRIPTION
Normal	A normal field.
FlowField	A flow field.
FlowFilter	A flow filter.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

FieldRef Data Type

3/31/2019 • 4 minutes to read

Identifies a field in a table and gives you access to this field.

The following methods are available on instances of the FieldRef data type.

METHOD NAME	DESCRIPTION
Number()	Gets the number of a field as a string.
Name()	Gets the name of a field as a string.
Caption()	Gets the current caption of a field as a String.
Type()	Gets the data type of the field that is currently selected.
Class()	Gets the value of the FieldClass Property of the field that is currently selected. This method returns an error if no field is selected.
OptionMembers()	Gets the list of options that are available in the field that is currently selected.
OptionString()	The 'OptionString' property has been deprecated and will be removed in the future. Use the 'OptionMembers' property instead.
OptionCaption()	Gets the option caption of the field that is currently selected.
Active()	Checks whether the field that is currently selected is enabled.
Record()	Gets the RecordRef of the field that is currently selected. This method returns an error if no field is selected.
Length()	Gets the maximum size of the field (the size specified in the DataLength property of the field). This method is usually used for finding the defined length of code and text fields.
Value([Any])	Sets or gets the value of the field that is currently selected. This method returns an error if no field is selected.
CalcField()	Updates FlowFields in a record.
CalcSum()	Calculates the total of all values of a SumIndexField in a table.
Validate([Any])	Use this method to enter a new value into a field and have the new value validated by the properties and code that have been defined for that field.

METHOD NAME	DESCRIPTION
FieldError([String])	Stops the execution of the code, causing a run-time error, and creates an error message for a field.
TestField()	Determines whether the contents of a field matches a given value. If the contents differ from the given value, an error message is displayed.
TestField(Byte)	Determines whether the contents of a field matches a given value. If the contents differ from the given value, an error message is displayed.
TestField(Boolean)	Determines whether the contents of a field matches a given value. If the contents differ from the given value, an error message is displayed.
TestField(Char)	Determines whether the contents of a field matches a given value. If the contents differ from the given value, an error message is displayed.
TestField(Option)	Determines whether the contents of a field matches a given value. If the contents differ from the given value, an error message is displayed.
TestField(Integer)	Determines whether the contents of a field matches a given value. If the contents differ from the given value, an error message is displayed.
TestField(BigInteger)	Determines whether the contents of a field matches a given value. If the contents differ from the given value, an error message is displayed.
TestField(Decimal)	Determines whether the contents of a field matches a given value. If the contents differ from the given value, an error message is displayed.
TestField(Guid)	Determines whether the contents of a field matches a given value. If the contents differ from the given value, an error message is displayed.
TestField(String)	Determines whether the contents of a field matches a given value. If the contents differ from the given value, an error message is displayed.
TestField(Text)	Determines whether the contents of a field matches a given value. If the contents differ from the given value, an error message is displayed.
TestField(Code)	Determines whether the contents of a field matches a given value. If the contents differ from the given value, an error message is displayed.
TestField(Date)	Determines whether the contents of a field matches a given value. If the contents differ from the given value, an error message is displayed.

METHOD NAME	DESCRIPTION
TestField(DateTime)	Determines whether the contents of a field matches a given value. If the contents differ from the given value, an error message is displayed.
TestField(Time)	Determines whether the contents of a field matches a given value. If the contents differ from the given value, an error message is displayed.
TestField(Variant)	Determines whether the contents of a field matches a given value. If the contents differ from the given value, an error message is displayed.
TestField(Any)	Determines whether the contents of a field matches a given value. If the contents differ from the given value, an error message is displayed.
Relation()	Finds the table relationship of a given field.
SetRange([Any], [Any])	Sets a simple filter on a field, such as a single range or a single value.
SetFilter(String, [Any,...])	Assigns a filter to a field that you specify.
GetFilter()	Gets the filter that is currently applied to the field referred to by FieldRef.
GetRangeMin()	Gets the minimum value in a range for a field.
GetRangeMax()	Gets the maximum value in a range for a field.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

FieldType Option Type

3/31/2019 • 3 minutes to read

Represents the type of a table field.

Members

MEMBER	DESCRIPTION
Boolean	Assumes the values true or false. When formatted, a Boolean field is shown as "Yes" or "No". The size of the corresponding SQL data type, TINYINT, is 1 byte.
Integer	Denotes an integer between -2,147,483,647 and 2,147,483,647. The size of the corresponding SQL data type, INTEGER, is 4 bytes.
BigInteger	A 64-bit integer.
Decimal	A decimal number between -10^{63} and 10^{63} . The exponent ranges from -63 to 63. Decimal numbers are held in memory with 18 significant digits. The representation of a decimal number is a Binary Coded Decimal (BCD). The size of the corresponding SQL data type, DECIMAL(38,20), is 17 bytes. We recommend that you construct decimals that operate on numbers in the range of +/- 999,999,999,999,999.99. You can construct larger numbers in some cases, but overflow, truncation or loss of precision can occur.
Option	An option field is defined by using an option string, which is a comma-separated list of strings that represent each valid value of the field. This string is used when a field of type Option is formatted and its value is converted into a string.
Text	Any alphanumeric string. The field must be defined to be between 1 and 250 characters. The number of bytes used by a text field equals (number of characters + 1) * 2. The additional character is used for the string terminating character, which is '0' in Unicode. The size of a Unicode character is 2 bytes. Therefore, you multiply the number of characters by two to get the size.
Code	An alphanumeric string, which is right-justified if the contents are numbers only. If letters or blanks occur among the numbers, the contents are left-aligned. All letters are converted to uppercase upon entry.
DateTime	Represents a point in time as a combined date and time. The datetime is stored in the database as Coordinated Universal Time (UTC) and is always displayed as local time in Dynamics 365 Business Central.

MEMBER	DESCRIPTION
Time	Any time in the range 00:00:00 to 23:59:59.999. A time field contains 1 plus the number of milliseconds since 00:00:00 o'clock, or 0 (zero), an undefined time.
Date	A date value in the range from January 1, 1753 to December 31, 9999. An undefined date is expressed as 0. All dates have a corresponding closing date. The system considers the closing date for a given date as a period that follows the given date but comes before the next normal date; that is, a closing date is sorted immediately after the corresponding normal date but before the next normal date.
DateFormula	Used to verify the date entered by the user.
Duration	Represents the difference between two points in time, in milliseconds. This value can be negative.
Guid	Globally unique identifier (GUID).
RecordId	Unique record identifier.
TableFilter	This data type is used to apply a filter to another table. Currently, this can only be used to apply security filters from the Permission table.
Blob	Binary Large Object. Used to store bitmaps and memos. Notice that the BLOB is not stored in the record, but in the BLOB area of the table.
Media	A complex type that encapsulates media files, such as image .jpg and .png files, in application database tables. The Media data type can be used as a table field data type, but cannot be used as a variable or parameter.
MediaSet	A complex type that encapsulates media, such as images, in application database tables. The MediaSet data type can be used as a table field data type, but cannot be used as variable or parameter.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

File Data Type

4/10/2019 • 3 minutes to read

Represents a file.

The following methods are available on the File data type.

METHOD NAME	DESCRIPTION
Erase(String)	Deletes a file.
Rename(String, String)	Renames an ASCII or binary file.
Copy(String, String)	Copies a file.
GetStamp(String, var Date, [var Time])	Gets the exact time that a file was last written to.
SetStamp(String, Date, [Time])	Sets a timestamp for a file.
Exists(String)	Determines whether a file exists.
UploadIntoStream(String, String, String, var Text, var InStream)	Sends a file from the client computer to the corresponding server. The client computer is the computer that is running the Windows client or the computer that is running a browser that accesses the web client.
DownloadFromStream(InStream, String, String, String, var Text)	Sends a file from server computer to the client computer. The client computer is the computer that is running the Windows client or the computer that is running the browser that accesses the web client.
Upload(String, String, String, String, var Text)	Sends a file from the client computer to the server computer. The client computer is the computer that is running the Windows client or the computer that is running a browser that accesses the web client.
Download(String, String, String, String, var Text)	Sends a file from a server computer to the client computer. The client computer is the computer that is running the Windows client or the computer that is running a browser that accesses the web client.
IsPathTemporary(String)	Validates whether the given path is located in the current users temporary folder within the current service.

The following methods are available on instances of the File data type.

METHOD NAME	DESCRIPTION
Open(String, [TextEncoding])	Opens an ASCII or binary file. This method does not create the file if it does not exist.
Create(String, [TextEncoding])	Creates an Automation object.

METHOD NAME	DESCRIPTION
Close()	Closes a file that has been opened by the OPEN method (File).
Name()	Gets the name of an ASCII or binary file.
Pos()	Gets the current position of the file pointer in an ASCII or binary file.
Len()	Gets the length of an ASCII or binary file.
Read(var Any)	Reads from an MS-DOS encoded file or binary file.
Write(Boolean)	Writes to an MS-DOS encoded file or binary file.
Write(Byte)	Writes to an MS-DOS encoded file or binary file.
Write(Char)	Writes to an MS-DOS encoded file or binary file.
Write(Integer)	Writes to an MS-DOS encoded file or binary file.
Write(BigInteger)	Writes to an MS-DOS encoded file or binary file.
Write(Decimal)	Writes to an MS-DOS encoded file or binary file.
Write(Guid)	Writes to an MS-DOS encoded file or binary file.
Write(Text)	Writes to an MS-DOS encoded file or binary file.
Write(Code)	Writes to an MS-DOS encoded file or binary file.
Write(Label)	Writes to an MS-DOS encoded file or binary file.
Write(BigText)	Writes to an MS-DOS encoded file or binary file.
Write(Date)	Writes to an MS-DOS encoded file or binary file.
Write(Time)	Writes to an MS-DOS encoded file or binary file.
Write(DateTime)	Writes to an MS-DOS encoded file or binary file.
Write(DateFormula)	Writes to an MS-DOS encoded file or binary file.
Write(Duration)	Writes to an MS-DOS encoded file or binary file.
Write(Option)	Writes to an MS-DOS encoded file or binary file.
Write(Record)	Writes to an MS-DOS encoded file or binary file.
Write(RecordId)	Writes to an MS-DOS encoded file or binary file.
Write(String)	Writes to an MS-DOS encoded file or binary file.

METHOD NAME	DESCRIPTION
Write(Any)	Writes to an MS-DOS encoded file or binary file.
Seek(Integer)	Sets a file pointer to a new position in an ASCII or binary file.
Trunc()	Truncate an ASCII or binary file to the current position of the file pointer.
WriteMode([Boolean])	Use this method before you use OPEN method (File)] to set or test whether you can write to a file in later calls.
TextMode([Boolean])	Sets whether a file should be opened as an ASCII file or a binary file. Gets the current setting of this option for a file.
CreateInStream(InStream)	Creates an InStream object for a file. This enables you to import or read data from the file.
CreateOutStream(OutStream)	Creates an OutStream object for a file. This enables you to export or write data to the file.
CreateTempFile([TextEncoding])	Creates a temporary file. This enables you to save data of any format to a temporary file. This file has a unique name and will be stored in a temporary file folder.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

FilterPageBuilder Data Type

3/31/2019 • 2 minutes to read

Stores filter configurations for a filter page. A filter page is a dynamic page type that contains one or more filter controls that enables users to set filters on fields of the underlying tables.

The following methods are available on instances of the FilterPageBuilder data type.

METHOD NAME	DESCRIPTION
AddTable(String, Integer)	Adds filter control for a table to a filter page.
AddRecord(String, Record)	Adds a filter control for a table to a filter page. The table is specified by a record data type variable that is passed to the method.
AddRecordRef(String, RecordRef)	Adds a filter control for a table to a filter page. The table is specified by a RecordRef variable that is passed to the method. This creates a filter control on the filter page, where users can set filter table data.
AddField(String, Any, [String])	Adds a table field to the filter control for a table on filter page.
AddField(String, FieldRef, [String])	Adds a table field to the filter control for a table on filter page.
AddFieldNo(String, Integer, [String])	Adds a table field to the filter control for a table on the filter page.
SetView(String, String)	Sets the current filter view, which defines the sort order, key, and filters, for a record in a filter control on a filter page. The view contains all fields that have default filters, but does not contain fields without filters.
GetView(String, [Boolean])	Gets the filter view (which defines the sort order, key, and filters) for the record in the specified filter control of a filter page. The view contains all fields in the filter control that have a default filter value.
RunModal()	Builds and runs the filter page that includes the filter controls that are stored in FilterPageBuilder object instance.
Count()	Gets the number of filter controls that are specified in the FilterPageBuilder object instance.
Name(Integer)	Gets the name of a table filter control that is included on a filter page based on an index number that is assigned to the filter control.
PageCaption([String])	Gets or sets the FilterPageBuilder UI caption. Defaults to the resource text if not explicitly set.

See Also

Guid Data Type

5/28/2019 • 3 minutes to read

Represents a 16 byte binary data type. This data type is used for the global identification of objects, programs, records, and so on. The important property of a GUID is that each value is globally unique. The value is generated by an algorithm, developed by Microsoft, which assures this uniqueness.

The GUID is a 16-byte binary data type that can be logically grouped into the following subgroups:

4byte-2byte-2byte-2byte-6byte.

The standard textual representation is {12345678-1234-1234-1234-1234567890AB}.

The virtual table OLE Control (2000000042) does not use the GUID data type. It uses a textual representation of the GUID in a text field instead. It is easier to make operations and references to this text field using the GUID data type than it is using the textual representation. The GUID data type is compatible with the existing textual representation.

The GUID data type is useful when you want to uniquely identify some data, so that it can be exchanged with external applications. For example, if you want to transfer an item catalog to an external application, you add a GUID field to the record in the table and use this as the primary reference when you communicate with the external application.

Compatibility

You can assign and compare the Text data type and the GUID data type. Assigning a GUID to a Text can be done as follows:

```
MyTableRec.MyGuid := MyTableRec.MyText;
```

The supported formats of `MyText` are:

```
'11111111-1111-1111-1111-111111111111' '{22222222-2222-2222-2222-222222222222}'
```

Methods and properties

The following AL methods can be used with the GUID data type:

```
Guid := CREATEGUID();
```

This method creates a new unique GUID value. The value can then be assigned to a field of the GUID data type or of the Text data type.

```
Ok := ISNULLGUID(Guid);
```

This method is a convenient way to check if a value has already been assigned to a GUID. A NULL GUID (consisting only of zeroes) is valid, but should never be used for reference purposes.

A NULL GUID is valid but is not useful in a table. Therefore, the **AutoSplitKey** property is implemented for the GUID data type when it is used in a page. When GUID is selected as a primary key, **AutoSplitKey** is enabled for the page, and the GUID value remains NULL. When you create a new record, a valid GUID is created and assigned

automatically.

The [CREATEGUID method](#) and [ISNULLGUID method](#) methods are available in the AL Symbol Menu under SYSTEM, Variables.

CREATEGUID takes no arguments and returns a valid 16-byte GUID value. If the result is assigned to a TEXT variable or field, the value is converted to a string and follows the syntax explained earlier. The algorithm that generates the new GUID value uses Microsoft's CoCreateGuid method.

ISNULLGUID takes a GUID value as a required argument and returns TRUE/FALSE depending on whether the GUID value is NULL. This method does not accept a Text value as an argument.

AutoSplitKey is a property, not a method and can be applied to pages. If you have defined a GUID field as part of the primary key, the **AutoSplitKey** property automatically generates a new valid GUID value. When a new record is created and the GUID field is left as NULL, the **AutoSplitKey** property ensures that a valid GUID value is automatically inserted into the field. If you then enter a NULL GUID into this record, for example, by using the CLEAR method, this new NULL GUID value is not automatically replaced by the **AutoSplitKey** property. The **AutoSplitKey** property only applies to new records.

Format

The GUID value can also be represented as text. You can use the standard AL methods [FORMAT](#) and [EVALUATE](#) to convert from GUID values to Text values. If you do not use the correct format when you edit a GUID value in its textual format, the following error message is displayed:

Invalid Format of GUID string. The correct format of the GUID string is {CDEF7890-ABCD-1234-ABCD-1234567890AB} where 0-9, A-F symbolizes hexadecimal digits.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

HttpClient Data Type

3/31/2019 • 2 minutes to read

Provides a data type for sending HTTP requests and receiving HTTP responses from a resource identified by a URI.

The following methods are available on instances of the HttpClient data type.

METHOD NAME	DESCRIPTION
Clear()	Sets the HttpClient variable to the default value.
GetBaseAddress()	Gets the base address of Uniform Resource Identifier (URI) of the Internet resource used when sending requests.
SetBaseAddress(String)	Sets the base address of Uniform Resource Identifier (URI) of the Internet resource used when sending requests.
DefaultRequestHeaders()	Gets the default request headers which should be sent with each request.
Timeout([Duration])	Gets or sets the duration in seconds to wait before the request times out.
Get(String, var HttpResponseMessage)	Sends a GET request to get the resource identified by the request URL.
Delete(String, var HttpResponseMessage)	Sends a DELETE request to delete the resource identified by the request URL.
Post(String, HttpContent, var HttpResponseMessage)	Sends a POST request to the specified URI as an asynchronous operation.
Put(String, HttpContent, var HttpResponseMessage)	Sends a PUT request to the specified URI as an asynchronous operation.
Send(HttpRequestMessage, var HttpResponseMessage)	Sends an HTTP request as an asynchronous operation.
AddCertificate(String, [String])	Adds a certificate to the HttpClient class.
UseDefaultNetworkWindowsAuthentication()	Sets the HttpClient credentials to use the default network credentials for Windows authentication. If this method is invoked after any HTTP request has started; a runtime error occurs.
UseWindowsAuthentication(String, String, [String])	Sets the HttpClient credentials to use the specified network credentials for Windows authentication. If this method is invoked after any HTTP request has started; a runtime error occurs.

See Also

HttpContent Data Type

3/31/2019 • 2 minutes to read

Represents an HTTP entity body and content headers.

The following methods are available on instances of the HttpContent data type.

METHOD NAME	DESCRIPTION
Clear()	Sets the HttpContent object to a default value. The content contains an empty string and empty headers.
WriteFrom(Text)	Sets HttpContent content to the provided text or stream.
WriteFrom(InStream)	Sets HttpContent content to the provided text or stream.
GetHeaders(var HttpHeaders)	Gets the HTTP content headers as defined in RFC 2616.
ReadAs(var Text)	Reads the content into the provided text.
ReadAs(var InStream)	Reads the content into the provided text.

An instance of HttpContent encapsulates the body and the associated headers of an HTTP request that will be sent to a remote endpoint or that is being received from a remote endpoint. The HttpContent data type is a value type. This means that when assigning an instance of HttpContent to a variable, a copy will be created.

Example

The following example illustrates how to use the HttpContent type to send a simple POST request containing JSON data.

```

codeunit 50110 MyCodeunit
{
    procedure MakeRequest(uri: Text; payload: Text) responseText: Text;
    var
        client: HttpClient;
        request: HttpRequestMessage;
        response: HttpResponseMessage;
        contentHeaders: HttpHeaders;
        content: HttpContent;
    begin
        // Add the payload to the content
        content.WriteFrom(payload);

        // Retrieve the contentHeaders associated with the content
        content.GetHeaders(contentHeaders);
        contentHeaders.Clear();
        contentHeaders.Add('Content-Type', 'application/json');

        // Assigning content to request.Content will actually create a copy of the content and assign it.
        // After this line, modifying the content variable or its associated headers will not reflect in
        // the content associated with the request message
        request.Content := content;

        request.SetRequestUri(uri);
        request.Method := 'POST';

        client.Send(request, response);

        // Read the response content as json.
        response.Content().ReadAs(responseText);
    end;
}

```

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

HttpHeaders Data Type

5/28/2019 • 2 minutes to read

Is a collection of headers and their values.

The following methods are available on instances of the HttpHeaders data type.

METHOD NAME	DESCRIPTION
Add(String, String)	Adds the specified header and its value into the HttpHeaders collection. Validates the provided value.
TryAddWithoutValidation(String, String)	Adds the specified header and its value into the HttpHeaders collection. Doesn't validate the provided value.
Contains(String)	Checks if the specified header exists in the HttpHeaders collection.
Clear()	Sets the HttpHeaders variable to the default value.
Remove(String)	Removes the specified header from the HttpHeaders collection.
GetValues(String, Array of [Text])	Gets the values for the specified key.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

HttpRequestMessage Data Type

5/28/2019 • 2 minutes to read

Represents an HTTP request message.

The following methods are available on instances of the HttpRequestMessage data type.

METHOD NAME	DESCRIPTION
Content([HttpContent])	Gets or sets the contents of the HTTP message.
Method([String])	Gets or sets the method type as defined in the HTTP standard.
GetRequestUri()	Gets the URI used for the HTTP request.
SetRequestUri(String)	Sets the URI used for the HTTP request.
GetHeaders(var HttpHeaders)	Gets a reference to the collection of HTTP request headers.

NOTE

For performance reasons all HTTP, JSON, TextBuilder, and XML types are reference types, not value types. Reference types holds a pointer to the data elsewhere in memory, whereas value types store its own data.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

HttpResponseMessage Data Type

3/31/2019 • 2 minutes to read

Represents a HTTP response message including the status code and data.

The following methods are available on instances of the HttpResponseMessage data type.

METHOD NAME	DESCRIPTION
Content()	Gets the contents of the HTTP response.
Headers()	Gets the HTTP response's HTTP headers.
IsSuccessStatusCode()	Gets a value that indicates if the HTTP response was successful.
IsBlockedByEnvironment()	Gets a value that indicates if the HTTP response is the result of the environment blocking an outgoing HTTP request.
ReasonPhrase()	Gets the reason phrase which typically is sent by servers together with the status code.
HttpStatusCode()	Gets the status code of the HTTP response.

NOTE

For performance reasons all HTTP, JSON, TextBuilder, and XML types are reference types, not value types. Reference types holds a pointer to the data elsewhere in memory, whereas value types store its own data.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

InStream Data Type

3/31/2019 • 2 minutes to read

Is a generic stream object that you can use to read from or write to files and BLOBs. You can define the internal structure of a stream as a flat stream of bytes. You can assign one stream to another. Reading from and writing to a stream occurs sequentially.

The following methods are available on instances of the InStream data type.

METHOD NAME	DESCRIPTION
Read(var Boolean, [Integer])	Reads a specified number of bytes from an InStream object. Data is read in binary format.
Read(var Byte, [Integer])	Reads a specified number of bytes from an InStream object. Data is read in binary format.
Read(var Char, [Integer])	Reads a specified number of bytes from an InStream object. Data is read in binary format.
Read(var Integer, [Integer])	Reads a specified number of bytes from an InStream object. Data is read in binary format.
Read(var BigInteger, [Integer])	Reads a specified number of bytes from an InStream object. Data is read in binary format.
Read(var Decimal, [Integer])	Reads a specified number of bytes from an InStream object. Data is read in binary format.
Read(var Guid, [Integer])	Reads a specified number of bytes from an InStream object. Data is read in binary format.
Read(var String, [Integer])	Reads a specified number of bytes from an InStream object. Data is read in binary format.
Read(var Any, [Integer])	Reads a specified number of bytes from an InStream object. Data is read in binary format.
EOS()	Indicates whether an input stream has reached End of Stream (EOS).
ReadText(var Text, [Integer])	Reads text from an InStream object.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Integer Data Type

3/31/2019 • 2 minutes to read

Stores whole numbers with values that range from -2,147,483,647 to 2,147,483,647.

Remarks

In addition to representing whole numbers in this range, you can use integers to represent Boolean values. For Boolean values, -1 represents **true** and 0 represents **false**.

If you assign -2,147,483,648 directly to an Integer variable, then you get an error when you try to compile the code. However, you can indirectly assign -2,147,483,648 to an Integer variable by using the following code.

```
IntegerVar := -2147483647;  
IntegerVar := IntegerVar - 1;
```

If you try to indirectly assign a value that is smaller than -2,147,483,648 or larger than 2,147,483,647, then you get a run-time error.

Example

The following are examples of integer values.

```
546  
-3425
```

Example

The following example is a decimal and not an integer.

```
342.45
```

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

IsolatedStorage Data Type

3/31/2019 • 2 minutes to read

Provides data isolation for extensions.

The following methods are available on the IsolatedStorage data type.

METHOD NAME	DESCRIPTION
Set(String, String, [DataScope])	Sets the value associated with the specified key.
Get(String, [DataScope], var Text)	Gets the value associated with the specified key.
Get(String, var Text)	Gets the value associated with the specified key.
Contains(String, [DataScope])	Determines whether the storage contains a value with the specified key.
Delete(String, [DataScope])	Deletes the value with the specified key from the isolated storage.
SetEncrypted(String, String, [DataScope])	Encrypts and sets the value associated with the specified key. The input string cannot exceed a length of 215 plain characters; be aware that special characters take up more space.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

JSONArray Data Type

4/10/2019 • 5 minutes to read

Is a container for any well-formed JSON array. A default JSONArray contains an empty JSON array.

The following methods are available on instances of the JSONArray data type.

METHOD NAME	DESCRIPTION
Path()	Retrieves the JSON path of the array relative to the root of its containing tree.
ReadFrom(String)	Reads the JSON data from the string into a JSONArray variable.
ReadFrom(InStream)	Reads the JSON data from the stream into a JSONArray variable.
WriteTo(var Text)	Serializes and writes the JSON data of the JSONArray to a given Text object.
WriteTo(OutStream)	Serializes and writes the JSON data of the JSONArray to a given OutStream object.
SelectToken(String, var JsonToken)	Selects a JsonToken using a JPath expression.
Clone()	Creates a deep-copy of the JSONArray value.
AsToken()	Converts the value in a JSONArray to a JsonToken data type.
Count()	Gets the number of elements in the JSONArray.
Get(Integer, var JsonToken)	Retrieves the value at the given index in the JSONArray.
Set(Integer, JsonToken)	Replaces the value at the given index with a new value.
Set(Integer, JsonObject)	Replaces the value at the given index with a new value.
Set(Integer, JSONArray)	Replaces the value at the given index with a new value.
Set(Integer, JsonValue)	Replaces the value at the given index with a new value.
Set(Integer, Boolean)	Replaces the value at the given index with a new value.
Set(Integer, Char)	Replaces the value at the given index with a new value.
Set(Integer, Byte)	Replaces the value at the given index with a new value.
Set(Integer, Option)	Replaces the value at the given index with a new value.
Set(Integer, Integer)	Replaces the value at the given index with a new value.

METHOD NAME	DESCRIPTION
Set(Integer, BigInteger)	Replaces the value at the given index with a new value.
Set(Integer, Decimal)	Replaces the value at the given index with a new value.
Set(Integer, Duration)	Replaces the value at the given index with a new value.
Set(Integer, Date)	Replaces the value at the given index with a new value.
Set(Integer, Time)	Replaces the value at the given index with a new value.
Set(Integer, DateTime)	Replaces the value at the given index with a new value.
Set(Integer, String)	Replaces the value at the given index with a new value.
RemoveAt(Integer)	Removes the token at the given index.
Insert(Integer, JsonToken)	Inserts the value at the given index in the array while shifting all the values to the right by one position.
Insert(Integer, JsonArray)	Inserts the value at the given index in the array while shifting all the values to the right by one position.
Insert(Integer, JsonObject)	Inserts the value at the given index in the array while shifting all the values to the right by one position.
Insert(Integer, JsonValue)	Inserts the value at the given index in the array while shifting all the values to the right by one position.
Insert(Integer, Boolean)	Inserts the value at the given index in the array while shifting all the values to the right by one position.
Insert(Integer, Char)	Inserts the value at the given index in the array while shifting all the values to the right by one position.
Insert(Integer, Byte)	Inserts the value at the given index in the array while shifting all the values to the right by one position.
Insert(Integer, Option)	Inserts the value at the given index in the array while shifting all the values to the right by one position.
Insert(Integer, Integer)	Inserts the value at the given index in the array while shifting all the values to the right by one position.
Insert(Integer, BigInteger)	Inserts the value at the given index in the array while shifting all the values to the right by one position.
Insert(Integer, Decimal)	Inserts the value at the given index in the array while shifting all the values to the right by one position.
Insert(Integer, Duration)	Inserts the value at the given index in the array while shifting all the values to the right by one position.

METHOD NAME	DESCRIPTION
Insert(Integer, Date)	Inserts the value at the given index in the array while shifting all the values to the right by one position.
Insert(Integer, Time)	Inserts the value at the given index in the array while shifting all the values to the right by one position.
Insert(Integer, DateTime)	Inserts the value at the given index in the array while shifting all the values to the right by one position.
Insert(Integer, String)	Inserts the value at the given index in the array while shifting all the values to the right by one position.
Add(JsonToken)	Adds a new value at the end of the JSONArray.
Add(JsonArray)	Adds a new value at the end of the JSONArray.
Add(JsonObject)	Adds a new value at the end of the JSONArray.
Add(JsonValue)	Adds a new value at the end of the JSONArray.
Add(Boolean)	Adds a new value at the end of the JSONArray.
Add(Char)	Adds a new value at the end of the JSONArray.
Add(Byte)	Adds a new value at the end of the JSONArray.
Add(Option)	Adds a new value at the end of the JSONArray.
Add(Integer)	Adds a new value at the end of the JSONArray.
Add(BigInteger)	Adds a new value at the end of the JSONArray.
Add(Decimal)	Adds a new value at the end of the JSONArray.
Add(Duration)	Adds a new value at the end of the JSONArray.
Add(Date)	Adds a new value at the end of the JSONArray.
Add(Time)	Adds a new value at the end of the JSONArray.
Add(DateTime)	Adds a new value at the end of the JSONArray.
Add(String)	Adds a new value at the end of the JSONArray.
IndexOf(JsonToken)	Determines the index of a specific value in the JSONArray.
IndexOf(JsonArray)	Determines the index of a specific value in the JSONArray.
IndexOf(JsonObject)	Determines the index of a specific value in the JSONArray.
IndexOf(JsonValue)	Determines the index of a specific value in the JSONArray.

METHOD NAME	DESCRIPTION
IndexOf(Boolean)	Determines the index of a specific value in the JsonArray.
IndexOf(Char)	Determines the index of a specific value in the JsonArray.
IndexOf(Byte)	Determines the index of a specific value in the JsonArray.
IndexOf(Option)	Determines the index of a specific value in the JsonArray.
IndexOf(Integer)	Determines the index of a specific value in the JsonArray.
IndexOf(BigInteger)	Determines the index of a specific value in the JsonArray.
IndexOf(Decimal)	Determines the index of a specific value in the JsonArray.
IndexOf(Duration)	Determines the index of a specific value in the JsonArray.
IndexOf(Date)	Determines the index of a specific value in the JsonArray.
IndexOf(Time)	Determines the index of a specific value in the JsonArray.
IndexOf(DateTime)	Determines the index of a specific value in the JsonArray.
IndexOf(String)	Determines the index of a specific value in the JsonArray.

NOTE

For performance reasons all HTTP, JSON, TextBuilder, and XML types are reference types, not value types. Reference types holds a pointer to the data elsewhere in memory, whereas value types store its own data.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

JsonObject Data Type

3/31/2019 • 3 minutes to read

Is a container for any well-formed JSON object. A default JsonObject contains an empty JSON object.

The following methods are available on instances of the JsonObject data type.

METHOD NAME	DESCRIPTION
Path()	Retrieves the JSON path of the object relative to the root of its containing tree.
Keys()	Gets a set of keys of the JsonObject.
Values()	Gets a set of values of the JsonObject.
ReadFrom(String)	Reads the JSON data from the string into a JsonObject variable.
ReadFrom(InStream)	Reads the JSON data from the stream into a JsonObject variable.
WriteTo(var Text)	Serializes and writes the JSON data of the JsonObject to a given Text object.
WriteTo(OutStream)	Serializes and writes the JSON data of the JsonObject to a given OutStream object.
SelectToken(String, var JsonToken)	Selects a JsonToken using a JPath expression.
Clone()	Creates a deep-copy of the JsonToken value.
AsToken()	Converts the value in a JsonObject to a JsonToken data type.
Contains(String)	Verifies if a JsonObject contains a property with a given key.
Get(String, var JsonToken)	Retrieves the value of a property with a given key from a JsonObject.
Add(String, JsonToken)	Adds a new property to a JsonObject.
Add(String, JsonObject)	Adds a new property to a JsonObject.
Add(String, JsonValue)	Adds a new property to a JsonObject.
Add(String, JsonArray)	Adds a new property to a JsonObject.
Add(String, Boolean)	Adds a new property to a JsonObject.
Add(String, Char)	Adds a new property to a JsonObject.

METHOD NAME	DESCRIPTION
Add(String, Byte)	Adds a new property to a JsonObject.
Add(String, Option)	Adds a new property to a JsonObject.
Add(String, Integer)	Adds a new property to a JsonObject.
Add(String, BigInteger)	Adds a new property to a JsonObject.
Add(String, Decimal)	Adds a new property to a JsonObject.
Add(String, Duration)	Adds a new property to a JsonObject.
Add(String, String)	Adds a new property to a JsonObject.
Add(String, Date)	Adds a new property to a JsonObject.
Add(String, Time)	Adds a new property to a JsonObject.
Add(String, DateTime)	Adds a new property to a JsonObject.
Replace(String, JsonToken)	Replaces the value of the property with the given key with the new value.
Replace(String, JsonArray)	Replaces the value of the property with the given key with the new value.
Replace(String, JsonObject)	Replaces the value of the property with the given key with the new value.
Replace(String, JsonValue)	Replaces the value of the property with the given key with the new value.
Replace(String, Boolean)	Replaces the value of the property with the given key with the new value.
Replace(String, Char)	Replaces the value of the property with the given key with the new value.
Replace(String, Byte)	Replaces the value of the property with the given key with the new value.
Replace(String, Integer)	Replaces the value of the property with the given key with the new value.
Replace(String, Option)	Replaces the value of the property with the given key with the new value.
Replace(String, BigInteger)	Replaces the value of the property with the given key with the new value.
Replace(String, Decimal)	Replaces the value of the property with the given key with the new value.

METHOD NAME	DESCRIPTION
Replace(String, Duration)	Replaces the value of the property with the given key with the new value.
Replace(String, Date)	Replaces the value of the property with the given key with the new value.
Replace(String, Time)	Replaces the value of the property with the given key with the new value.
Replace(String, DateTime)	Replaces the value of the property with the given key with the new value.
Replace(String, String)	Replaces the value of the property with the given key with the new value.
Remove(String)	Removes the property with the given key from the object.

NOTE

For performance reasons all HTTP, JSON, TextBuilder, and XML types are reference types, not value types. Reference types holds a pointer to the data elsewhere in memory, whereas value types store its own data.

Remarks

An uninitialized variable of JsonObject type represents an empty JSON object. Given a value of JsonObject type, you can check if it is empty by checking that the number of keys in the object is 0.

```
jsonObject.Keys.Count = 0
```

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

JsonToken Data Type

3/31/2019 • 2 minutes to read

Is a container for any well-formed JSON data. A default JsonToken object contains the JSON value of NULL.

The following methods are available on instances of the JsonToken data type.

METHOD NAME	DESCRIPTION
IsObject()	Indicates whether a JsonToken contains a JSON object.
IsValue()	Indicates whether a JsonToken contains a JSON value.
IsArray()	Indicates whether a JsonToken represents a JSON array.
AsArray()	Converts the value in a JsonToken to a JsonArray data type.
AsValue()	Converts the value in a JsonToken to a JsonValue data type.
AsObject()	Converts the value in a JsonToken to a JsonObject data type.
Path()	Retrieves the JSON path of the token relative to the root of its containing tree.
ReadFrom(String)	Reads the JSON data from the string into a JsonToken variable.
ReadFrom(InStream)	Reads the JSON data from the stream into a JsonToken variable.
WriteTo(var Text)	Serializes and writes the JSON data of the JsonToken to a given Text object.
WriteTo(OutStream)	Serializes and writes the JSON data of the JsonToken to a given OutStream object.
SelectToken(String, var JsonToken)	Selects a JsonToken using a JPath expression.
Clone()	Creates a deep-copy of the JsonToken value.

NOTE

For performance reasons all HTTP, JSON, TextBuilder, and XML types are reference types, not value types. Reference types holds a pointer to the data elsewhere in memory, whereas value types store its own data.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

JsonValue Data Type

3/31/2019 • 2 minutes to read

Is a container for any well-formed fundamental JSON value. A default JsonValue is set to the JSON value of NULL.

The following methods are available on instances of the JsonValue data type.

METHOD NAME	DESCRIPTION
Path()	Retrieves the JSON path of the value relative to its containing tree.
ReadFrom(String)	Reads the JSON data into a JsonValue variable.
ReadFrom(InStream)	Reads the JSON data from the stream into a JsonValue variable.
WriteTo(var Text)	Serializes and writes the JSON data of the JsonValue to a given object.
WriteTo(OutStream)	Serializes and writes the JSON data of the JsonValue to a given object.
SelectToken(String, var JsonToken)	Selects a JsonToken using a JPath expression.
Clone()	Creates a deep-copy of the JsonToken value.
AsToken()	Converts the value in a JsonValue to a JsonToken data type.
IsNull()	Indicates whether the JsonValue contains the JSON value of NULL.
IsUndefined()	Indicates whether the JsonValue contains the JSON value of UNDEFINED.
AsBoolean()	Converts the value in a JsonValue to a Boolean data type.
AsChar()	Converts the value in a JsonValue to a Char data type.
AsByte()	Converts the value in a JsonValue to a Byte data type.
AsOption()	Converts the value in a JsonValue to an Option data type.
AsInteger()	Converts the value in a JsonValue to an Integer data type.
AsBigInteger()	Converts the value in a JsonValue to an BigInteger data type.
AsDecimal()	Converts the value in a JsonValue to a Decimal data type.

METHOD NAME	DESCRIPTION
AsDuration()	Converts the value in a JsonValue to a Duration data type.
AsDate()	Converts the value in a JsonValue to a Date data type.
AsTime()	Converts the value in a JsonValue to a Time data type.
AsDateTime()	Converts the value in a JsonValue to a DateTime data type.
AsText()	Converts the value in a JsonValue to a Text data type.
AsCode()	Converts the value in a JsonValue to a Code data type.
SetValue(Boolean)	Set the contents of the JsonValue variable to the JSON representation of the given value.
SetValue(Char)	Set the contents of the JsonValue variable to the JSON representation of the given value.
SetValue(Byte)	Set the contents of the JsonValue variable to the JSON representation of the given value.
SetValue(Option)	Set the contents of the JsonValue variable to the JSON representation of the given value.
SetValue(Integer)	Set the contents of the JsonValue variable to the JSON representation of the given value.
SetValue(BigInteger)	Set the contents of the JsonValue variable to the JSON representation of the given value.
SetValue(Decimal)	Set the contents of the JsonValue variable to the JSON representation of the given value.
SetValue(Duration)	Set the contents of the JsonValue variable to the JSON representation of the given value.
SetValue(Date)	Set the contents of the JsonValue variable to the JSON representation of the given value.
SetValue(Time)	Set the contents of the JsonValue variable to the JSON representation of the given value.
SetValue(DateTime)	Set the contents of the JsonValue variable to the JSON representation of the given value.
SetValue(String)	Set the contents of the JsonValue variable to the JSON representation of the given value.
SetValueToNull()	Set the contents of the JsonValue variable to the JSON representation of NULL.
SetValueToUndefined()	Set the contents of the JsonValue variable to the JSON representation of UNDEFINED.

NOTE

For performance reasons all HTTP, JSON, TextBuilder, and XML types are reference types, not value types. Reference types holds a pointer to the data elsewhere in memory, whereas value types store its own data.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

KeyRef Data Type

3/31/2019 • 2 minutes to read

Identifies a key in a table and the fields in this key.

The following methods are available on instances of the KeyRef data type.

METHOD NAME	DESCRIPTION
FieldCount()	Gets the number of fields that have been defined in a key. Returns an error if no key is selected.
FieldIndex(Integer)	Gets the FieldRef of the field that has this index in the key referred to by the KeyRef variable. Returns an error if no key is selected.
Record()	Returns a RecordRef for the current record referred to by the key.
Active()	Indicates whether the key is enabled.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Label Data Type

4/24/2019 • 2 minutes to read

Denotes a string constant that can be optionally translated into multiple languages.

Parameters

All of the parameters below are optional and the order is not enforced.

ATTRIBUTE	DESCRIPTION
Comment	It is used for general comments about the label, specifically about the placeholders in that label.
Locked	When Locked is set to true , the label should not be translated. Default value is false .
MaxLength	<div>It determines how much of the label is used.</div> <div><pre>Label 'ALLOWED POSTING DATE', Comment='{MaxLength=30}';</pre></div> <div>If no maximum length is specified, the string can be any length.</div>

Syntax example

```
var  
a:Label 'LabelText',Comment='Foo',MaxLength=999,Locked=true;
```

Remarks

The `Label` data type is used in .xlf files for translations. For more information, see [Working with Translation Files](#).

For information about naming, see [CodeCop Rule AA0074](#).

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

List Data Type

5/28/2019 • 2 minutes to read

Represents a strongly typed list of objects that can be accessed by index.

The following methods are available on instances of the List data type.

METHOD NAME	DESCRIPTION
Count()	Gets the number of elements contained in the List.
Add(T)	Adds a value to the end of the List.
AddRange(T, [T,...])	Adds the elements of the specified collection to the end of the list.
AddRange(List of [T])	Adds the elements of the specified collection to the end of the list.
Get(Integer, var T)	Gets the element at the specified index.
Get(Integer)	Gets the element at the specified index. This method will raise an error if the index is outside the valid range.
Set(Integer, T)	Sets the element at the specified index.
Set(Integer, T, var T)	Sets the element at the specified index.
Contains(T)	Determines whether an element is in the List.
IndexOf(T)	Searches for the specified value and returns the one-based index of the first occurrence within the entire List.
Insert(Integer, T)	Inserts an element into the List at the specified index.
LastIndexOf(T)	Searches for the specified value and returns the one-based index of the last occurrence within the entire List.
Remove(T)	Removes the first occurrence of a specified value from the List.
RemoveAt(Integer)	Removes the element at the specified index of the List.
RemoveRange(Integer, Integer)	Removes a range of elements from the List.
GetRange(Integer, Integer)	Get a shallow copy of a range of elements in the source.
GetRange(Integer, Integer, var List of [T])	Get a shallow copy of a range of elements in the source.
Reverse()	Reverses the order of the elements in the entire List.

Remarks

The List can only be used with simple types i.e. you can have a List of [Integer] but cannot have a List of [Blob].

Example

In the following example, the variable `CustomerNames` is a list of Text values which represent customer names. The procedure `WorkWithListOfCustomers` displays how one would work with the List data type. The `Add` method is used to add the string `'John'` to the `CustomerNames` list. The `Contains` method is used to check whether the list contains the specified value, in this case, the string `'John'`. We continue by using the Message procedure to display a relevant message.

```
procedure WorkWithListOfCustomers();
var
    customerNames : List of [Text];
begin
    // Adding an element to the list
    customerNames.Add('John');

    // Checking if the list contains an element
    if customerNames.Contains('John') then
        Message('John is in the list')
    else
        Message('John is not in the list')
end;
```

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Media Data Type

3/31/2019 • 2 minutes to read

Encapsulates media files, such as image .jpg and .png files, in application database tables. The Media data type can be used as a table field data type, but cannot be used as a variable or parameter. The Media data type enables you to import a media file to the application database and reference the file from records, making it possible to display the media file in the client user interface. You can also export media from the database to files and streams.

The following methods are available on instances of the Media data type.

METHOD NAME	DESCRIPTION
HasValue()	Checks whether a Media data type field in a record has been initialized with a media object and that the specified media object exists in the database.
ExportFile(String)	Exports the media object (such as an image) that is currently used on record to a file on your computer or network. On the record, the media object is referenced in a Media data type field.
ExportStream(OutStream)	Exports the current media object (such as a JPEG image) that is used on record to an OUTSTREAM object. The OUTSTREAM object can be created from a BLOB field, a FILE or from a .NET Framework interoperability object. In the record, the media is referenced in a Media data type field.
ImportFile(Text, Text, [Text])	Adds a media type, such as a JPEG image, from a file to a Media data type field of a record for displaying the media with the record in the client. The media file is imported to the application database, and a reference to the media is included in the Media data type field.
ImportStream(InStream, Text, [Text])	Adds a media type (MIME), such as jpeg image, from an InStream object to a Media data type field of a record for displaying the media in the client. The media file is imported to the application database and a reference to the media is included in the Media data type field.
ImportStream(InStream, Text, Text, Text)	Adds a media type (MIME), such as jpeg image, from an InStream object to a Media data type field of a record for displaying the media in the client. The media file is imported to the application database and a reference to the media is included in the Media data type field.
MediaId()	Gets the unique identifier of a media object on a record.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

MediaSet Data Type

3/31/2019 • 2 minutes to read

Encapsulates media, such as images, in application database tables.

The following methods are available on instances of the MediaSet data type.

METHOD NAME	DESCRIPTION
Count()	Gets the number of media objects that are included in the MediaSet of a record.
ExportFile(String)	Exports the media objects in the current media set of a record to individual files on your computer or network. In the record, the media set is referenced in a MediaSet data type field.
ImportFile(String, String, [String])	Adds a media, such as a JPEG image, to the MediaSet data type field of a record for displaying the media in the client. The media is imported to the database and included in a MediaSet for the record.
ImportStream(InStream, String, [String])	Adds a media file, such as a JPEG image, from an InStream object to the MediaSet of record for displaying in the client. The media is imported to the database and included in a MediaSet for the record.
Insert(Guid)	Adds a media object that already exists in the database to a MediaSet of a record.
Item(Integer)	Gets the unique identifier (GUID) of a media object that is assigned to a MediaSet on a record.
MediaId()	Gets the unique identifier that is assigned to a MediaSet of a record. The MediaSet is a collection of media objects that are used on the record that can be displayed in the client.
Remove(Guid)	Removes a media object from a MediaSet of a record.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

ModuleDependencyInfo Data Type

3/31/2019 • 2 minutes to read

Provides information about a dependent module.

The following methods are available on instances of the ModuleDependencyInfo data type.

METHOD NAME	DESCRIPTION
Id()	Gets the app ID of the specified app.
Name()	Gets the name of the specified application.
Publisher()	Gets the publisher of the specified application.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

ModuleInfo Data Type

3/31/2019 • 2 minutes to read

Represents information about an application consumable from AL.

The following methods are available on instances of the ModuleInfo data type.

METHOD NAME	DESCRIPTION
AppVersion()	Gets the version of the specified application's metadata.
DataVersion()	Gets the version of the specified application's data in the context of a given tenant. This indicates the last version that was installed or successfully upgraded to and will not match the application version if the tenant is in a data upgrade pending state.
Dependencies()	Gets the collection of application dependencies.
Id()	Gets the ID of the specified application.
Name()	Gets the name of the specified application.
Publisher()	Gets the publisher of the specified application.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

NavApp Data Type

3/31/2019 • 2 minutes to read

Provides information about a NavApp.

The following methods are available on the NavApp data type.

METHOD NAME	DESCRIPTION
GetArchiveRecordRef(Integer, var RecordRef)	Returns a RecordRef for the specified table.
GetArchiveVersion()	Returns the version of the extension that the specified table is part of.
RestoreArchiveData(Integer, [Boolean])	Restores archived data for a specified table of an extension during installation.
DeleteArchiveData(Integer)	Deletes the archived data for a specified table of an extension during installation.
LoadPackageData(Integer)	Loads default, or starting, table data into the specified table of an extension during installation.
IsInstalling()	Returns true if the application that contains the AL object that is currently running is being installed, otherwise it returns false .
GetCurrentModuleInfo(var ModuleInfo)	Gets information about the application that contains the AL object that is currently running.
GetModuleInfo(Guid, var ModuleInfo)	Gets information about the specified AL application.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

None Data Type

3/31/2019 • 2 minutes to read

Is used implicitly when a method does not return a value.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Notification Data Type

3/31/2019 • 2 minutes to read

Provides a programmatic way to send non-intrusive information to the user interface (UI) in the Business Central Web client.

The following methods are available on instances of the Notification data type.

METHOD NAME	DESCRIPTION
Id([Guid])	Specifies the identifier for a notification.
AddAction(String, Integer, String)	Specifies an action for the notification.
GetData(String)	Retrieves data that was passed to a notification instance as specified by a SETDATA method call.
Message([String])	Specifies the content of the notification.
Scope([NotificationScope])	Specifies the context in which the notification appears in the client.
Send()	Sends the notification to the client, where it will display in the UI.
Recall()	Recall a sent notification.
SetData(String, String)	Specifies a data property value for the notification. The data is specified as text in a key-value pair.
HasData(String)	Checks if data was passed to a notification instance as specified by a SETDATA method call.

See Also

[Notifications](#)

[Getting Started with AL](#)

[Developing Extensions](#)

NotificationScope Option Type

3/31/2019 • 2 minutes to read

Specifies the context in which the notification appears in the client.

Members

MEMBER	DESCRIPTION
GlobalScope	The notifications are not directly related to the user's current task. Note: GlobalScope is currently not supported, so do not use this value.
LocalScope	The notification appears in context of the user's current task, on the page the user is currently working on. This is the default value.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

ObjectType Option Type

3/31/2019 • 2 minutes to read

The different types of objects.

Members

MEMBER	DESCRIPTION
Codeunit	The Codeunit object type
MenuSuite	The Menusuite object type
Page	The Page object type
Query	The Query object type
Report	The Report object type
Table	The Table object type
XmlPort	The XMLPort object type

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Option Data Type

3/31/2019 • 2 minutes to read

Denotes an option value. In the code snippet below, you can see how the Option data type is declared.

```
procedure HelloWorldWithOptions(OptionParameter : Option Alpha, "Bra-vo")
var
    OptionVariable : Option C, "or D";
begin
    Message(OptionParameter::Alpha);
    Message(OptionVariable::C);
end;
```

NOTE

It is not possible to reference the members of the `OptionParameter` from outside the body of the procedure.

Remarks

In the [OptionString Property](#) of the field or variable, you can enter the option values as a comma-separated list. The Option type is a zero-based enumerator type, which means that the option values are assigned to sequential numbers, starting with 0. You can convert option data types to integers.

Example

In the Purchase Header table, the Status field is an Option data type. In the following example, the option value is converted into an integer. This example requires that you create the following variables.

NAME	DATA TYPE	SUBTYPE
Number	Integer	Not applicable
PurchHeaderRec	Record	Purchase Header

```
Number := PurchHeaderRec."Document Type";
```

Example

This example shows how you can use the value of an option field as a constant in your AL code.

```
PurchHeaderRec."Document Type" := PurchHeaderRec."Document Type"::Invoice;
```

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

OutStream Data Type

3/31/2019 • 2 minutes to read

Is a generic stream object that you can use to write to files and BLOBs.

The following methods are available on instances of the OutStream data type.

METHOD NAME	DESCRIPTION
Write(Variant, [Integer])	Writes a specified number of bytes to the stream. Data is written in binary format.
Write(Boolean, [Integer])	Writes a specified number of bytes to the stream. Data is written in binary format.
Write(Byte, [Integer])	Writes a specified number of bytes to the stream. Data is written in binary format.
Write(Char, [Integer])	Writes a specified number of bytes to the stream. Data is written in binary format.
Write(Integer, [Integer])	Writes a specified number of bytes to the stream. Data is written in binary format.
Write(BigInteger, [Integer])	Writes a specified number of bytes to the stream. Data is written in binary format.
Write(Decimal, [Integer])	Writes a specified number of bytes to the stream. Data is written in binary format.
Write(Guid, [Integer])	Writes a specified number of bytes to the stream. Data is written in binary format.
Write(Text, [Integer])	Writes a specified number of bytes to the stream. Data is written in binary format.
Write(Code, [Integer])	Writes a specified number of bytes to the stream. Data is written in binary format.
Write(Label, [Integer])	Writes a specified number of bytes to the stream. Data is written in binary format.
Write(TextConst, [Integer])	Writes a specified number of bytes to the stream. Data is written in binary format.
Write(BigText, [Integer])	Writes a specified number of bytes to the stream. Data is written in binary format.
Write(Date, [Integer])	Writes a specified number of bytes to the stream. Data is written in binary format.

METHOD NAME	DESCRIPTION
Write(Time, [Integer])	Writes a specified number of bytes to the stream. Data is written in binary format.
Write(DateTime, [Integer])	Writes a specified number of bytes to the stream. Data is written in binary format.
Write(DateFormula, [Integer])	Writes a specified number of bytes to the stream. Data is written in binary format.
Write(Duration, [Integer])	Writes a specified number of bytes to the stream. Data is written in binary format.
Write(Option, [Integer])	Writes a specified number of bytes to the stream. Data is written in binary format.
Write(Record, [Integer])	Writes a specified number of bytes to the stream. Data is written in binary format.
Write(RecordId, [Integer])	Writes a specified number of bytes to the stream. Data is written in binary format.
Write(String, [Integer])	Writes a specified number of bytes to the stream. Data is written in binary format.
Write(Any, [Integer])	Writes a specified number of bytes to the stream. Data is written in binary format.
WriteText([String], [Integer])	Writes text to an OutStream object.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

Page Data Type

3/31/2019 • 2 minutes to read

Contains a number of simpler elements called controls. Controls are used to display information to the user or receive information from the user.

The following methods are available on the Page data type.

METHOD NAME	DESCRIPTION
RunModal(Integer, [Record], [Any])	Creates, opens, and closes a page that you specify. When a page is run modally, no input, such as a keyboard or mouse click, can occur except for objects on the modal page.
RunModal(Integer, Record, Integer)	Creates, opens, and closes a page that you specify. When a page is run modally, no input, such as a keyboard or mouse click, can occur except for objects on the modal page.
RunModal(Integer, Record, FieldRef)	Creates, opens, and closes a page that you specify. When a page is run modally, no input, such as a keyboard or mouse click, can occur except for objects on the modal page.
Run(Integer, [Record], [Any])	Creates and launches a page that you specify. You can use CLEAR method to remove the page.
Run(Integer, Record, Integer)	Creates and launches a page that you specify. You can use CLEAR method to remove the page.
SetBackgroundTaskResult(Dictionary of [Text, Text])	Sets the page background task result as a dictionary. When the task is completed, the OnPageBackgroundCompleted trigger will be invoked on the page with this result dictionary.
GetBackgroundParameters()	Gets the page background task input parameters.

The following methods are available on instances of the Page data type.

METHOD NAME	DESCRIPTION
Editable([Boolean])	Gets or sets the default editability of the page.
Caption([String])	The caption shown in the title bar. For example, the default value in English (United States) is the same as the name of the page.
LookupMode([Boolean])	Gets or sets the default lookup mode for the page.
ObjectId([Boolean])	Returns a string in the "Page xxx" format, where xxx is the caption or ID of the application object.
SaveRecord()	Saves the current record as if performed by the client. If the record does not exist it is inserted, otherwise it is modified.

METHOD NAME	DESCRIPTION
Update([Boolean])	Saves the current record and then updates the controls on the page. If you set the SaveRecord parameter to false, this method will not save the record before the page is updated.
GetRecord(var Record)	Gets the current record of the page.
SetRecord(var Record)	Sets the current record for the page.
SetTableView(var Record)	Applies the table view on the current record as the table view for the page, report, or XmlPort.
SetSelectionFilter(var Record)	Notes the records that the user has selected on the page, marks those records in the table specified, and sets the filter to "marked only".
Activate([Boolean])	Activates the current page on the client if possible. The data on the page will not be refreshed.
Close()	Closes the current page.
RunModal()	Creates, opens, and closes a page that you specify. When a page is run modally, no input, such as a keyboard or mouse click, can occur except for objects on the modal page.
Run()	Creates and launches a page that you specify. You can use CLEAR method to remove the page.
EnqueueBackgroundTask(var Integer, Integer, [var Dictionary of [Text, Text]], [Boolean], [Integer])	Creates and queues a background task that runs the specified codeunit (without a UI) in a child session of the page session. If the task completes successfully, the OnPageBackgroundTaskCompleted trigger is invoked. If an error occurs, the OnPageBackgroundTaskError trigger is invoked. If the page is closed before the task completes, the task is cancelled.
CancelBackgroundTask(Integer)	Attempt to cancel a page background task.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

ProductName Data Type

3/31/2019 • 2 minutes to read

An application can have a full name, marketing name, and short name. The PRODUCTNAME functions enable you to retrieve these name variations.

The following methods are available on the ProductName data type.

METHOD NAME	DESCRIPTION
Full()	FULL returns a text string that contains the application's full name.
Short()	SHORT returns a text string that contains the application's short name.
Marketing()	MARKETING returns a text string that contains the application's marketing name.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Query Data Type

5/28/2019 • 2 minutes to read

Enables you to retrieve data from multiple tables and combine the data in single dataset.

The following methods are available on the Query data type.

METHOD NAME	DESCRIPTION
SaveAsCsv(Integer, String, [Integer], [String])	Saves the resulting data set of a query as a comma-separated values (CSV) file.
SaveAsCsv(Integer, OutStream, [Integer], [String])	Saves the resulting data set of a query as a comma separated values (CSV) file.
SaveAsXml(Integer, String)	Saves the resulting data set of a query as an .xml file.
SaveAsXml(Integer, OutStream)	Saves the resulting data set of a query as an .xml file.

The following methods are available on instances of the Query data type.

METHOD NAME	DESCRIPTION
SetFilter(Any, String, [Any,...])	Sets a filter on a column of a query to limit the records in the resulting data set of a query.
Open()	Runs a query object and generates a data set that can be read. The following code shows the syntax of the OPEN method. Query is a variable of the Query data type that specifies the query object.
Read()	Reads data from a row in the resulting data set of a query.
Close()	Closes a query data set and returns the query instance to the initialized state. The following code shows the syntax of the CLOSE method. Query is a variable of the Query data type that specifies the query object.
ColumnName(Any)	Returns the name of a query column as a text string.
ColumnCaption(Any)	Returns the current caption of a query column as a text string.
ColumnNo(Any)	Returns the ID that is assigned to a query column in the query definition.
GetFilter(Any)	Returns the filters that are set on the field of a specified column in the query. The following code shows the syntax of the GETFILTER method. Query is a variable of the Query data type that specifies the query object.

METHOD NAME	DESCRIPTION
GetFilters()	Returns the filters that are applied to all columns in the query. The following code shows the syntax of the GETFILTERS method. Query is a variable of the Query data type that specifies the query object.
SetRange(Any, [Any], [Any])	Sets a filter on a range of values on a column of a query data set.
SaveAsCsv(String, [Integer], [String])	Saves the resulting data set of a query as comma separated values (CSV)
SaveAsCsv(OutStream, [Integer], [String])	Saves the resulting data set of a query as comma separated values (CSV)
SaveAsXml(String)	Saves the resulting data set of a query as XML
SaveAsXml(OutStream)	Saves the resulting data set of a query as XML
TopNumberOfRows([Integer])	Specifies the maximum number of rows to include in the resulting data set of a query.
SecurityFiltering([SecurityFilter])	Gets or sets how security filters are applied to the query.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Record Data Type

6/25/2019 • 6 minutes to read

Is a complex data type.

The following methods are available on instances of the Record data type.

METHOD NAME	DESCRIPTION
FindFirst()	Finds the first record in a table based on the current key and filter.
FindLast()	Finds the last record in a table based on the current key and filter.
FindSet([Boolean], [Boolean])	Finds a set of records in a table based on the current key and filter.
Get([Any,...])	Gets a record based on values stored in primary key fields.
Find([String])	Finds a record in a table that is based on the values stored in keys.
Next([Integer])	Steps through a specified number of records and retrieves a record.
Reset()	Removes all filters, including any special filters set by MARKEDONLY, and changes the current key to the primary key. Also removes any marks on the record and clears any AL variables on the record.
SetCurrentKey(Any, [Any,...])	Selects a key for a table.
Ascending([Boolean])	Gets or sets the order in which the system searches through a table.
SetAscending(Any, Boolean)	Sets the sort order for the records returned. Use this method after you have set the keys to sort after, using SETCURRENTKEY. The default sort order is ascending. You can use SETASCENDING to change the sort order to descending for a specific field, while the other fields in the specified key are sorted in ascending order.
GetAscending(Any)	Gets the sort order for the records returned. You can use GETASCENDING to identify the sort order of the specified field because fields can be sorted in ascending or descending order. For example, you can read data from an ODATA web service where the data is sorted in ascending order on the Name field but in descending order on the City field.
LockTable([Boolean], [Boolean])	Locks a table to protect it from write transactions that conflict with each other.

METHOD NAME	DESCRIPTION
CalcFields(Any, [Any,...])	Calculates the FlowFields in a record. You specify which fields to calculate by using parameters.
CalcSums(Any, [Any,...])	Calculates the total of a column in a table. You specify which fields to calculate by using parameters.
SetAutoCalcFields([Any,...])	Sets the FlowFields that you specify to be automatically calculated when the record is retrieved from the database.
Count()	Counts the number of records in a table.
IsEmpty()	Determines whether a table or a filtered set of records is empty.
CountApprox()	Returns an approximate count of the number of records in the table, for example, for updating progress bars or displaying informational messages.
TableName()	Gets the name of a table.
TableCaption()	Gets the current caption of a table as a string.
ChangeCompany([String])	Redirects references to table data from one company to another.
CurrentKey()	Gets the current key of a database table.
Consistent(Boolean)	Marks a table as being consistent or inconsistent.
GetPosition([Boolean])	Gets a string that contains the primary key of the current record.
SetPosition(String)	Sets the fields in a primary key on a record to the values specified in the supplied string. The remaining fields are not changed.
Init()	Initializes a record in a table.
Insert([Boolean])	Inserts a record into a table.
Modify([Boolean])	Modifies a record in a table.
Delete([Boolean])	Deletes a record in a table.
Rename(Any, [Any,...])	Changes the value of a primary key in a table.
ModifyAll(Any, Any, [Boolean])	Modifies a field in all records within a range that you specify.
DeleteAll([Boolean])	Deletes all records in a table that fall within a specified range.

METHOD NAME	DESCRIPTION
ReadPermission()	Determines whether a user is granted read permission to the table that contains a record. This method can test for both full read permission and partial read permission that has been granted with a security filter.
WritePermission()	Determines whether a user can write to a table. This method can test for both full write permission and partial write permission that has been granted with a security filter. A write permission consists of Insert, Delete, and Modify permissions.
ReadConsistency()	Determines if the table supports read consistency.
RecordLevelLocking()	Determines whether the table supports record-level locking.
Copy(var Record, [Boolean])	Copies a specified record's filters, views, automatically calculated FlowFields, marks, fields and keys that are associated with the record from a table or creates a reference to a record.
AddLink(String, [String])	Adds a link to a record.
DeleteLink(Integer)	Deletes a specified link from a record in a table.
DeleteLinks()	Deletes all of the links that have been added to a record.
CopyLinks(var Record)	Copies all the links from a specified record.
CopyLinks(RecordRef)	Copies all the links from a specified record.
HasLinks()	Determines whether a record contains any links.
SetRange(Any, [Any], [Any])	Sets a simple filter, such as a single range or a single value, on a field.
SetFilter(Any, String, [Any,...])	Assigns a filter to a field that you specify.
GetFilter(Any)	Gets a list of the filters within the current filter group that are applied to a field.
GetFilters()	Gets a string that contains a list of the filters within the current filter group for all fields in a record. In addition, this method also returns the state of the MARKEDONLY method (Record).
GetView([Boolean])	Gets a string that describes the current sort order, key, and filters on a table.
SetView(String)	Sets the current sort order, key, and filters on a table.
GetRangeMin(Any)	Gets the minimum value in a range for a field.
GetRangeMax(Any)	Gets the maximum value in a range for a field.

METHOD NAME	DESCRIPTION
CopyFilter(Any, Any)	Copies the filter that has been set for one field and applies it to another field.
CopyFilters(var Record)	Copies all the filters set by the SETFILTER method (Record) or the SETRANGE method (Record) from one record to another.
HasFilter()	Determines whether a filter is attached to a record within the current filter group.
SetRecFilter()	Sets the values in the current key of the current record as a record filter.
FilterGroup([Integer])	Gets or sets the filter group that is applied to a table.
SetPermissionFilter()	Applies the user's security filter.
Mark([Boolean])	Marks a record. You can also use this method to determine whether a record is marked.
ClearMarks()	Removes all the marks from a record.
MarkedOnly([Boolean])	Activates a special filter. After you use this function, your view of the table includes only records marked by this function.
Validate(Any, [Any])	Calls the OnValidate trigger for the field that you specify.
TestField(Any)	Tests whether the contents of a field match a given value.
TestField(Any, Boolean)	Tests whether the contents of a field match a given value.
TestField(Any, Integer)	Tests whether the contents of a field match a given value.
TestField(Any, BigInteger)	Tests whether the contents of a field match a given value.
TestField(Any, Decimal)	Tests whether the contents of a field match a given value.
TestField(Any, Guid)	Tests whether the contents of a field match a given value.
TestField(Any, Text)	Tests whether the contents of a field match a given value.
TestField(Any, Label)	Tests whether the contents of a field match a given value.
TestField(Any, TextConst)	Tests whether the contents of a field match a given value.
TestField(Any, Code)	Tests whether the contents of a field match a given value.
TestField(Any, String)	Tests whether the contents of a field match a given value.
TestField(Any, Any)	Tests whether the contents of a field match a given value.

METHOD NAME	DESCRIPTION
FieldError(Any, [String])	Stops the execution of the code causing a run-time error, and creates an error message for a field.
TransferFields(var Record, [Boolean])	Copies all matching fields in one record to another record.
FieldName(Any)	Gets the name of a field as a string.
FieldCaption(Any)	Gets the current caption of the specified field as a string.
FieldActive(Any)	Checks whether a field is enabled.
FieldNo(Any)	Gets the number assigned to a field in the table description.
Relation(Any)	Determines the table relationship of a given field.
SecurityFiltering([SecurityFilter])	
RecordId()	
IsTemporary()	Determines whether a record refers to a temporary table.
CurrentCompany()	Gets the current company of a database table record.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

RecordId Data Type

3/31/2019 • 2 minutes to read

Contains the table number and the primary key of a table.

The following methods are available on instances of the RecordId data type.

METHOD NAME	DESCRIPTION
TableNo()	Gets the table number of the table that is identified by RecordID. This function returns an error if the record is blank.
GetRecord()	Gets a RecordRef that refers to the record identified by the RecordID.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

RecordRef Data Type

3/31/2019 • 6 minutes to read

References a record in a table.

The following methods are available on instances of the RecordRef data type.

METHOD NAME	DESCRIPTION
Open(Integer, [Boolean], [String])	Causes a RecordRef variable to refer to a table, which is identified by its number in a particular company.
Close()	Closes the current page or table.
GetTable(Record)	Gets the table of a Record variable and causes the RecordRef to refer to the same table.
SetTable(Record)	Sets the table to which a Record variable refers as the same table as a RecordRef variable.
Duplicate()	Duplicates the table that contains the RecordRef.
IsTemporary()	Determines whether a RecordRef refers to a temporary table.
CurrentCompany()	Gets the current company of a database table referred to by a RecordRef.
Get(RecordId)	Gets a record based on the ID of the record.
Find([String])	Finds a record in a table based on the values stored in the key fields.
Next([Integer])	Steps through a specified number of records and retrieves a record.
FindFirst()	Finds the first record in a table based on the current key and filter.
FindLast()	Finds the last record in a table based on the current key and filter.
FindSet([Boolean], [Boolean])	Finds a set of records in a table based on the current key and filter. FINDSET can only retrieve records in ascending order.
Reset()	Removes all filters, including any special filters set by the MARKEDONLY method (Record) and changes the current key to the primary key. Also removes any marks on the record and clears any AL variables on the record.
Ascending([Boolean])	Changes or checks the order in which a search through the table that is referred to by RecordRef will be performed.

METHOD NAME	DESCRIPTION
LockTable([Boolean], [Boolean])	Locks a table to protect it from write transactions that conflict with each other.
Count()	Counts the number of records that are in the filters that are currently applied to the table referred to by the RecordRef.
IsEmpty()	Determines whether any records exist in a filtered set of records in a table.
CountApprox()	Gets an approximate count of the number of records in the table
CurrentKey()	Gets the current key of the table referred to by the RecordRef. The current key is returned as a string.
CurrentKeyIndex([Integer])	Gets or sets the current key of the table referred to by the RecordRef. The current key is set or returned as a number. This first key = 1, and so on. If RecordRef does not have an active record, CURRENTKEYINDEX will return -1. If this value is then passed to KEYINDEX, an index out of bounds error will occur. Therefore it is important to implement a check of the RecordRef parameter.
GetPosition([Boolean])	Gets a string that contains the primary key of the current record.
SetPosition(String)	Sets the fields in a primary key on a record to the values specified in the String parameter. The remaining fields are not changed.
Number()	Gets the table ID (number) of the table that contains the record that was referred to by the RecordRef.
Name()	Identifies the name of the table
Caption()	Gets the caption of the table that is currently selected. Returns an error if no table is selected.
RecordId()	Gets the RecordID of the record that is currently selected in the table. If no table is selected, an error is generated.
ChangeCompany([String])	Redirects references to table data from one company to another.
Init()	Initializes a record in a table.
Insert([Boolean])	Inserts a record into a table.
Modify([Boolean])	Modifies a record in a table.
Delete([Boolean])	Deletes a record in a table.
DeleteAll([Boolean])	Deletes all records in a table that fall within a specified range.

METHOD NAME	DESCRIPTION
ReadPermission()	Determines if you can read from a table.
WritePermission()	Determines if you can write to a table.
ReadConsistency()	Gets a value indicating whether read consistency is enabled.
RecordLevelLocking()	Gets a value indicating whether record level locking is enabled.
AddLink(String, [String])	Adds a link to a record in a table.
DeleteLink(Integer)	Deletes a specified link from a record in a table.
DeleteLinks()	Deletes all of the links that have been added to a record.
CopyLinks(Record)	Copies all the links from a particular record.
CopyLinks(RecordRef)	Copies all the links from a particular record.
CopyLinks(Variant)	Copies all the links from a particular record.
HasLinks()	Determines whether a record contains any links.
FieldCount()	Gets the number of fields in the table that are currently selected or returns the number of fields that have been defined in a key. Returns an error if no table or no key is selected.
Field(Integer)	Gets a FieldRef for the field that has the number FieldNo in the table that is currently selected. If no field has this number, the method returns an error.
FieldExist(Integer)	Determines if the field that has the number FieldNo exists in the table that is referred to by the RecordRef. Returns an error if no table is currently selected.
FieldIndex(Integer)	Gets the FieldRef of the field that has the specified index in the table that is referred to by the RecordRef.
KeyCount()	Gets the number of keys that exist in the table that is referred to by the RecordRef. Returns an error if no table is selected.
KeyIndex(Integer)	Gets the KeyRef of the key that has the index specified in the table that is currently selected. The key can be composed of fields of any supported data type. Data types that are not supported include BLOBs, FlowFilters, variables, and functions. If the sorting key is set to a field that is not part of a key, then the KEYINDEX is -1.
GetFilters()	Determines which filters have been applied to the table referred to by the RecordRef.
GetView([Boolean])	Returns a string that describes the current sort order, key, and filters on a table.

METHOD NAME	DESCRIPTION
SetView(String)	Sets the current sort order, key, and filters on a table.
HasFilter()	Determines whether a filter has been applied to the table that the RecordRef refers to.
SetRecFilter()	Sets a filter on a record that is referred to by a RecordRef.
FilterGroup([Integer])	Changes the filter group that is being applied to the table. You can also use this method to return the number of the current filtergroup. You cannot return the number of the filtergroup and set a new filtergroup at the same time.
SetPermissionFilter()	Applies the user's security filter to the referenced record. The security filter is combined with any other filters that are placed on the record with SetFilter or SetRange. The combined filter will not include any records outside the range of the security filter and this will prevent a runtime permission error from occurring when the record is read. If the permission filter is not set, an error can occur if you attempt to read a record that is outside the range of the user's security filter.
Rename(Any, [Any,...])	Changes the value of a primary key in a table.
SecurityFiltering([SecurityFilter])	Gets or sets how security filters are applied to the RecordRef.

The RecordRef object can refer to any table in the database. Use the [OPEN method](#) to use the table number to select the table that you want to access, or use the [GETTABLE method](#) to use another record variable to select the table that you want to access.

If one RecordRef variable is assigned to another RecordRef variable, then they both refer to the same table instance.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

Report Data Type

3/31/2019 • 6 minutes to read

Is used to display, print, or process information from a database.

The following methods are available on the Report data type.

METHOD NAME	DESCRIPTION
Run(Integer, [Boolean], [Boolean], [var Record])	Loads and executes the report that you specify.
RunModal(Integer, [Boolean], [Boolean], [var Record])	Loads and executes the report that you specify.
SaveAsHtml(Integer, String, [var Record])	Saves a report as an HTML file. The file is saved on the computer where the server instance is running, and then downloaded to the client when ready. > This method is only supported when a report uses a Word report layout when it is run.
SaveAsXml(Integer, String, [var Record])	Saves the resulting data set of a query as an .xml file. The following code shows the syntax of the SAVEASXML function. The first line of code is the syntax for an instance method call. The second line of code is the syntax for a static method call.
SaveAsPdf(Integer, String, [var Record])	Saves a report as a .pdf file.
SaveAsExcel(Integer, String, [var Record])	Saves a report on the computer that is running the server as a Microsoft Excel (.xls) workbook.
SaveAsWord(Integer, String, [var Record])	Saves a report on the computer that is running the server as a Microsoft Word (.doc) document.
WordXmlPart(Integer, [Boolean])	Returns the report data structure as structured XML that is compatible with Microsoft Word custom XML parts. The method has an instance call and a static call. The following code shows the syntax of the WORDXMLPART function. The first line of code is the syntax for an instance method call. The second line of code is the syntax for a static method call.
WordLayout(Integer, InStream)	Gets the Word report layout that is used on a report and returns it as a data stream. The method has an instance call and a static call. The following code shows the syntax of the WORDLAYOUT function. The first line of code is the syntax for an instance method call. The second line of code is the syntax for a static method call.
RdlcLayout(Integer, InStream)	Gets the RDLC layout that is used on a report and returns it as a data stream. The method has an instance call and a static call. The following code shows the syntax of the RDLC function. The first line of code is the syntax for an instance method call. The second line of code is the syntax for a static method call.

METHOD NAME	DESCRIPTION
DefaultLayout(Integer)	Gets the default built-in layout type that is used on a specified report.
RunRequestPage(Integer, [String])	Runs the request page for a report without running the report. Returns an XML string that contains the request page parameters that are entered on the request page.
Execute(Integer, String, [RecordRef])	Runs a report in preview or processing-only mode without showing the request page in the client. The method gets the request page parameter values as an input parameter string from a RUNREQUESTPAGE method call. The OnOpen and OnClose triggers on the request page will run even though the request page is not shown.
Print(Integer, String, [String], [RecordRef])	Prints a specified report without running the request page. Instead of using the request page to obtain parameters at runtime, the method gets the parameter values as an input parameter string, typically from a RUNREQUESTPAGE method call.
SaveAs(Integer, String, ReportFormat, var OutStream, [RecordRef])	Runs a specific report without a request page and saves the report as a PDF, Excel, Word, HTML, or XML file. Instead of using the request page to obtain parameters at runtime, the method gets the parameter values as an input parameter string, typically from the return value of a RUNREQUESTPAGE method call.
GetSubstituteReportId(Integer)	Gets the ID of the report that will be run by the platform after considering any substitutions made by extensions.

The following methods are available on instances of the Report data type.

METHOD NAME	DESCRIPTION
Break()	Exits from a loop or a trigger in a data item trigger of a report or XmlPort.
Skip()	Skips the current iteration of the current report or XmlPort.
CreateTotals(var Decimal, [var Decimal,...])	Maintains totals for a variable in AL.
CreateTotals(Array of [Decimal])	Maintains totals for a variable in AL.
TotalsCausedBy()	Determines which field caused a group total to be calculated. This determines which field changed contents and thereby concluded a group.
WordXmlPart([Boolean])	Gets the report data structure as structured XML that is compatible with Microsoft Word custom XML parts.
ShowOutput()	Returns the current setting of whether a section should be printed, and changes this setting.
ShowOutput(Boolean)	Returns the current setting of whether a section should be printed, and changes this setting.

METHOD NAME	DESCRIPTION
PageNo([Integer])	Gets or sets the current page number of a report.
NewPage()	Forces a page break when printing a report.
Quit()	Aborts the processing of a report or XmlPort.
Preview()	Indicates whether a report is being printed in preview mode.
RunModal()	Loads and executes the report that you specify.
Run()	Loads and executes the report that you specify.
SaveAsHtml(String)	Saves a report as an HTML file. The file is saved on the computer where the server instance is running, and then downloaded to the client when ready. > This method is only supported when a report uses a Word report layout when it is run.
SaveAsXml(String)	Saves the resulting data set of a query as an .xml file.The following code shows the syntax of the SAVEASXML method. The first line of code is the syntax for an instance method call. The second line of code is the syntax for a static method call.
SaveAsPdf(String)	Saves a report as a .pdf file.
SaveAsExcel(String)	Saves a report on the computer that is running the server as a Microsoft Excel (.xls) workbook.
SaveAsWord(String)	Saves a report on the computer that is running the server as a Microsoft Word (.doc) document.
WordLayout(var InStream)	Gets the Word report layout that is used on a report and returns it as a data stream. The method has an instance call and a static call. The following code shows the syntax of the WORDLAYOUT method. The first line of code is the syntax for an instance method call. The second line of code is the syntax for a static method call.
RDLCLayout(var InStream)	Gets the RDLC layout that is used on a report and returns it as a data stream.The method has an instance call and a static call. The following code shows the syntax of the RDLC method. The first line of code is the syntax for an instance method call. The second line of code is the syntax for a static method call.
DefaultLayout()	Gets the default built-in layout type that is used on a specified report.
RunRequestPage([String])	Runs the request page for a report without running the report. Returns an XML string that contains the request page parameters that are entered on the request page.

METHOD NAME	DESCRIPTION
Execute(String, [RecordRef])	Runs a report in preview or processing-only mode without showing the request page in the client. The method gets the request page parameter values as an input parameter string from a RUNREQUESTPAGE method call. The OnOpen and OnClose triggers on the request page will run even though the request page is not shown.
Print(String, [String], [RecordRef])	Prints a specified report without running the request page. Instead of using the request page to obtain parameters at runtime, the method gets the parameter values as an input parameter string, typically from a RUNREQUESTPAGE method call.
SaveAs(String, ReportFormat, var OutStream, [RecordRef])	Runs a specific report without a request page and saves the report as a PDF, Excel, Word, or XML file. Instead of using the request page to obtain parameters at runtime, the method gets the parameter values as an input parameter string, typically from the return value of a RUNREQUESTPAGE method call.
SetTableView(var Record)	Applies the table view on the current record as the table view for the page, report, or XmlPort.
PrintOnlyIfDetail([Boolean])	Gets or sets the current settings of the PrintOnlyIfDetail property.
UseRequestPage([Boolean])	Gets or sets whether a request page is presented to the user.
NewPagePerRecord([Boolean])	Gets or sets the current setting of the NewPagePerRecord property.
Language([Integer])	Gets or sets the current language setting for the report.
ObjectId([Boolean])	Gets or sets the name or number of the report.
PaperSource(Integer, [Integer])	Gets or sets the paper source used for the current page or a specified page.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

ReportFormat Option Type

5/28/2019 • 2 minutes to read

Specifies the format of the report.

Members

MEMBER	DESCRIPTION
Excel	Saves the report as an Excel file.
Html	Saves the report in HTML format.
Pdf	Saves the report in PDF format.
Word	Saves the report in Word format.
Xml	Saves the report in XML format.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

RequestPage Data Type

5/28/2019 • 2 minutes to read

Is a page that is run before the report starts to execute. Request pages enable end-users to specify options and filters for a report.

The following methods are available on instances of the RequestPage data type.

METHOD NAME	DESCRIPTION
Editable([Boolean])	Gets or sets the default editability of the page.
Caption([String])	Shows the caption in the title bar. For example, the default value in English (United States) is the same as the name of the page.
LookupMode([Boolean])	Gets or sets the default lookup mode for the page.
ObjectId([Boolean])	Returns a string in the "Page xxx" format, where xxx is the caption or ID of the application object.
SaveRecord()	Saves the current record as if performed by the client. If the record does not exist, it is inserted, otherwise it is modified.
Update([Boolean])	Saves the current record and then updates the controls on the page. If you set the SaveRecord parameter to false, this method will not save the record before the page is updated.
SetSelectionFilter(var Record)	
Activate([Boolean])	Activates the current page on the client if possible. The data on the page will not be refreshed.
Close()	Closes the current page.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

SecurityFilter Option Type

3/31/2019 • 2 minutes to read

Specifies how security filters are applied to the record.

Members

MEMBER	DESCRIPTION
Validated	All security filters are applied to this instance of the record and if any code tries to access a record that is outside the range of the security filters, then an error occurs.
Filtered	All security filters are applied to this instance of the record.
Ignored	All security filters are ignored for this instance of the record.
Disallowed	Security filters are not allowed on the record. If any security filters are set, then you receive an error when you run the object that uses this instance of the record.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Session Data Type

3/31/2019 • 2 minutes to read

Represents a Microsoft Dynamics Business Central session.

The following methods are available on the Session data type.

METHOD NAME	DESCRIPTION
ApplicationArea([String])	Gets or sets the application areas for the current session.
StartSession(var Integer, Integer, [String], [var Record])	Starts a session without a UI and runs the specified codeunit.
IsSessionActive(Integer)	Tests if the specified SessionID is active on the server instance where it was started.
CurrentExecutionMode()	Specifies the mode in which the session is running.
StopSession(Integer, [String])	Stops a session.
CurrentClientType()	Gets the client type that is running in current session.
DefaultClientType()	Gets the default client that is configured for the server instance that is used by the current session.
BindSubscription(Codeunit)	Binds the event subscriber methods in the codeunit to the current codeunit instance for handling the events that they subscribe to. This essentially activates the subscriber functions for the codeunit instance.
UnbindSubscription(Codeunit)	Unbinds the event subscriber methods from in the codeunit instance. This essentially deactivates the subscriber methods for the codeunit instance.
ApplicationIdentifier()	Gets the application ID associated with the current thread.
SendTraceTag(String, String, Verbosity, String, [DataClassification])	Send a trace tag to the telemetry service.
GetExecutionContext()	Gets the current session's execution context.
GetModuleExecutionContext([Guid])	Gets the current session's execution context scoped to a specific module.
GetCurrentModuleExecutionContext()	Gets the current session's execution context for the currently executing module.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

SessionSettings Data Type

3/31/2019 • 2 minutes to read

Is a complex data type for passing user personalization settings for a client session as an object. The object contains properties that correspond to the fields in the system table **2000000073 User Personalization**, including: App ID, Company, Language ID, Locale ID, Profile ID, Scope, and Time Zone. You can use the AL methods of the SessionSettings data type to get, set, and send the user personalization settings for the current client session.

The following methods are available on instances of the SessionSettings data type.

METHOD NAME	DESCRIPTION
Company([String])	Gets or sets the company property in a SessionSettings object.
ProfileId([String])	Gets or sets the profile ID property in a SessionSettings object.
ProfileAppId([Guid])	Gets or sets the ID of an extension, which provides a profile, in a SessionSettings object.
ProfileSystemScope([Boolean])	Gets or sets the profile scope property in a SessionSettings object.
LanguageId([Integer])	Gets or sets the language ID property in a SessionSettings object.
LocaleId([Integer])	Gets or sets the locale ID property in a SessionSettings object.
TimeZone([String])	Gets or sets the time zone property in a SessionSettings object.
Init()	Populates the instance of a SessionsSettings with the current client user's personalization properties (such as Profile ID and Company) that are stored in the database.
RequestSessionUpdate(Boolean)	Passes a SessionSettings object to the client to request a new session that uses the user personalization properties that are set in the object. The current client session is abandoned and a new session is started.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

String Data Type

3/31/2019 • 2 minutes to read

Denotes a sequence of characters. It can be represented by a string literal, a text value or a code value.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

System Data Type

4/10/2019 • 6 minutes to read

Is a complex data type.

The following methods are available on the System data type.

METHOD NAME	DESCRIPTION
Round(Decimal, [Decimal], [String])	Rounds the value of a numeric variable.
Abs(Decimal)	Calculates the absolute value of a number (Decimal, Integer or BigInteger). ABS always returns a positive numeric value or zero.
Power(Decimal, Decimal)	Raises a number to a power. For example, you can use this method to square the number 2 to get the result of 4.
Randomize([Integer])	Generates a set of random numbers from which the RANDOM method (Integer) will select a random number.
Random(Integer)	Returns a pseudo-random number.
Today()	Gets the current date set in the operating system.
Time()	Gets the current time from the operating system.
WorkDate([Date])	Gets and sets the work date for the current session.
CalcDate(String, [Date])	Calculates a new date that is based on a date expression and a reference date.
CalcDate(DateFormula, [Date])	Calculates a new date that is based on a date expression and a reference date.
NormalDate(Date)	Gets the regular date (instead of the closing date) for the argument Date.
ClosingDate(Date)	Gets the closing date for a Date Data Type.
Date2DMY(Date, Integer)	Gets the day, month, or year of a Date Data Type.
Date2DWY(Date, Integer)	Gets the day of the week, week number, or year of a Date Data Type.
DMY2Date(Integer, [Integer], [Integer])	Gets a Date object based on a day, month, and year.
DWY2Date(Integer, [Integer], [Integer])	Gets a Date that is based on a week day, a week, and a year.
Hyperlink(String)	Passes a URL as an argument to an Internet browser, such as Windows Internet Explorer.

METHOD NAME	DESCRIPTION
Sleep(Integer)	Returns control to the operating system for a specified time.
GuiAllowed()	Checks whether the AL code can show any information on the screen.
ArrayLen(Array of [Any], [Integer])	Returns the total number of elements in an array or the number of elements in a specific dimension.
CompressArray(Array of [String])	Moves all non-empty strings (text) in an array to the beginning of the array. The resulting StringArray has the same number of elements as the input array, but empty entries appear at the end of the array.
CopyArray(Array of [Any], Array of [Any], Integer, [Integer])	Copies one or more elements in an array to a new array.
Clear(var Array of [Any])	Clears the value of a single variable. Also, it clears all the filters that were set if the variable is a record and resets the key to the primary key and the company on a record variable.
Clear(var Any)	Clears the value of a single variable. Also, it clears all the filters that were set if the variable is a record and resets the key to the primary key and the company on a record variable.
ClearAll()	Clears all internal variables (except REC variables), keys, and filters in the object and in any associated objects, such as reports, pages, codeunits, and so on that contain AL code.
Evaluate(var Any, String, [Integer])	Evaluates a string representation of a value into its typical representation. The result is assigned to a variable.
Format(Any, [Integer], [Integer])	Formats a value into a string.
Format(Any, Integer, String)	Formats a value into a string.
WindowsLanguage()	Gets the current Windows language setting.
GlobalLanguage([Integer])	Gets and sets the current global language setting.
DaTi2Variant(Date, Time)	Creates a variant that contains an encapsulation of a COM VT_DATE.
Variant2Date(Variant)	Gets a date from a variant.
Variant2Time(Variant)	Gets a time from a variant.
CopyStream(OutStream, InStream, [Integer])	Copies the information that is contained in an InStream to an OutStream.
CreateGuid()	Creates a new unique GUID. The value can then be assigned to a GUID data type or a text data type. Use the text data type if you want to compare the GUID to another text string.
CurrentDateTime()	Gets the current DateTime.

METHOD NAME	DESCRIPTION
IsNullGuid(Guid)	Indicates whether a value has been assigned to a GUID. A null GUID that consists only of zeros is valid but must never be used for references.
RoundDateTime(DateTime, [BigInteger], [String])	Rounds a DateTime.
CreateDateTime(Date, Time)	Creates a DateTime object from a date and a time.
DT2Date(DateTime)	Gets the date part of a DateTime object.
DT2Time(DateTime)	Gets the time part of a DateTime object.
GetLastErrorText()	Gets the last error that occurred in the debugger.
GetLastErrorCode()	Gets the classification of the last error that occurred.
GetLastErrorObject()	Gets the last System.Exception object that occurred.
ClearLastError()	Removes the last error message from memory.
ApplicationPath()	Returns the path of the directory where the executable file for the product is installed.
TemporaryPath()	Gets the path of the directory where the temporary file is stored.
IsServiceTier()	Gets a value indicating whether the runtime is a service tier.
ExportObjects(String, var Record, [Integer])	Exports application objects to a file.
ImportObjects(String, [Integer])	Imports application objects from a file.
IsNull(DotNet)	Gets a value indicating whether a DotNet object has been created or not.
GetLastErrorCallStack()	Gets the call stack from where the last error occurred.
CodeCoverageLog([Boolean], [Boolean])	Starts and stops the logging of code. You can also use this method to retrieve the current logging status.
CodeCoverageInclude(var Record)	Includes the code that has been logged.
CodeCoverageRefresh()	Refreshes the code that has been logged.
CodeCoverageLoad()	Loads the code that has been logged.
GetDotNetType(Any)	Gets the System.Type that corresponds to the given value.
CanLoadType(DotNet)	Tests if the specified .NET Framework type can be loaded.

METHOD NAME	DESCRIPTION
CaptionClassTranslate(String)	Returns a translated version of the caption string. The string is translated to the current local language.
GetUrl(ClientType, [String], [ObjectType], [Integer], [Record], [Boolean])	Generates a URL for the specified client target that is based on the configuration of the server instance. If the code runs in a multitenant deployment architecture, the generated URL will automatically apply to the tenant ID of the current user.
GetUrl(ClientType, String, ObjectType, Integer, RecordRef, [Boolean])	Generates a URL for the specified client target that is based on the configuration of the server instance. If the code runs in a multitenant deployment architecture, the generated URL will automatically apply to the tenant ID of the current user.
Encrypt(String)	Takes a string as input and returns the encrypted value of the string.
Decrypt(String)	Takes a string as input and returns the decrypted value of the string.
ExportEncryptionKey(String)	Returns a password protected temporary filepath containing the encryption key. When encrypting or decrypting data in Dynamics 365 Business Central, an encryption key is used. A single key is used per tenant and every tenant will have a different key. Keys can be exported to a file which may be necessary in the case of upgrading or migrating a system from one set of hardware to another. The EXPORTENCRYPTIONKEY method allows an administrator to specify a destination file for the key and specify a password protection for the file.
ImportEncryptionKey(String, String)	Points to a password protected file that contains the key on the current server. When encrypting or decrypting data in Dynamics 365 Business Central, an encryption key is used. A single key is used per tenant, and every tenant will have a different key. Keys can be created or imported if one exists already, as may be the case if upgrading or migrating a system from one set of hardware to another. The IMPORTENCRYPTIONKEY method allows an administrator to specify a file (password protected) which contains a key and imports it to the current Dynamics 365 Business Central service.
CreateEncryptionKey()	Creates an encryption key for the current tenant.
DeleteEncryptionKey()	Deletes an encryption key for the current tenant.
EncryptionKeyExists()	Checks whether an encryption key for the current tenant is present on the server tenant.
EncryptionEnabled()	Checks if the tenant is configured to allow encryption.
GetDocumentUrl(Guid)	Gets the URL for the specified temporary media object ID.
ImportStreamWithUrlAccess(InStream, String, [Integer])	Imports an object into a media container to be used in a temporary URL with a default expiration time.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

TableConnectionType Option Type

3/31/2019 • 2 minutes to read

Use variables of this data type to specify the type of connection to an external database.

Members

MEMBER	DESCRIPTION
CRM	Specifies the table as an integration table for integrating Dynamics 365 Business Central with Dynamics 365 for Sales. The table is typically based on an entity in Dynamics 365 for Sales, such as the Accounts entity.
ExternalSQL	Specifies the table as a table or view in SQL Server that is not in the Dynamics 365 Business Central database.
Exchange	This is for internal use only.
MicrosoftGraph	This is for internal use only.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

TaskScheduler Data Type

5/24/2019 • 2 minutes to read

Is a complex data type for creating and managing tasks in the task scheduler, which runs codeunits at scheduled times.

The following methods are available on the TaskScheduler data type.

METHOD NAME	DESCRIPTION
CreateTask(Integer, Integer, [Boolean], [String], [DateTime], [RecordId])	Adds a task to ensure that a codeunit is not run before the specified time.
TaskExists(Guid)	Checks whether a specific task exists.
CancelTask(Guid)	Cancels and deletes a scheduled task that runs a specific codeunit.
SetTaskReady(Guid, [DateTime])	Sets a task that runs a codeunit to the ready state. The task will not run unless it is in the ready state.
CanCreateTask()	Checks whether it is possible to schedule tasks in this session.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

TestAction Data Type

3/31/2019 • 2 minutes to read

Represents a test action on a page.

The following methods are available on instances of the TestAction data type.

METHOD NAME	DESCRIPTION
Invoke()	Invokes an action on a test page.
Enabled()	Enables an action on a test page.
Visible()	Sets whether to display the action on a test page.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

TestField Data Type

3/31/2019 • 2 minutes to read

Represents a testable field on a page.

The following methods are available on instances of the TestField data type.

METHOD NAME	DESCRIPTION
Lookup()	Provides a lookup window for a text box on a test page.
AssistEdit()	Provides assist-edit functionality to a field on a test page.
Drilldown()	Applies drill-down capability for a field on a test page.
AssertEquals(Any)	Asserts that the value in a field on a test page equals a specified value.
SetValue(Any)	Sets a value for a field on a test page.
Invoke()	Invokes the default action on the field.
Activate()	Activates a field on a test page.
AsInteger()	Converts the value of the field on a test page to an Integer data type.
AsBoolean()	Converts the value in a field on a test page to a Boolean data type.
AsDecimal()	Converts the value in a field on a test page to a Date data type.
AsDate()	Converts the value in a field on a test page to a Date data type.
AsTime()	Converts the value in a field on a test page to a Time data type.
AsDateTime()	Converts the value in a field on a test page to a DateTime data type.
Value([String])	Gets or sets the value of this field.
ValidationErrorCount()	Gets the number of validation errors that occurred on the test page.
GetValidationError([Integer])	Gets the validation error that occurred on a test page.
Visible()	Gets the visible state for the field.

METHOD NAME	DESCRIPTION
Enabled()	Gets the enabled state for the field.
Editable()	Gets the editable state for the field.
HideValue()	Gets the hide value state for the field.
Caption()	Gets the current caption of the field as a String.
OptionCount()	Gets the number of options in a field on a test page.
GetOption([Integer])	Gets the options for a field on a test page.
ShowMandatory()	Gets the ShowMandatory state for the field.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

TestFilter Data Type

3/31/2019 • 2 minutes to read

Represents a test filter on a page.

The following methods are available on instances of the TestFilter data type.

METHOD NAME	DESCRIPTION
SetFilter(TestFilterField, String)	Applies a filter to the specified field on a test page.
GetFilter(TestFilterField)	Gets the filter that is applied to the specified field in a data set that is displayed on a test page.
SetCurrentKey(TestFilterField, [TestFilterField,...])	Sets the specified fields in a data set on a test page as the current key.
Ascending([Boolean])	Gets or sets the order in which to search through a data set on a test page.
CurrentKey()	Gets the current key of a data set that is displayed on a test page.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

TestFilterField Data Type

3/31/2019 • 2 minutes to read

Represents the type of a field filter in a test filter on a page or on a request page.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

TestPage Data Type

3/31/2019 • 2 minutes to read

Represents a variable type that can be used to test Page Application Objects.

The following methods are available on instances of the TestPage data type.

METHOD NAME	DESCRIPTION
OpenNew()	Opens a blank test page in edit mode.
OpenEdit()	Opens a test page in edit mode.
OpenView()	Opens a test page in view mode.
Close()	Closes an open test page.
New()	Sets the current row of the test page to an empty row in a data set.
GetField(Integer)	Gets a field on a test page.
ValidationErrorCount()	Gets the number of validation errors that occurred on the test page.
GetValidationError([Integer])	Gets the list of all validation error that occurred on a test page as a string.
Caption()	Gets the caption of the test page.
Trap()	Traps the next test page that is invoked and assigns it to the test page variable.
Next()	Sets the current row of the test page as the next row in the data set.
Previous()	Sets the current row of the test page as the previous row in the data set.
Prev()	Sets the current row of the test page as the previous row in the data set.
First()	Sets the current row of the test page as the first row in the data set.
Last()	Sets the current row of the test page as the last row in the data set.
IsExpanded()	Specifies if rows on a test page are expanded.
Expand(Boolean)	Expands rows on a test page.

METHOD NAME	DESCRIPTION
GoToRecord(Record)	Finds the specified record in a data set on a test page.
GoToKey([Any,...])	Finds the row in a data set on the test page that is identified by the specified values.
FindFirstField(TextField, Any)	Finds the first field in the data set that is displayed on a test page.
FindNextField(TextField, Any)	Finds the next field in the data set that is displayed on a test page.
FindPreviousField(TextField, Any)	Finds the previous field in the data set that is displayed on a test page.
Editable()	Gets the runtime value of the Editable property on a test page.
View()	Gets the View system action.
Edit()	Gets the Edit system action.
OK()	Gets the OK system action.
Cancel()	Gets the Cancel system action.
Yes()	Gets the Yes system action.
No()	Gets the No system action.
RunPageBackgroundTask(Integer, [var Dictionary of [Text, Text]])	Runs the page background task codeunit in the current session. Note that no triggers are invoked at this point.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

TestPart Data Type

3/31/2019 • 2 minutes to read

Represents a variable type that can be used to test Page Application Objects of type Part.

The following methods are available on instances of the TestPart data type.

METHOD NAME	DESCRIPTION
New()	Sets the current row of the test page to an empty row in a data set.
ValidationErrorCount()	Gets the number of validation errors that occurred on the test page.
GetValidationError([Integer])	Gets the list of all validation error that occurred on a test page as a string.
GetField(Integer)	Gets a field on a test page.
Caption()	Gets the caption of the test page.
Next()	Sets the current row of the test page as the next row in the data set.
Previous()	Sets the current row of the test page as the previous row in the data set.
Prev()	Sets the current row of the test page as the previous row in the data set.
First()	Sets the current row of the test page as the first row in the data set.
Last()	Sets the current row of the test page as the last row in the data set.
IsExpanded()	Specifies if the current row on the test page is expanded.
Expand(Boolean)	Expands rows on a test page.
GoToRecord(Record)	Finds the specified record in a data set on a test page.
GoToKey([Any,...])	Finds the row in a data set on the test page that is identified by the specified values.
FindFirstField(TestField, Any)	Finds the first field in the data set that is displayed on a test page.
FindNextField(TestField, Any)	Finds the next field in the data set that is displayed on a test page.

METHOD NAME	DESCRIPTION
FindPreviousField(TestField, Any)	Finds the previous field in the data set that is displayed on a test page.
Editable()	Gets the runtime value of the Editable property on a test page.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

TestPermissions Option Type

3/31/2019 • 2 minutes to read

Specifies a value that can be used to determine which permission sets are used on tests that are run by test codeunits or test functions.

Members

MEMBER	DESCRIPTION
InheritFromTestCodeunit	Is only relevant for test methods; not test codeunits. It specifies that a test method uses the TestPermissions property setting of the test codeunit to which it belongs. If you use this value on a test codeunit, the property will resolve to Restrictive at runtime.
Restrictive	Does not perform any operations or have any specific behavior. Instead, you programmatically define what each value does, and the permissions sets it applies at runtime, by adding code in a test runner codeunit.
NonRestrictive	Does not perform any operations or have any specific behavior. Instead, you programmatically define what each value does, and the permissions sets it applies at runtime, by adding code in a test runner codeunit.
Disabled	Does not perform any operations or have any specific behavior. Instead, you programmatically define what each value does, and the permissions sets it applies at runtime, by adding code in a test runner codeunit.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

TestRequestPage Data Type

3/31/2019 • 3 minutes to read

Stores test request pages. A test request page part is a logical representation of a request page on a report. A test request page does not display a user interface (UI). The subtype of a test request page is the report whose request page you want to test.

The following methods are available on instances of the TestRequestPage data type.

METHOD NAME	DESCRIPTION
New()	Sets the current row of the test page to an empty row in a data set.
ValidationErrorCount()	Gets the number of validation errors that occurred on the test page.
GetValidationError([Integer])	Gets the validation error that occurred on a test page.
Caption()	Gets the caption of the test page.
Next()	Sets the current row of the test page as the next row in the data set.
Previous()	Sets the current row of the test page as the previous row in the data set.
First()	Sets the current row of the test page as the first row in the data set.
Last()	Sets the current row of the test page as the last row in the data set.
IsExpanded()	Specifies if rows on a test page are expanded.
Expand(Boolean)	Expands rows on a test page.
GoToRecord(Record)	Finds the specified record in a data set on a test page. The record is searched from the beginning of the dataset defined by the current filter. The search is performed by iterating across the rows, comparing the primary key with the primary key fields of the record. For large dataset, use SetFilter to limit the dataset.
GoToKey([Any,...])	Finds the row in a data set on the test page that is identified by the specified values. The key is searched from the beginning of the dataset defined by the current filter. The search is performed by iterating across the rows, comparing the primary key with the primary key fields of the record. For large dataset, use SetFilter to limit the dataset.

METHOD NAME	DESCRIPTION
FindFirstField(TestField, Any)	Finds the first field in the data set that is displayed on a test page. The row is searched from the beginning of the dataset defined by the current filter. The search is performed by iterating across the rows, comparing the primary key with the primary key fields of the record. For large dataset, use SetFilter to limit the dataset.
FindNextField(TestField, Any)	Finds the next field in the data set that is displayed on a test page. The row is searched from the beginning of the dataset defined by the current filter. The search is performed by iterating across the rows, comparing the primary key with the primary key fields of the record. For large dataset, use SetFilter to limit the dataset.
FindPreviousField(TestField, Any)	Finds the previous field in the data set that is displayed on a test page. The row is searched from the beginning of the dataset defined by the current filter. The search is performed by iterating across the rows, comparing the primary key with the primary key fields of the record. For large dataset, use SetFilter to limit the dataset.
Editable()	Gets the runtime value of the Editable property on a test page.
SaveAsPdf(String)	Saves a report as an Adobe Acrobat (.pdf) file.
SaveAsWord(String)	Saves a report as a Microsoft Word (.doc) file.
SaveAsExcel(String)	Saves a report as a Microsoft Excel (.xls) file.
SaveAsXml(String, String)	Saves a report data set and the labels on a report as two XML (.xml) files.
OK()	Gets a TestAction representing an action on the Page under Test.
Cancel()	Gets a TestAction representing an action on the Page under Test.
Print()	Gets a the Print representing an action on the Page under Test.
Preview()	Gets a TestAction representing an action on the Page under Test.
Schedule()	Gets a TestAction representing an action on the Page under Test.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

Text Data Type

3/31/2019 • 4 minutes to read

Denotes a text string.

The following methods are available on the Text data type.

METHOD NAME	DESCRIPTION
StrSubstNo(String, [Any,...])	Replaces %1, %2, %3... and #1, #2, #3... fields in a string with the values you provide as optional parameters.
StrPos(String, String)	Searches for the first occurrence of substring inside a string.
StrLen(String)	Gets the length of a string you define.
IncStr(String)	Increases a positive number or decrease a negative number inside a string by one (1).
CopyStr(String, Integer, [Integer])	Copies a substring of any length from a specific position in a string (text or code) to a new string.
MaxStrLen(String)	Gets the maximum defined length of a string variable.
MaxStrLen(Variant)	Gets the maximum defined length of a variant variable. Calling this method always results in a run-time exception.
PadStr(String, Integer, [String])	Changes the length of a string to a specified length. If the string is shorter than the specified length, length spaces are added at the end of the string to match the length. If the string is longer than the specified length, the string is truncated. If the specified length is less than 0, an exception is thrown.
DelChr(String, [String], [String])	Deletes chars contained in the which parameter in a string based on the contents on the where parameter. If the where parameter contains an equal-sign, then all occurrences of characters in which is deleted from the current value. If the where parameter contains a less-than, then the characters are only deleted when they are first in the string. If the where parameter contains a greater-than, then the characters are only deleted when they are the last in the string. If the where parameter contains any other char, an exception is thrown. If the where parameter or the which parameter is empty, the source is returned unmodified. The which parameter is to be considered as an array of chars to delete where the order does not matter.

METHOD NAME	DESCRIPTION
StrChecksum(String, [String], [Integer])	Calculates a checksum for a string that contains a number. If the source is empty, 0 is returned. Each char in the source and in the weight must be a numeric character 0-9, otherwise an exception is thrown. If the WeightString parameter is shorter than the source, it is padded with '1' up until the length of source. If the WeightString parameter is longer than the source, an exception is thrown.
ConvertStr(String, String, String)	Replaces all chars in source found in FromCharacters with the corresponding char in ToCharacters and returns the converted string. If the length of the FromCharacters parameter and the ToChars parameter are different, an exception is thrown. If the parameter FromCharacters or the parameter ToChars is empty, the source is returned unmodified. Each element in source is only converted ONCE a double-replacement cannot happen.
LowerCase(String)	Converts all letters in a string to lowercase.
UpperCase(String)	Converts all letters in a string to uppercase.
SelectStr(Integer, String)	Retrieves a substring from a comma-separated string.
DelStr(String, Integer, [Integer])	Deletes a substring inside a string (text or code).
InsStr(String, String, Integer)	Inserts a substring into a string.

The following methods are available on instances of the Text data type.

METHOD NAME	DESCRIPTION
Contains(Text)	Returns a value indicating whether a specified substring occurs within this string.
EndsWith(Text)	Determines whether the end of this string instance matches the specified string.
IndexOf(Text, [Integer])	Reports the one-based index of the first occurrence of the specified string in this instance.
IndexOfAny(Text, [Integer])	Reports the one-based index of the first occurrence of the specified string in this instance. The search starts at a specified character position.
IndexOfAny(List of [Char], [Integer])	Reports the one-based index of the first occurrence in this instance of any character in a specified array of Unicode characters. The search starts at a specified character position.
LastIndexOf(Text, [Integer])	Reports the one-based index position of the last occurrence of a specified string in this instance.
PadLeft(Integer, [Char])	Returns a new Text that right-aligns the characters in this instance by padding them on the left, for a specified total length.

METHOD NAME	DESCRIPTION
PadRight(Integer, [Char])	Returns a new string that left-aligns the characters in this string by padding them with spaces on the right, for a specified total length.
Remove(Integer, [Integer])	Returns a new Text in which a specified number of characters from the current string are deleted.
Replace(Text, Text)	Returns a new Text in which all occurrences of a specified string in the current instance are replaced with another specified string.
StartsWith(Text)	Determines whether the beginning of this instance matches a specified string.
Split([Text],...)	Splits a string into a maximum number of substrings based on a collection of separators.
Split(List of [Text])	Splits a string into a maximum number of substrings based on a collection of separators.
Split(List of [Char])	Splits a string into a maximum number of substrings based on a collection of separators.
Substring(Integer, [Integer])	Retrieves a substring from this instance.
ToLower()	Returns a copy of this string converted to lowercase.
ToUpper()	Returns a copy of this string converted to uppercase.
Trim()	Returns a new Text in which all leading and trailing white-space characters from the current Text object are removed.
TrimStart([Text])	Removes all leading occurrences of a set of characters specified in an array from the current Text object.
TrimEnd([Text])	Removes all trailing occurrences of a set of characters specified in an array from the current Text object.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

StringBuilder Data Type

3/31/2019 • 2 minutes to read

Represents a lightweight wrapper for the .Net implementation of StringBuilder.

The following methods are available on instances of the StringBuilder data type.

METHOD NAME	DESCRIPTION
Append(Text)	Appends a copy of the specified string to this StringBuilder instance.
AppendLine([Text])	Appends a copy of the specified string followed by the default line terminator to the end of the current StringBuilder object. If this parameter is omitted, only the line terminator will be appended.
Capacity([Integer])	Gets or sets the maximum number of characters that can be contained in the memory allocated by the current instance.
Clear()	Removes all characters from the current StringBuilder instance.
EnsureCapacity(Integer)	Ensures that the capacity of this StringBuilder instance is at least the specified value.
Insert(Integer, Text)	Inserts a string into this StringBuilder instance at the specified character position.
Length([Integer])	Gets or sets the length of this StringBuilder instance.
MaxCapacity()	Gets the maximum capacity of this StringBuilder instance.
Remove(Integer, Integer)	Removes the specified range of characters from this StringBuilder instance.
Replace(Text, Text)	Replaces all occurrences of a specified string in this StringBuilder instance with another specified string.
Replace(Text, Text, Integer, Integer)	Replaces, within a substring of this instance, all occurrences of a specified string in this StringBuilder instance with another specified string.
ToText()	Converts the value of this StringBuilder instance to a Text.
ToText(Integer, Integer)	Converts the value of a substring of this StringBuilder instance to a Text.

NOTE

For performance reasons all HTTP, JSON, StringBuilder, and XML types are reference types, not value types. Reference types holds a pointer to the data elsewhere in memory, whereas value types store its own data.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

TextConst Data Type

4/8/2019 • 2 minutes to read

Denotes a multi-language string constant.

Remarks

The `TextConst` data type is typically used for UI messages; process or error messages. Keeping the `TextConst` data type in global scope, makes it easier to reuse the same message for several situations. For information about naming, see [CodeCop Rule AA0074](#).

IMPORTANT

The `TextConst` data type is not included in the .xlf files for translation. Make sure to use the [Label Data Type](#) instead.

The data type can be declared with the syntax as shown below:

```
codeunit 50100 MyCodeunit
{
    procedure MyProcedure()
    var
        localTextConst: TextConst ENU = 'My text', DAN = 'Min tekst';
    begin
        Message(localTextConst);
    end;

    var
        globalTextConst: TextConst ENU = 'My text', DAN = 'Min tekst';
}
```

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

TextEncoding Option Type

3/31/2019 • 2 minutes to read

Represents a file encoding.

Members

MEMBER	DESCRIPTION
MSDos	MSDos encoding.
UTF8	UTF8 encoding.
UTF16	UTF16 encoding.
Windows	Windows encoding.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Time Data Type

4/24/2019 • 2 minutes to read

Denotes a time ranging from 00:00:00.000 to 23:59:59.999. An undefined or blank time is specified by 0T.

The displayed text format of the time is determined by your Regional and Language Options in Windows.

The following are examples of valid assignments of times to a Time variable *MyTime*. Time must be set by specifying hours, minutes, and seconds.

```
MyTime := 0T;  
MyTime := 115900T;  
MESSAGE(FORMAT(MyTime));  
MyTime := 115934T;  
MESSAGE(FORMAT(MyTime));  
MyTime := 115934.444T;  
MESSAGE(FORMAT(MyTime));  
MyTime := 235900T;  
MESSAGE(FORMAT(MyTime));  
MyTime := 030000T;  
MESSAGE(FORMAT(MyTime));
```

The following shows what the message windows display on a computer with the regional format set to English (United States).

11:59:00 AM

11:59:34 AM

11:59:34.444 AM

11:59:00 PM

3:00:00 AM

SQL Server

Microsoft SQL Server stores information about both date and time in columns of the DATETIME type. Dynamics 365 uses only the time part and inserts a constant value for the date: 01-01-1754.

The Dynamics 365 undefined time is represented by the same value as an undefined date. The undefined date is represented by the earliest valid DATETIME in SQL Server, which is 01-01-1753 00:00:00.000.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

TransactionModel Option Type

3/31/2019 • 2 minutes to read

Represents a test transaction model.

Members

MEMBER	DESCRIPTION
AutoCommit	The transaction automatically commits after the Test method has run.
AutoRollback	The transaction is automatically rolled back after the Test method has run.
None	No write-transaction is open in the test-method code, and writes will fail. The transaction model mirrors the model used by the "real" client. Every call from the TestPage to the "server" has its own transaction.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

TransactionType Option Type

3/31/2019 • 2 minutes to read

Represents a transaction type.

Members

MEMBER	DESCRIPTION
UpdateNoLocks	This is an update transaction. Modifications can occur within the transaction. All read operations are performed with READ UNCOMMITTED locking until the table is either modified by a write operation or locked with the LOCKTABLE Method (Record). From this point until the end of the transaction, all read operations are performed with UPDLOCK locking. This transaction type improves concurrency for all tables that users access within the transaction by delaying locking as much as it can. However, the disadvantage is that you must know when to lock the tables for the required transaction behavior.
Update	This is an update transaction. Modifications can occur within the transaction. All read operations are performed with REPEATABLE READ locking until the table is either modified by any write operation or locked with the LOCKTABLE method. From this point forward, all read operations are performed with UPDLOCK locking. This transaction type provides full transaction isolation from the start of the transaction, regardless of the lock status of tables that users access within the transaction.
Snapshot	This is a read-only transaction. Modifications cannot occur within the transaction. All read operations are performed with REPEATABLE READ locking. Therefore, shared locks are added on all data and are maintained until the end of the transaction. This prevents other transactions from modifying any rows that have been read by the current transaction.
Browse	This is a read-only transaction. Modifications cannot occur within the transaction. All read operations are performed with READ UNCOMMITTED locking. Therefore, no locks are added and locks that are added by other sessions are not honored. This means that the transaction may read uncommitted data.
Report	Report option maps to one of the basic options. This enables a report to use the most concurrent read-only form of data access for the connected server. When you use Dynamics 365 Business Central database server, it maps to Snapshot and when you use SQL Server, it maps to Browse.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Variant Data Type

3/31/2019 • 3 minutes to read

Represents an AL variable object. The AL variant data type can contain many AL data types.

The following methods are available on instances of the Variant data type.

METHOD NAME	DESCRIPTION
IsRecord()	Indicates whether an AL variant contains a Record variable.
IsFile()	Indicates whether an AL variant contains a File variable.
IsAction()	Indicates whether an AL variant contains an Action variable.
IsCodeunit()	Indicates whether an AL variant contains a Codeunit variable.
IsAutomation()	Indicates whether an AL variant contains an Automation variable.
IsBoolean()	Indicates whether an AL variant contains a Boolean variable.
IsOption()	Indicates whether an AL variant contains an Option variable.
IsInteger()	Indicates whether an AL variant contains an Integer variable.
IsDecimal()	Indicates whether an AL variant contains a Decimal variable.
IsChar()	Indicates whether an AL variant contains a Char variable.
IsText()	Indicates whether an AL variant contains a Text variable.
IsCode()	Indicates whether an AL variant contains a Code variable.
IsDate()	Indicates whether an AL variant contains a Date variable.
IsTime()	Indicates whether an AL variant contains a Time variable.
IsBinary()	Indicates whether an AL variant contains a Binary variable.
IsDateFormula()	Indicates whether an AL variant contains a DateFormula variable.
IsTransactionType()	Indicates whether an AL variant contains a TransactionType variable.
IsInStream()	Indicates whether an AL variant contains an InStream variable.
IsOutStream()	Indicates whether an AL variant contains an OutStream variable.

METHOD NAME	DESCRIPTION
IsDotNet()	Indicates whether an AL variant contains a DotNet variable.
IsWideChar()	Indicates whether an AL variant contains a WideChar variable.
IsExecutionMode()	Indicates whether an AL variant contains an ExecutionMode variable.
IsDateTime()	Indicates whether an AL variant contains a DateTime variable.
IsGuid()	Indicates whether an AL variant contains a Guid variable.
IsRecordId()	Indicates whether an AL variant contains a RecordId variable.
IsDuration()	Indicates whether an AL variant contains a Duration variable.
IsBigInteger()	Indicates whether an AL variant contains a BigInteger variable.
IsRecordRef()	Indicates whether an AL variant contains a RecordRef variable.
IsFieldRef()	Indicates whether an AL variant contains a FieldRef variable.
IsFilterPageBuilder()	Indicates whether an AL variant contains a FilterPageBuilder variable.
IsClientType()	Indicates whether an AL variant contains a ClientType variable.
IsObjectType()	Indicates whether an AL variant contains an ObjectType variable.
IsTextEncoding()	Indicates whether an AL variant contains a TextEncoding variable.
IsReportFormat()	Indicates whether an AL variant contains a RecordFormat variable.
IsDefaultLayout()	Indicates whether an AL variant contains a DefaultLayout variable.
IsTableConnectionType()	Indicates whether an AL variant contains a TableConnectionType variable.
IsSecurityFiltering()	Indicates whether an AL variant contains a SecurityFiltering variable.
IsDataClassificationType()	Indicates whether a AL variant contains a DataClassification variable.
IsTextConstant()	Indicates whether an AL variant contains a Text constant.
IsByte()	Indicates whether an AL variant contains a Byte data type variable.

METHOD NAME	DESCRIPTION
IsNotification()	Indicates whether an AL variant contains a Notification variable.
IsTestPermissions()	Indicates whether an AL variant contains a TestPermissions variable.
IsJsonArray()	Indicates whether an AL variant contains a JsonArray variable.
IsJsonObject()	Indicates whether an AL variant contains a JsonObject variable.
IsJsonToken()	Indicates whether an AL variant contains a JsonToken variable.
IsJsonValue()	Indicates whether an AL variant contains a JsonValue variable.
IsXmlAttribute()	Indicates whether an AL variant contains an XmlAttribute variable.
IsXmlAttributeCollection()	Indicates whether an AL variant contains an XmlAttributeCollection variable.
IsXmlCDATA()	Indicates whether an AL variant contains an XmlCDATA variable.
IsXmlComment()	Indicates whether an AL variant contains an XmlComment variable.
IsXmlDeclaration()	Indicates whether an AL variant contains an XmlDeclaration variable.
IsXmlDocument()	Indicates whether an AL variant contains an XmlDocument variable.
IsXmlDocumentType()	Indicates whether an AL variant contains an XmlDocumentType variable.
IsXmlElement()	Indicates whether an AL variant contains an XmlElement variable.
IsXmlNamespaceManager()	Indicates whether an AL variant contains an XmlNamespaceManager variable.
IsXmlNameTable()	Indicates whether an AL variant contains an XmlNameTable variable.
IsXmlNode()	Indicates whether an AL variant contains an XmlNode variable.
IsXmlNodeList()	Indicates whether an AL variant contains an XmlNodeList variable.
IsXmlProcessingInstruction()	Indicates whether an AL variant contains an XmlProcessingInstruction variable.

METHOD NAME	DESCRIPTION
IsXmlReadOptions()	Indicates whether an AL variant contains an XmlReadOptions variable.
IsXmlText()	Indicates whether an AL variant contains an XmlText variable.
IsXmlWriteOptions()	Indicates whether an AL variant contains an XmlWriteOptions variable.
IsTextBuilder()	Indicates whether an AL variant contains a TextBuilder variable.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

Verbosity Option Type

3/31/2019 • 2 minutes to read

Represents the security level of events.

Members

MEMBER	DESCRIPTION
Critical	Identifies an abnormal exit or termination event.
Error	Identifies a severe error event.
Warning	Identifies a warning event such as an allocation failure.
Normal	Identifies a non-error event such as an entry or exit event.
Verbose	Identifies a detailed trace event.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Version Data Type

3/31/2019 • 2 minutes to read

Represents a version matching the format: Major.Minor.Build.Revision .

The following methods are available on the Version data type.

METHOD NAME	DESCRIPTION
Create(String)	Creates a version object from the provided string. The string should be in the format W.X.Y.Z, where W, X, Y and Z represent positive integers and where Y and Z are optional. If the input string is not in the expected format, an exception is thrown.
Create(Integer, Integer, [Integer], [Integer])	Creates a version object from the major, minor, build and revision numbers provided.

The following methods are available on instances of the Version data type.

METHOD NAME	DESCRIPTION
Major()	Gets the major number of the version.
Minor()	Gets the minor number of the version.
Build()	Gets the build number of the version.
Revision()	Gets the revision number from the version.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

WebServiceActionContext Data Type

4/24/2019 • 2 minutes to read

Represents an AL WebServiceActionContext.

The following methods are available on instances of the WebServiceActionContext data type.

METHOD NAME	DESCRIPTION
AddEntityKey(Integer, Any)	Add a new <fieldId, value> pair to the collection of entity keys.
SetObjectType(ObjectType)	Sets the object type.
GetObjectType()	Gets the object type.
SetObjectId(Integer)	Sets the object ID.
GetObjectId()	Gets the object ID.
SetResultCode(WebServiceActionResultCode)	Sets the web service action result status code.
GetResultCode()	Gets the web service action result status code.

See Also

[Creating and Interacting with an OData V4 Bound Action](#)

[Getting Started with AL](#)

[Developing Extensions](#)

WebServiceActionResultCode Option Type

3/31/2019 • 2 minutes to read

Represents a web service action status code.

Members

MEMBER	DESCRIPTION
None	No status code.
Get	Item read.
Created	Item created.
Updated	Item updated.
Deleted	Item deleted.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

XmlAttribute Data Type

3/31/2019 • 2 minutes to read

Represents an XML attribute.

The following methods are available on the XmlAttribute data type.

METHOD NAME	DESCRIPTION
Create(String, String)	Creates an XmlAttribute node.
Create(String, String, String)	Creates an XmlAttribute node.
CreateNamespaceDeclaration(String, String)	Creates an attribute that represents a namespace declaration.

The following methods are available on instances of the XmlAttribute data type.

METHOD NAME	DESCRIPTION
Name()	The qualified name of the attribute.
LocalName()	Gets the local name of the attribute.
NamespaceUri()	Gets the namespace URI of the attribute.
NamespacePrefix()	Gets the prefix of the attribute (if any).
IsNamespaceDeclaration()	Determines if this attribute is a namespace declaration.
Value([String])	Gets or sets the value of the attribute.
AsXmlNode()	Converts the node to an XmlNode.
GetParent(var XmlElement)	Gets the parent XmlElement of this node.
GetDocument(var XmlDocument)	Gets the XmlDocument for this node.
AddAfterSelf(Any,...)	Adds the specified content immediately after this node.
AddBeforeSelf(Any,...)	Adds the specified content immediately before this node.
ReplaceWith(Any,...)	Replaces this node with the specified content.
Remove()	Removes this node from its parent element.
WriteTo(OutputStream)	Serializes and saves the current node to the given variable.
WriteTo(XmlWriteOptions, OutputStream)	Serializes and saves the current node to the given variable.

METHOD NAME	DESCRIPTION
WriteTo(var Text)	Serializes and saves the current node to the given variable.
WriteTo(XmlWriteOptions, var Text)	Serializes and saves the current node to the given variable.
SelectSingleNode(String, var XmlNode)	Selects the first XmlNode that matches the XPath expression.
SelectSingleNode(String, XmlNamespaceManager, var XmlNode)	Selects the first XmlNode that matches the XPath expression.
SelectNodes(String, var XmlNodeList)	Selects a list of nodes matching the XPath expression.
SelectNodes(String, XmlNamespaceManager, var XmlNodeList)	Selects a list of nodes matching the XPath expression.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

XmlAttributeCollection Data Type

3/31/2019 • 2 minutes to read

Represents a collection of XML attributes.

The following methods are available on instances of the XmlAttributeCollection data type.

METHOD NAME	DESCRIPTION
Count()	Gets the number of attributes in the XmlAttributeCollection.
Get(Integer, var XmlAttribute)	Gets the specified attribute.
Get(String, var XmlAttribute)	Gets the specified attribute.
Get(String, String, var XmlAttribute)	Gets the specified attribute.
RemoveAll()	Removes all attributes from the collection.
Remove(XmlAttribute)	Removes the specified attribute from the collection.
Remove(String)	Removes the specified attribute from the collection.
Remove(String, String)	Removes the specified attribute from the collection.
Set(String, String)	Sets the value of the specified attribute or creates it if is not part of the collection.
Set(String, String, String)	Sets the value of the specified attribute or creates it if is not part of the collection.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

XmlCData Data Type

3/31/2019 • 2 minutes to read

Represents a CData section.

The following methods are available on the XmlCData data type.

METHOD NAME	DESCRIPTION
Create(String)	Creates an XmlCData node.

The following methods are available on instances of the XmlCData data type.

METHOD NAME	DESCRIPTION
Value([String])	Gets or sets the value of this node.
AsXmlNode()	Converts the node to an XmlNode.
GetParent(var XmlElement)	Gets the parent XmlElement of this node.
GetDocument(var XmlDocument)	Gets the XmlDocument for this node.
AddAfterSelf(Any,...)	Adds the specified content immediately after this node.
AddBeforeSelf(Any,...)	Adds the specified content immediately before this node.
ReplaceWith(Any,...)	Replaces this node with the specified content.
Remove()	Removes this node from its parent element.
WriteTo(OutputStream)	Serializes and saves the current node to the given variable.
WriteTo(XmlWriteOptions, OutputStream)	Serializes and saves the current node to the given variable.
WriteTo(var Text)	Serializes and saves the current node to the given variable.
WriteTo(XmlWriteOptions, var Text)	Serializes and saves the current node to the given variable.
SelectSingleNode(String, var XmlNode)	Selects the first XmlNode that matches the XPath expression.
SelectSingleNode(String, XmlNamespaceManager, var XmlNode)	Selects the first XmlNode that matches the XPath expression.
SelectNodes(String, var XmlNodeList)	Selects a list of nodes matching the XPath expression.
SelectNodes(String, XmlNamespaceManager, var XmlNodeList)	Selects a list of nodes matching the XPath expression.

See Also

XmlComment Data Type

3/31/2019 • 2 minutes to read

Represents an XML comment.

The following methods are available on the XmlComment data type.

METHOD NAME	DESCRIPTION
Create(String)	Creates an XmlComment node.

The following methods are available on instances of the XmlComment data type.

METHOD NAME	DESCRIPTION
Value([String])	Gets or sets the string value of this comment.
AsXmlNode()	Converts the node to an XmlNode.
GetParent(var XmlElement)	Gets the parent XmlElement of this node.
GetDocument(var XmlDocument)	Gets the XmlDocument for this node.
AddAfterSelf(Any,...)	Adds the specified content immediately after this node.
AddBeforeSelf(Any,...)	Adds the specified content immediately before this node.
ReplaceWith(Any,...)	Replaces this node with the specified content.
Remove()	Removes this node from its parent element.
WriteTo(OutputStream)	Serializes and saves the current node to the given variable.
WriteTo(XmlWriteOptions, OutputStream)	Serializes and saves the current node to the given variable.
WriteTo(var Text)	Serializes and saves the current node to the given variable.
WriteTo(XmlWriteOptions, var Text)	Serializes and saves the current node to the given variable.
SelectSingleNode(String, var XmlNode)	Selects the first XmlNode that matches the XPath expression.
SelectSingleNode(String, XmlNamespaceManager, var XmlNode)	Selects the first XmlNode that matches the XPath expression.
SelectNodes(String, var XmlNodeList)	Selects a list of nodes matching the XPath expression.
SelectNodes(String, XmlNamespaceManager, var XmlNodeList)	Selects a list of nodes matching the XPath expression.

See Also

XmlDeclaration Data Type

3/31/2019 • 2 minutes to read

Represents an XML declaration.

The following methods are available on the XmlDeclaration data type.

METHOD NAME	DESCRIPTION
Create(String, String, String)	Creates an XmlDeclaration node.

The following methods are available on instances of the XmlDeclaration data type.

METHOD NAME	DESCRIPTION
Encoding([String])	Gets or sets the encoding of the XML document.
Standalone([String])	Gets or sets the standalone property for this document.
Version([String])	Gets or sets the version property for this document.
AsXmlNode()	Converts the node to an XmlNode.
GetParent(var XmlElement)	Gets the parent XmlElement of this node.
GetDocument(var XmlDocument)	Gets the XmlDocument for this node.
AddAfterSelf(Any,...)	Adds the specified content immediately after this node.
AddBeforeSelf(Any,...)	Adds the specified content immediately before this node.
ReplaceWith(Any,...)	Replaces this node with the specified content.
Remove()	Removes this node from its parent element.
WriteTo(OutputStream)	Serializes and saves the current node to the given variable.
WriteTo(XmlWriteOptions, OutputStream)	Serializes and saves the current node to the given variable.
WriteTo(var Text)	Serializes and saves the current node to the given variable.
WriteTo(XmlWriteOptions, var Text)	Serializes and saves the current node to the given variable.
SelectSingleNode(String, var XmlNode)	Selects the first XmlNode that matches the XPath expression.
SelectSingleNode(String, XmlNamespaceManager, var XmlNode)	Selects the first XmlNode that matches the XPath expression.
SelectNodes(String, var XmlNodeList)	Selects a list of nodes matching the XPath expression.

METHOD NAME	DESCRIPTION
SelectNodes(String, XmlNamespaceManager, var XmlNodeList)	Selects a list of nodes matching the XPath expression.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

XmlDocument Data Type

3/31/2019 • 2 minutes to read

Represents an XML document.

The following methods are available on the XmlDocument data type.

METHOD NAME	DESCRIPTION
ReadFrom(String, var XmlDocument)	Reads and parses the XML document from the given data source.
ReadFrom(String, XmlReadOptions, var XmlDocument)	Reads and parses the XML document from the given data source.
ReadFrom(InStream, var XmlDocument)	Reads and parses the XML document from the given data source.
ReadFrom(InStream, XmlReadOptions, var XmlDocument)	Reads and parses the XML document from the given data source.
Create()	Creates an XmlDocument.
Create(Any,...)	Creates an XmlDocument.

The following methods are available on instances of the XmlDocument data type.

METHOD NAME	DESCRIPTION
GetRoot(var XmlElement)	Gets the root element of the XML tree for this document.
GetDeclaration(var XmlDeclaration)	Gets the XML declaration for this document.
SetDeclaration(XmlDeclaration)	Sets the XML declaration for this document.
GetDocumentType(var XmlDocumentType)	Gets the Document Type Definition (DTD) for this document.
NameTable()	Gets the XmlNameTable associated with this document.
AsXmlNode()	Converts the node to an XmlNode.
GetParent(var XmlElement)	Gets the parent XmlElement of this node.
GetDocument(var XmlDocument)	Gets the XmlDocument for this node.
AddAfterSelf(Any,...)	Adds the specified content immediately after this node.
AddBeforeSelf(Any,...)	Adds the specified content immediately before this node.
ReplaceWith(Any,...)	Replaces this node with the specified content.

METHOD NAME	DESCRIPTION
<code>Remove()</code>	Removes this node from its parent element.
<code>WriteTo(OutputStream)</code>	Serializes and saves the current node to the given variable.
<code>WriteTo(XmlWriteOptions, OutputStream)</code>	Serializes and saves the current node to the given variable.
<code>WriteTo(var Text)</code>	Serializes and saves the current node to the given variable.
<code>WriteTo(XmlWriteOptions, var Text)</code>	Serializes and saves the current node to the given variable.
<code>SelectSingleNode(String, var XmlNode)</code>	Selects the first XmlNode that matches the XPath expression.
<code>SelectSingleNode(String, XmlNamespaceManager, var XmlNode)</code>	Selects the first XmlNode that matches the XPath expression.
<code>SelectNodes(String, var XmlNodeList)</code>	Selects a list of nodes matching the XPath expression.
<code>SelectNodes(String, XmlNamespaceManager, var XmlNodeList)</code>	Selects a list of nodes matching the XPath expression.
<code>Add(Any,...)</code>	Adds the specified content as a child of this document.
<code>AddFirst(Any,...)</code>	Adds the specified content at the start of the child list of this document.
<code>ReplaceNodes(Any,...)</code>	Replaces the children nodes of this document with the specified content.
<code>RemoveNodes()</code>	Removes the child nodes from this document.
<code>GetChildNodes()</code>	Gets a list containing the child elements for this document, in document order.
<code>GetChildElements()</code>	Gets a list containing the child elements for this document, in document order.
<code>GetChildElements(String)</code>	Gets a list containing the child elements for this document, in document order.
<code>GetChildElements(String, String)</code>	Gets a list containing the child elements for this document, in document order.
<code>GetDescendantNodes()</code>	Gets a list containing the descendant nodes for this document, in document order.
<code>GetDescendantElements()</code>	Gets a list containing the descendant elements for this document, in document order.
<code>GetDescendantElements(String)</code>	Gets a list containing the descendant elements for this document, in document order.
<code>GetDescendantElements(String, String)</code>	Gets a list containing the descendant elements for this document, in document order.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

XmlDocumentType Data Type

3/31/2019 • 2 minutes to read

Represents an XML document type.

The following methods are available on the XmlDocumentType data type.

METHOD NAME	DESCRIPTION
Create(String)	Creates an XmlDocumentType node.
Create(String, String)	Creates an XmlDocumentType node.
Create(String, String, String)	Creates an XmlDocumentType node.
Create(String, String, String, String)	Creates an XmlDocumentType node.

The following methods are available on instances of the XmlDocumentType data type.

METHOD NAME	DESCRIPTION
GetName(var Text)	Gets the name for this Document Type Definition (DTD).
SetName(String)	Sets the name for this Document Type Definition (DTD).
GetSystemId(var Text)	Gets the system identifier for this Document Type Definition (DTD).
SetSystemId(String)	Sets the system identifier for this Document Type Definition (DTD).
GetInternalSubset(var Text)	Gets the internal subset for this Document Type Definition (DTD).
SetInternalSubset(String)	Sets the internal subset for this Document Type Definition (DTD).
GetPublicId(var Text)	Gets the public identifier for this Document Type Definition (DTD).
SetPublicId(String)	Sets the public identifier for this Document Type Definition (DTD).
AsXmlNode()	Converts the node to an XmlNode.
GetParent(var XmlElement)	Gets the parent XmlElement of this node.
GetDocument(var XmlDocument)	Gets the XmlDocument for this node.
AddAfterSelf(Any,...)	Adds the specified content immediately after this node.

METHOD NAME	DESCRIPTION
AddBeforeSelf(Any,...)	Adds the specified content immediately before this node.
ReplaceWith(Any,...)	Replaces this node with the specified content.
Remove()	Removes this node from its parent element.
WriteTo(OutputStream)	Serializes and saves the current node to the given variable.
WriteTo(XmlWriteOptions, OutputStream)	Serializes and saves the current node to the given variable.
WriteTo(var Text)	Serializes and saves the current node to the given variable.
WriteTo(XmlWriteOptions, var Text)	Serializes and saves the current node to the given variable.
SelectSingleNode(String, var XmlNode)	Selects the first XmlNode that matches the XPath expression.
SelectSingleNode(String, XmlNamespaceManager, var XmlNode)	Selects the first XmlNode that matches the XPath expression.
SelectNodes(String, var XmlNodeList)	Selects a list of nodes matching the XPath expression.
SelectNodes(String, XmlNamespaceManager, var XmlNodeList)	Selects a list of nodes matching the XPath expression.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

XmlElement Data Type

3/31/2019 • 2 minutes to read

Represents an XML element.

The following methods are available on the XmlElement data type.

METHOD NAME	DESCRIPTION
Create(String)	Creates an XmlElement node.
Create(String, String)	Creates an XmlElement node.
Create(String, String, Any,...)	Creates an XmlElement node.
Create(String, Any,...)	Creates an XmlElement node.

The following methods are available on instances of the XmlElement data type.

METHOD NAME	DESCRIPTION
HasAttributes()	Gets a boolean value indicating whether this element has at least one attribute.
HasElements()	Gets a value indicating whether this element has at least one child element.
IsEmpty()	Gets a value indicating whether this element contains no content.
Name()	Gets the fully qualified name of this element.
LocalName()	Gets the local name of this element.
NamespaceUri()	Gets the namespace URI of this element.
InnerXml()	Gets the markup representing only the child nodes of this node.
InnerText()	Gets the concatenated values of the node and all its child nodes.
GetNamespaceOfPrefix(String, var Text)	Gets the namespace associated with a particular prefix for this element.
GetPrefixOfNamespace(String, var Text)	Gets the prefix associated with a namespace URI for this element.
RemoveAllAttributes()	Removes the attributes of this element.
RemoveAttribute(String)	Removes the specified attribute from this element.

METHOD NAME	DESCRIPTION
RemoveAttribute(String, String)	Removes the specified attribute from this element.
RemoveAttribute(XmlAttribute)	Removes the specified attribute from this element.
SetAttribute(String, String)	Sets the value of the specified attribute or create it if is not part of the element's attribute collection.
SetAttribute(String, String, String)	Sets the value of the specified attribute or create it if is not part of the element's attribute collection.
Attributes()	Gets a collection of the attributes of this element.
AsXmlNode()	Converts the node to an XmlNode.
GetParent(var XmlElement)	Gets the parent XmlElement of this node.
GetDocument(var XmlDocument)	Gets the XmlDocument for this node.
AddAfterSelf(Any,...)	Adds the specified content immediately after this node.
AddBeforeSelf(Any,...)	Adds the specified content immediately before this node.
ReplaceWith(Any,...)	Replaces this node with the specified content.
Remove()	Removes this node from its parent element.
WriteTo(OutputStream)	Serializes and saves the current node to the given variable.
WriteTo(XmlWriteOptions, OutputStream)	Serializes and saves the current node to the given variable.
WriteTo(var Text)	Serializes and saves the current node to the given variable.
WriteTo(XmlWriteOptions, var Text)	Serializes and saves the current node to the given variable.
SelectSingleNode(String, var XmlNode)	Selects the first XmlNode that matches the XPath expression.
SelectSingleNode(String, XmlNamespaceManager, var XmlNode)	Selects the first XmlNode that matches the XPath expression.
SelectNodes(String, var XmlNodeList)	Selects a list of nodes matching the XPath expression.
SelectNodes(String, XmlNamespaceManager, var XmlNodeList)	Selects a list of nodes matching the XPath expression.
Add(Any,...)	Adds the specified content as a child of this element.
AddFirst(Any,...)	Adds the specified content at the start of the child list of this element.
ReplaceNodes(Any,...)	Replaces the children nodes of this element with the specified content.

METHOD NAME	DESCRIPTION
RemoveNodes()	Removes the child nodes from this element.
GetChildNodes()	Gets a list containing the child elements for this element, in document order.
GetChildElements()	Gets a list containing the child elements for this element, in document order.
GetChildElements(String)	Gets a list containing the child elements for this element, in document order.
GetChildElements(String, String)	Gets a list containing the child elements for this element, in document order.
GetDescendantNodes()	Gets a list containing the descendant nodes for this element, in document order.
GetDescendantElements()	Gets a list containing the descendant elements for this element, in document order.
GetDescendantElements(String)	Gets a list containing the descendant elements for this element, in document order.
GetDescendantElements(String, String)	Gets a list containing the descendant elements for this element, in document order.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

XmlNamespaceManager Data Type

3/31/2019 • 2 minutes to read

Represents a namespace manager that can be used to resolve, add and remove namespaces to a collection. It also provides scope management for these namespaces.

The following methods are available on instances of the XmlNamespaceManager data type.

METHOD NAME	DESCRIPTION
NameTable([XmlNameTable])	Gets or sets the XmlNameTable associated with this object.
AddNamespace(String, String)	Adds the given namespace to the collection.
HasNamespace(String)	Gets a value indicating whether the supplied prefix has a namespace defined for the current scope.
LookupNamespace(String, var Text)	Gets the namespace URI for the specified prefix.
LookupPrefix(String, var Text)	Finds the prefix declared for the given namespace URI.
RemoveNamespace(String, String)	Removes the given namespace for the given prefix.
PushScope()	Pushes a namespace scope onto the stack.
PopScope()	Pops a namespace scope off the stack.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

XmlNameTable Data Type

3/31/2019 • 2 minutes to read

Represents a table of atomized string objects.

The following methods are available on instances of the XmlNameTable data type.

METHOD NAME	DESCRIPTION
Add(String)	Atomizes the specified string and adds it to the XmlNameTable.
Get(String, var Text)	Gets the atomized string with the specified value.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

XmlNode Data Type

3/31/2019 • 2 minutes to read

Represents a XML node which can either be for instance an XML attribute, an XML element or a XML document.

The following methods are available on instances of the XmlNode data type.

METHOD NAME	DESCRIPTION
IsXmlAttribute()	Gets a value indicating whether this node is an XmlAttribute.
IsXmlCData()	Gets a value indicating whether this node is an XmlCData.
IsXmlComment()	Gets a value indicating whether this node is an XmlComment.
IsXmlDeclaration()	Gets a value indicating whether this node is an XmlDeclaration.
IsXmlDocument()	Gets a value indicating whether this node is an XmlDocument.
IsXmlDocumentType()	Gets a value indicating whether this node is an XmlDocumentType.
IsXmlElement()	Gets a value indicating whether this node is an XmlElement.
IsXmlProcessingInstruction()	Gets a value indicating whether this node is an XmlProcessingInstruction.
IsXmlText()	Gets a value indicating whether this node is an XmlText.
GetParent(var XmlElement)	Gets the parent XmlElement of this node.
GetDocument(var XmlDocument)	Gets the XmlDocument for this node.
AddAfterSelf(Any,...)	Adds the specified content immediately after this node.
AddBeforeSelf(Any,...)	Adds the specified content immediately before this node.
ReplaceWith(Any,...)	Replaces this node with the specified content.
Remove()	Removes this node from its parent element.
WriteTo(OutputStream)	Serializes and saves the current node to the given variable.
WriteTo(XmlWriteOptions, OutputStream)	Serializes and saves the current node to the given variable.
WriteTo(var Text)	Serializes and saves the current node to the given variable.
WriteTo(XmlWriteOptions, var Text)	Serializes and saves the current node to the given variable.

METHOD NAME	DESCRIPTION
SelectSingleNode(String, var XmlNode)	Selects the first XmlNode that matches the XPath expression.
SelectSingleNode(String, XmlNamespaceManager, var XmlNode)	Selects the first XmlNode that matches the XPath expression.
SelectNodes(String, var XmlNodeList)	Selects a list of nodes matching the XPath expression.
SelectNodes(String, XmlNamespaceManager, var XmlNodeList)	Selects a list of nodes matching the XPath expression.
AsXmlAttribute()	Converts the node to an XmlAttribute node. The operation will fail if the node is not an XmlAttribute.
AsXmlCData()	Converts the node to an XmlCData node. The operation will fail if the node is not an XmlCData.
AsXmlComment()	Converts the node to an XmlComment node. The operation will fail if the node is not an XmlComment.
AsXmlDeclaration()	Converts the node to an XmlDeclaration node. The operation will fail if the node is not an XmlDeclaration.
AsXmlDocument()	Converts the node to an XmlDocument node. The operation will fail if the node is not an XmlDocument.
AsXmlDocumentType()	Converts the node to an XmlDocumentType node. The operation will fail if the node is not an XmlDocumentType.
AsXmlElement()	Converts the node to an XmlElement node. The operation will fail if the node is not an XmlElement.
AsXmlProcessingInstruction()	Converts the node to an XmlProcessingInstruction node. The operation will fail if the node is not an XmlProcessingInstruction.
AsXmlText()	Converts the node to an XmlText node. The operation will fail if the node is not an XmlText.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

XmlNodeList Data Type

3/31/2019 • 2 minutes to read

Represents a collection of XML nodes.

The following methods are available on instances of the XmlNodeList data type.

METHOD NAME	DESCRIPTION
Count()	Gets the number of nodes in the XmlNodeList.
Get(Integer, var XmlNode)	Gets a node at the given index.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Xmlport Data Type

3/31/2019 • 2 minutes to read

XmlPorts are used to export or import data between an external source and a Microsoft Dynamics Business Central database.

The following methods are available on the Xmlport data type.

METHOD NAME	DESCRIPTION
Import(Integer, var InStream, [var Record])	Reads and parses an incoming XML data stream (XML document).
Export(Integer, var OutStream, [var Record])	Creates an XML data stream (XML document) and sends it to a chosen destination.
Run(Integer, [Boolean], [Boolean], [var Record])	Loads and executes the XmlPort that you specify.

The following methods are available on instances of the Xmlport data type.

METHOD NAME	DESCRIPTION
CurrentPath()	Returns the CurrentPath for a given node, used when exporting an XmlPort.
Export()	Creates an XML data stream (XML document) and sends it to a chosen destination.
Import()	Reads and parses an incoming XML data stream (XML document).
Run()	Loads and executes the XmlPort that you specify.
FieldDelimiter([String])	Gets and sets the FiledDelimiter used when running, importing or exporting the XmlPort.
FieldSeparator([String])	Gets and sets the FieldSeparator used when running, importing or exporting the XmlPort.
RecordSeparator([String])	Gets and sets the RecordSeparator used when running, importing or exporting the XmlPort.
TableSeparator([String])	Gets and sets the TableSeparator used when running, importing or exporting the XmlPort.
Filename([String])	Gets the current value of the FileName Property of an XmlPort and sets this property to a new value.
TextEncoding([TextEncoding])	Gets and sets the TextEncoding used when running, importing or exporting the XmlPort.

METHOD NAME	DESCRIPTION
SetSource(var InStream)	Sets the source InStream of the XmlPort.
SetDestination(var OutStream)	Sets the destination OutStream of the XmlPort.
ImportFile([Boolean])	Gets or sets the ImportFile property.
SetTableView(var Record)	Applies the table view on the current record as the table view for the page, report, or XmlPort.
Break()	Exits from a loop or a trigger in a data item trigger of a report or XmlPort.
BreakUnbound()	Exits from a loop on records in an XmlPort trigger.
Skip()	Skips the current iteration of the current report or XmlPort.
Quit()	Aborts the processing of a report or XmlPort.

See Also

[Getting Started with AL](#)
[Developing Extensions](#)

XmlProcessingInstruction Data Type

3/31/2019 • 2 minutes to read

Represents a processing instruction, which XML defines to keep processor-specific information in the text of the document.

The following methods are available on the XmlProcessingInstruction data type.

METHOD NAME	DESCRIPTION
Create(String, String)	Creates an XmlProcessingInstruction node.

The following methods are available on instances of the XmlProcessingInstruction data type.

METHOD NAME	DESCRIPTION
GetData(var Text)	Gets the content of the processing instruction, excluding the target.
SetData(String)	Sets the content of the processing instruction, excluding the target.
GetTarget(var Text)	Gets the target of the processing instruction.
SetTarget(String)	Sets the target of the processing instruction.
AsXmlNode()	Converts the node to an XmlNode.
GetParent(var XmlElement)	Gets the parent XmlElement of this node.
GetDocument(var XmlDocument)	Gets the XmlDocument for this node.
AddAfterSelf(Any,...)	Adds the specified content immediately after this node.
AddBeforeSelf(Any,...)	Adds the specified content immediately before this node.
ReplaceWith(Any,...)	Replaces this node with the specified content.
Remove()	Removes this node from its parent element.
WriteTo(OutputStream)	Serializes and saves the current node to the given variable.
WriteTo(XmlWriteOptions, OutputStream)	Serializes and saves the current node to the given variable.
WriteTo(var Text)	Serializes and saves the current node to the given variable.
WriteTo(XmlWriteOptions, var Text)	Serializes and saves the current node to the given variable.
SelectSingleNode(String, var XmlNode)	Selects the first XmlNode that matches the XPath expression.

METHOD NAME	DESCRIPTION
SelectSingleNode(String, XmlNamespaceManager, var XmlNode)	Selects the first XmlNode that matches the XPath expression.
SelectNodes(String, var XmlNodeList)	Selects a list of nodes matching the XPath expression.
SelectNodes(String, XmlNamespaceManager, var XmlNodeList)	Selects a list of nodes matching the XPath expression.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

XmlReadOptions Data Type

3/31/2019 • 2 minutes to read

Represents the options configuring how XML is loaded from a data source.

The following methods are available on instances of the XmlReadOptions data type.

METHOD NAME	DESCRIPTION
PreserveWhitespace([Boolean])	Gets or sets a value that indicates whether insignificant white space should be preserved during parsing.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

XmlText Data Type

3/31/2019 • 2 minutes to read

Represents the text content of an element or attribute.

The following methods are available on the XmlText data type.

METHOD NAME	DESCRIPTION
Create(String)	Creates an XmlText node.

The following methods are available on instances of the XmlText data type.

METHOD NAME	DESCRIPTION
Value([String])	Gets or sets the value of this node.
AsXmlNode()	Converts the node to an XmlNode.
GetParent(var XmlElement)	Gets the parent XmlElement of this node.
GetDocument(var XmlDocument)	Gets the XmlDocument for this node.
AddAfterSelf(Any,...)	Adds the specified content immediately after this node.
AddBeforeSelf(Any,...)	Adds the specified content immediately before this node.
ReplaceWith(Any,...)	Replaces this node with the specified content.
Remove()	Removes this node from its parent element.
WriteTo(OutputStream)	Serializes and saves the current node to the given variable.
WriteTo(XmlWriteOptions, OutputStream)	Serializes and saves the current node to the given variable.
WriteTo(var Text)	Serializes and saves the current node to the given variable.
WriteTo(XmlWriteOptions, var Text)	Serializes and saves the current node to the given variable.
SelectSingleNode(String, var XmlNode)	Selects the first XmlNode that matches the XPath expression.
SelectSingleNode(String, XmlNamespaceManager, var XmlNode)	Selects the first XmlNode that matches the XPath expression.
SelectNodes(String, var XmlNodeList)	Selects a list of nodes matching the XPath expression.
SelectNodes(String, XmlNamespaceManager, var XmlNodeList)	Selects a list of nodes matching the XPath expression.

See Also

XmlWriteOptions Data Type

3/31/2019 • 2 minutes to read

Represents the options configuring how XML is saved.

The following methods are available on instances of the XmlWriteOptions data type.

METHOD NAME	DESCRIPTION
PreserveWhitespace([Boolean])	Gets or sets a value that indicates whether insignificant white space should be preserved during serialization.

See Also

[Getting Started with AL](#)

[Developing Extensions](#)

Properties Overview

5/21/2019 • 2 minutes to read

This section describes the properties that are available to developers in Dynamics 365 Business Central. Properties can be set explicitly in AL code using syntax such as:

```
Promoted = true;
```

```
PromotedCategory = Process;
```

```
ApplicationArea = All;
```

In the sections below, properties are sorted according to the object(s) they apply to.

- [Table and Table Extension Properties](#)
- [Page and Page Extension Properties- Codeunit Properties](#)
- [Query Properties](#)
- [Report Properties](#)
- [XMLPort Properties](#)
- [Control Add-In Properties](#)
- [View Properties](#)
- [Integrating with Dynamics 365 for Sales](#)

See Also

[Methods](#)

[Triggers](#)

Table and Table Extension Properties

5/21/2019 • 10 minutes to read

The following topic lists properties that apply to the [Table Object](#) and, in some cases, to the [Table Extension Object](#) as specified below.

PROPERTY NAME	AVAILABLE FOR TABLE EXTENSION OBJECT	APPLIES TO
AccessByPermission Property		<ul style="list-style-type: none">• BLOB field• BigInteger field• Boolean field• Code field• Date field• DateFormula field• DateTime field• Decimal field• Duration field• GUID field• Integer field• OemCode field• OemText field• Option field• RecordID field• TableFilter field• Text field• Time field
AutoFormatExpression Property		<ul style="list-style-type: none">• BigInteger field• Boolean field• Code field• Date field• DateFormula field• DateTime field• Decimal field• Duration field• GUID field• Integer field• OemCode field• OemText field• Option field• RecordID field• Text field• Time field

PROPERTY NAME	AVAILABLE FOR TABLE EXTENSION OBJECT	APPLIES TO
AutoFormatType Property		<ul style="list-style-type: none"> • BigInteger field • Boolean field • Code field • Date field • DateFormula field • DateTime field • Decimal field • Duration field • GUID field • Integer field • OemCode field • OemText field • Option field • RecordID field • Text field • Time field
AutoIncrement Property		<ul style="list-style-type: none"> • BigInteger field
BlankNumbers Property		<ul style="list-style-type: none"> • BigInteger field • Boolean field • Date field • DateTime field • Decimal field • Duration field • Integer field • Option field • Time field
BlankZero Property		<ul style="list-style-type: none"> • BigInteger field • Boolean field • Decimal field • Duration field • Integer field • Option

PROPERTY NAME	AVAILABLE FOR TABLE EXTENSION OBJECT	APPLIES TO
CalcFormula Property		<ul style="list-style-type: none"> • BigInteger field • Boolean field • Code field • Date field • DateTime field • Decimal field • Duration field • GUID field • Integer field • Media field • MediaSet field • Option field • RecordID field • Text field • Time field
Caption Property	X	<ul style="list-style-type: none"> • Table object • BLOB Field • BigInteger field • Boolean field • Code field • Date field • DateFormula field • DateTime field • Decimal field • Duration field • GUID field • Integer field • OemCode field • OemText field • Option field • RecordID field • TableFilter field • Text field • Time field
CaptionClass Property	X	<ul style="list-style-type: none"> • BigInteger field • Boolean field • Code field • Date field • DateFormula field • DateTime field • Decimal field • Duration field • GUID field • Integer field • OemCode field • OemText field • Option field • RecordID field • Text field • Time field

PROPERTY NAME	AVAILABLE FOR TABLE EXTENSION OBJECT	APPLIES TO
CaptionML Property	X	<ul style="list-style-type: none"> • Table object • BLOB Field • BigInteger field • Boolean field • Code field • Date field • DateFormula field • DateTime field • Decimal field • Duration field • GUID field • Integer field • OemCode field • OemText field • Option field • RecordID field • TableFilter field • Text field • Time field
CharAllowed Property		<ul style="list-style-type: none"> • Code field • OemCode field • OemText field • Text field
ClosingDates Property	X	<ul style="list-style-type: none"> • Date field
Compressed Property		<ul style="list-style-type: none"> • BLOB field
Data Type		<ul style="list-style-type: none"> • Table fields
DataClassification		<ul style="list-style-type: none"> • Table object
DataCaptionFields Property	X	<ul style="list-style-type: none"> • Table object
DataPerCompany Property		<ul style="list-style-type: none"> • Table object
DateFormula Property		<ul style="list-style-type: none"> • Code • OemCode field • OemText field • Text field

PROPERTY NAME	AVAILABLE FOR TABLE EXTENSION OBJECT	APPLIES TO
Description Property	X	<ul style="list-style-type: none"> • Table object • BLOB field • BigInteger field • Boolean field • Code field • Date field • DateFormula field • DateTime field • Decimal field • Duration field • GUID field • Integer field • OemCode field • OemText field • Option field • RecordID field • TableFilter field • Text field • Time field
DrillDownPageID Property		<ul style="list-style-type: none"> • Table object
Editable Property		<ul style="list-style-type: none"> • BigInteger field • Boolean field • Code field • Date field • DateFormula field • DateTime field • Decimal field • Duration field • GUID field • Integer field • OemCode field • OemText field • Option field • RecordID field • Text field • Time field

PROPERTY NAME	AVAILABLE FOR TABLE EXTENSION OBJECT	APPLIES TO
Enabled Property		<ul style="list-style-type: none"> • BLOB field • BigInteger field • Boolean field • Code field • Date field • DateFormula field • DateTime field • Decimal field • Duration field • GUID field • Integer field • OemCode field • OemText field • Option field • RecordID field • TableFilter field • Text field • Time field
ExtendedDataType Property		<ul style="list-style-type: none"> • BigInteger field • Boolean field • Code field • Date field • DateFormula field • DateTime field • Decimal field • Duration field • GUID field • Integer field • OemCode field • OemText field • Option field • RecordID field • Text field • Time field
ExternalName Property		<ul style="list-style-type: none"> • Table object
ExternalSchema Property		<ul style="list-style-type: none"> • Table object

PROPERTY NAME	AVAILABLE FOR TABLE EXTENSION OBJECT	APPLIES TO
FieldClass Property		<ul style="list-style-type: none"> • BigInteger field • Boolean field • Code field • Date field • DateFormula field • DateTime field • Decimal field • Duration field • GUID field • Integer field • OemCode field • OemText field • Option field • RecordID field • Text field • Time field
ID Property		<ul style="list-style-type: none"> • Table object
InitValue Property		<ul style="list-style-type: none"> • BigInteger field • Boolean field • Code field • Date field • DateFormula field • Decimal field • Duration field • GUID field • Integer field • OemCode field • OemText field • Option field • RecordID field • Text field • Time field
LinkedInTransaction Property		<ul style="list-style-type: none"> • Table object
LinkableObject Property		<ul style="list-style-type: none"> • Table object
LookupPageID Property		<ul style="list-style-type: none"> • Table object

PROPERTY NAME	AVAILABLE FOR TABLE EXTENSION OBJECT	APPLIES TO
MaxValue Property		<ul style="list-style-type: none"> • BigInteger field • Boolean field • Date field • DateTime field • Decimal field • Duration field • Integer field • Option field • Time field
MinValue Property		<ul style="list-style-type: none"> • BigInteger field • Boolean field • Date field • DateTime field • Decimal field • Duration field • Integer field • Option field • Time field
Name Property		<ul style="list-style-type: none"> • Table object
NotBlank Property		<ul style="list-style-type: none"> • BigInteger field • Boolean field • Code field • Date field • DateFormula field • DateTime field • Decimal field • Duration field • GUID field • Integer field • OemCode field • OemText field • Option field • RecordID field • Text field • Time field
Numeric Property		<ul style="list-style-type: none"> • Code field • OemCode field • OemText field • Text field
ObsoleteReason		<ul style="list-style-type: none"> • Table object • Table keys • Text field

PROPERTY NAME	AVAILABLE FOR TABLE EXTENSION OBJECT	APPLIES TO
ObsoleteState		<ul style="list-style-type: none"> • Table object • Table keys • Text field
OptionCaption Property	X	<ul style="list-style-type: none"> • Option field
OptionMembers Property		<ul style="list-style-type: none"> • Option field
PastelsValid Property		<ul style="list-style-type: none"> • Table object
Permissions Property		<ul style="list-style-type: none"> • Table object
ReplicateData Property	<ul style="list-style-type: none"> • Table object 	
SignDisplacement Property		<ul style="list-style-type: none"> • BigInteger field • Boolean field • Date field • DateTime field • Decimal field • Duration field • Integer field • Option field • Time field
SQLDataType Property		<ul style="list-style-type: none"> • Code field • OemCode field
SqlTimeStamp Property		<ul style="list-style-type: none"> • BigInteger
SubType Property (BLOB)		<ul style="list-style-type: none"> • BLOB field

PROPERTY NAME	AVAILABLE FOR TABLE EXTENSION OBJECT	APPLIES TO
TableRelation Property		<ul style="list-style-type: none"> • BigInteger field • Boolean field • Code field • Date field • DateFormula field • DateTime field • Decimal field • Duration field • GUID field • Integer field • OemCode field • OemText field • Option field • RecordID field • Text field • Time field
TableType Property		<ul style="list-style-type: none"> • Table object
ValidateTableRelation Property		<ul style="list-style-type: none"> • BigInteger field • Boolean field • Code field • Date field • DateFormula field • DateTime field • Decimal field • Duration field • GUID field • Integer field • OemCode field • OemText field • Option field • RecordID field • Text field • Time field

PROPERTY NAME	AVAILABLE FOR TABLE EXTENSION OBJECT	APPLIES TO
ValuesAllowed Property		<ul style="list-style-type: none">• BigInteger field• Boolean field• Code field• Date field• DateFormula field• DateTime field• Decimal field• Duration field• GUID field• Integer field• OemCode field• OemText field• Option field• RecordID field• Text field• Time field
Width Property	X	<ul style="list-style-type: none">• BigInteger field• Code field• Decimal field• Duration field• Integer field• OemCode field• OemText field• Text field

See Also

- [Properties](#)
- [Page and Page Extension Properties](#)

Page and Page Extension Properties Overview

5/21/2019 • 4 minutes to read

This topic lists properties that apply to the [Page Object](#) and [Page Extension Object](#).

Page object properties

The following properties all apply to the page object, only some of these properties can be set for a page extension object as specified below.

PROPERTY NAME	AVAILABLE FOR PAGE EXTENSION OBJECT	APPLIES TO
AccessByPermission Property		<ul style="list-style-type: none">• Page object• Field control• Part control• Action
AdditionalSearchTerms Property		<ul style="list-style-type: none">• Page object
AdditionalSearchTermsML Property		<ul style="list-style-type: none">• Page object
ApplicationArea Property	X	<ul style="list-style-type: none">• Page object• Field control• Part control• Action
AssistEdit Property		<ul style="list-style-type: none">• Field control
AutoFormatExpression Property		<ul style="list-style-type: none">• Field control
AutoFormatType Property		<ul style="list-style-type: none">• Field control
AutoSplitKey Property		<ul style="list-style-type: none">• Page object
BlankNumbers Property		<ul style="list-style-type: none">• Field control
BlankZero Property		<ul style="list-style-type: none">• Field control

PROPERTY NAME	AVAILABLE FOR PAGE EXTENSION OBJECT	APPLIES TO
CaptionML Property	X	<ul style="list-style-type: none"> • Page object • Container control • Group control • Field control • Part control • ActionGroup • Action • Separator
CaptionClass Property	X	<ul style="list-style-type: none"> • Field control
CharAllowed Property		<ul style="list-style-type: none"> • Field control
ClosingDates Property	X	<ul style="list-style-type: none"> • Field control
ColumnSpan Property		<ul style="list-style-type: none"> • Field control
ContextSensitiveHelpPage Property	X	<ul style="list-style-type: none"> • Page object
ContainerType Property	X	<ul style="list-style-type: none"> • Container control
DataCaptionExpression Property	X	<ul style="list-style-type: none"> • Page object
DateFormula Property		<ul style="list-style-type: none"> • Field control
DecimalPlaces Property		<ul style="list-style-type: none"> • Field control
DelayedInsert Property		<ul style="list-style-type: none"> • Page object
DeleteAllowed Property		<ul style="list-style-type: none"> • Page object
Description Property	X	<ul style="list-style-type: none"> • Page object • Container control • Group control • Field control • Part control • ActionGroup • Action
DrillDown Property		<ul style="list-style-type: none"> • Field control

PROPERTY NAME	AVAILABLE FOR PAGE EXTENSION OBJECT	APPLIES TO
DrillDownPageId Property		<ul style="list-style-type: none"> Field control
Editable Property		<ul style="list-style-type: none"> Page object Group control Field control Part control
Ellipsis Property		<ul style="list-style-type: none"> Action
Enabled Property	X	<ul style="list-style-type: none"> Group control Field control Part control ActionGroup Action
EntityName Property		<ul style="list-style-type: none"> Page object Part control
EntitySetName Property		<ul style="list-style-type: none"> Page object Part control
ExtendedDataType Property		<ul style="list-style-type: none"> Field control
FreezeColumn Property	X	<ul style="list-style-type: none"> Group control
GridLayout Property		<ul style="list-style-type: none"> Group control
Gesture Property		<ul style="list-style-type: none"> Action
HideValue Property	X	<ul style="list-style-type: none"> Field control
IndentationColumn Property		<ul style="list-style-type: none"> Group control
IndentationControls Property		<ul style="list-style-type: none"> Group control
InFooterBar Property	X	<ul style="list-style-type: none"> Action
InsertAllowed Property		<ul style="list-style-type: none"> Page object

PROPERTY NAME	AVAILABLE FOR PAGE EXTENSION OBJECT	APPLIES TO
InstructionalTextML Property	X	<ul style="list-style-type: none"> • Page object • Group control
Image Property		<ul style="list-style-type: none"> • Field control • ActionGroup • Action
Importance Property	X	<ul style="list-style-type: none"> • Field control
LinksAllowed Property		<ul style="list-style-type: none"> • Page object
LookupPageld Property		<ul style="list-style-type: none"> • Field control
Lookup Property		<ul style="list-style-type: none"> • Field control
ModifyAllowed Property		<ul style="list-style-type: none"> • Page object
MultipleNewLines Property		<ul style="list-style-type: none"> • Page object
MinValue Property		<ul style="list-style-type: none"> • Field control
MaxValue Property		<ul style="list-style-type: none"> • Field control
MultiLine Property		<ul style="list-style-type: none"> • Field control
NotBlank Property		<ul style="list-style-type: none"> • Field control
Numeric Property		<ul style="list-style-type: none"> • Field control
ODataEDMType Property	X	<ul style="list-style-type: none"> • Page object • Field control
ODataKeyFields Property		<ul style="list-style-type: none"> • Page object
OptionCaptionML Property		<ul style="list-style-type: none"> • Field control
PageType Property		<ul style="list-style-type: none"> • Page object

PROPERTY NAME	AVAILABLE FOR PAGE EXTENSION OBJECT	APPLIES TO
Permissions Property		<ul style="list-style-type: none"> Page object
PopulateAllFields Property		<ul style="list-style-type: none"> Page object
Promoted Property	X	<ul style="list-style-type: none"> Action
PromotedActionCategoriesML Property	X	<ul style="list-style-type: none"> Page object
PromotedCategory Property	X	<ul style="list-style-type: none"> Action
PromotedIsBig Property	X	<ul style="list-style-type: none"> Action
PromotedOnly Property	X	<ul style="list-style-type: none"> Action
Provider Property		<ul style="list-style-type: none"> Part control
QuickEntry Property	X	<ul style="list-style-type: none"> Field control
RefreshOnActivate Property		<ul style="list-style-type: none"> Page object
RowSpan property		<ul style="list-style-type: none"> Field control
RunObject Property		<ul style="list-style-type: none"> Action
RunPageLink Property		<ul style="list-style-type: none"> Action
RunPageMode Property		<ul style="list-style-type: none"> Action
RunPageOnRec Property		<ul style="list-style-type: none"> Action
RunPageView Property		<ul style="list-style-type: none"> Action
SaveValues Property		<ul style="list-style-type: none"> Page object
Scope Property		<ul style="list-style-type: none"> Action
ShortcutKey Property		<ul style="list-style-type: none"> Action

PROPERTY NAME	AVAILABLE FOR PAGE EXTENSION OBJECT	APPLIES TO
ShowAsTree Property		<ul style="list-style-type: none"> • Group control
ShowCaption Property	X	<ul style="list-style-type: none"> • Group control • Field control
ShowFilter Property		<ul style="list-style-type: none"> • Page object • Part control
ShowMandatory Property		<ul style="list-style-type: none"> • Field control
SignDisplacement Property		<ul style="list-style-type: none"> • Field control
SourceTable Property		<ul style="list-style-type: none"> • Page object
SourceTableTemporary Property		<ul style="list-style-type: none"> • Page object
SourceTableView Property		<ul style="list-style-type: none"> • Page object
Style Property	X	<ul style="list-style-type: none"> • Field control
StyleExpr Property	X	<ul style="list-style-type: none"> • Field control
SubPageView Property		<ul style="list-style-type: none"> • Part control
SubPageLink Property		<ul style="list-style-type: none"> • Part control
TableRelation Property		<ul style="list-style-type: none"> • Field control
ToolTipML Property	X	<ul style="list-style-type: none"> • Field control • Part control • ActionGroup • Action
UpdatePropagation Property		<ul style="list-style-type: none"> • Part control
UsageCategory Property		<ul style="list-style-type: none"> • Page object
ValuesAllowed Property		<ul style="list-style-type: none"> • Field control

PROPERTY NAME	AVAILABLE FOR PAGE EXTENSION OBJECT	APPLIES TO
Visible Property	X	<ul style="list-style-type: none">• Group control• Field control• Part control• ActionGroup• Action
Width Property	X	<ul style="list-style-type: none">• Field control

See Also

- [Properties](#)
- [Page Object](#)
- [Page Extension Object](#)
- [Report Object](#)
- [Table and Table Extension Properties](#)

Codeunit Properties

5/21/2019 • 2 minutes to read

This topic lists properties that apply to the Codeunit object, variables, text constants, methods, parameters, and return values.

PROPERTY NAME	APPLIES TO
ConstValue Property	<ul style="list-style-type: none">• Global Text Constants• AL Locals Text Constants
ConstValueML Property	<ul style="list-style-type: none">• Global Text Constants• AL Locals Text Constants
Dimensions Property	<ul style="list-style-type: none">• Global Variables• AL Locals Variables• AL Locals Parameters• AL Locals Return Values
Event Property	<ul style="list-style-type: none">• Global Methods
EventMethod Property	<ul style="list-style-type: none">• Global Methods
EventPublisherElement Property	<ul style="list-style-type: none">• Global Methods
EventPublisherObject Property	<ul style="list-style-type: none">• Global Methods
EventSubscriberInstance Property	<ul style="list-style-type: none">• Codeunit Object
EventType Property	<ul style="list-style-type: none">• Global Methods
GlobalVarAccess Property	<ul style="list-style-type: none">• Global Methods
HandlerFunctions Property	<ul style="list-style-type: none">• Global Methods
ID Property	<ul style="list-style-type: none">• Global Variables• Global Text Constants• Global Methods• AL Locals Variables• AL Locals Text Constants• AL Locals Parameters• AL Locals Return Values

PROPERTY NAME	APPLIES TO
IncludeInDataSet Property	<ul style="list-style-type: none"> Global Variables
IncludeSender Property	<ul style="list-style-type: none"> Global Methods
Local Property	<ul style="list-style-type: none"> Global Methods
MethodType Property (Upgrade Codeunits)	<ul style="list-style-type: none"> Global Methods
MethodType Property (Test Codeunits)	<ul style="list-style-type: none"> Global Methods
Name Property	<ul style="list-style-type: none"> Codeunit Object
OptionString Property	<ul style="list-style-type: none"> Global Variables AL Locals Variables
Permissions Property	<ul style="list-style-type: none"> Codeunit Object
RunOnClient Property	<ul style="list-style-type: none"> AL Locals variables
SingleInstance Property	<ul style="list-style-type: none"> Codeunit Object
SubType Property (Codeunit)	<ul style="list-style-type: none"> Codeunit Object
SuppressDispose Property	<ul style="list-style-type: none"> AL Locals variables
TableNo Property	<ul style="list-style-type: none"> Codeunit Object
Temporary Property	<ul style="list-style-type: none"> AL Locals Variables
TestIsolation Property	<ul style="list-style-type: none"> Codeunit Object
TransactionModel Property	<ul style="list-style-type: none"> Global Methods
TryMethod Property	<ul style="list-style-type: none"> Global Methods
WithEvents Property	<ul style="list-style-type: none"> Global Variables

See Also

[Developing Extensions](#)

[AL Development Environment](#)

[Table and Table Extension Properties](#)

[Page and Page Extension Properties Overview](#)

Query Properties

5/28/2019 • 2 minutes to read

This topic lists properties that apply to the query object.

PROPERTY NAME	APPLIES TO
APIPublisher Property	<ul style="list-style-type: none">• Query Object
APIVersion Property (Query)	<ul style="list-style-type: none">• Query Object
APIGroup Property	<ul style="list-style-type: none">• Query Object
Caption Property	<ul style="list-style-type: none">• Query Object• Column control• Filter control
CaptionML Property	<ul style="list-style-type: none">• Query Object• Column control• Filter control
ColumnFilter Property	<ul style="list-style-type: none">• Column control• Filter control
DataItemLink Property (Query)	<ul style="list-style-type: none">• DataItem control
DataItemLinkType Property	<ul style="list-style-type: none">• DataItem control
DataItemTable Property	<ul style="list-style-type: none">• DataItem control
DataItemTableFilter Property	<ul style="list-style-type: none">• DataItem control
DataSource Property	<ul style="list-style-type: none">• Column control• Filter control
Description Property	<ul style="list-style-type: none">• Query Object• DataItem control• Column control• Filter control
EntityName Property	<ul style="list-style-type: none">• Query Object

PROPERTY NAME	APPLIES TO
EntitySetName Property	<ul style="list-style-type: none"> • Query Object
ID Property	<ul style="list-style-type: none"> • Query Object • DataItem control • Column control • Filter control
Indentation Property (Query)	<ul style="list-style-type: none"> • DataItem control • Column control • Filter control
Method Property	<ul style="list-style-type: none"> • Column control
MethodType Property	<ul style="list-style-type: none"> • Column control
Name Property	<ul style="list-style-type: none"> • Query Object • DataItem control • Column control • Filter control
OrderBy Property	<ul style="list-style-type: none"> • Query Object
Permissions Property	<ul style="list-style-type: none"> • Query Object
QueryCategory Property	<ul style="list-style-type: none"> • Query Object
QueryType Property	<ul style="list-style-type: none"> • Query Object
ReadState Property	<ul style="list-style-type: none"> • Query Object
ReverseSign Property	<ul style="list-style-type: none"> • Column control
TopNumberOfRows Property	<ul style="list-style-type: none"> • Query Object

See Also

[Properties](#)

[Table and Table Extension Properties](#)

[Page and Page Extension Properties Overview](#)

Report Properties

6/17/2019 • 2 minutes to read

This topic lists properties of the report object.

PROPERTY NAME	APPLIES TO	
AccessByPermission Property	<ul style="list-style-type: none">Report Object	
AdditionalSearchTerms Property		<ul style="list-style-type: none">Report object
AdditionalSearchTermsML Property		<ul style="list-style-type: none">Report object
ApplicationArea Property	<ul style="list-style-type: none">Report Object	
AutoCalcField Property	<ul style="list-style-type: none">Column controls	
AutoFormatExpr Property	<ul style="list-style-type: none">Column controls	
AutoFormatType Property	<ul style="list-style-type: none">Column controls	
Caption Property	<ul style="list-style-type: none">Report ObjectReport Labels	
CaptionML Property	<ul style="list-style-type: none">Report ObjectReport Labels	
CalcFields Property	<ul style="list-style-type: none">Dataltem control	
ContextSensitiveHelpPage Property	<ul style="list-style-type: none">Report Object	
DataltemLink Property (Reports)	<ul style="list-style-type: none">Dataltem controls	
DataltemLinkReference Property	<ul style="list-style-type: none">Dataltem controls	
DataltemTable Property	<ul style="list-style-type: none">Dataltem controls	
DataltemTableView Property	<ul style="list-style-type: none">Dataltem controls	

PROPERTY NAME	APPLIES TO	
DecimalPlaces Property	<ul style="list-style-type: none"> Column controls 	
Description Property	<ul style="list-style-type: none"> Report Object Column controls Report Labels 	
EnableExternalAssemblies Property	<ul style="list-style-type: none"> Report Object 	
EnableExternalImages Property	<ul style="list-style-type: none"> Report Object 	
EnableHyperlinks Property	<ul style="list-style-type: none"> Report Object 	
ID Property	<ul style="list-style-type: none"> Report Object Dataltem controls Column controls Report Labels 	
IncludeCaption Property	<ul style="list-style-type: none"> Column controls 	
Indentation Property (Reports)	<ul style="list-style-type: none"> Dataltem controls Column controls 	
MaxIteration Property	<ul style="list-style-type: none"> Dataltem controls 	
Name Property	<ul style="list-style-type: none"> Report Object Dataltem controls Column controls Report Labels 	
OptionCaption Property	<ul style="list-style-type: none"> Column controls 	
OptionCaptionML Property	<ul style="list-style-type: none"> Column controls 	
OptionString Property	<ul style="list-style-type: none"> Column controls 	
PaperSourceDefaultPage Property	<ul style="list-style-type: none"> Report Object 	
PaperSourceFirstPage Property	<ul style="list-style-type: none"> Report Object 	
PaperSourceLastPage Property	<ul style="list-style-type: none"> Report Object 	

PROPERTY NAME	APPLIES TO	
PDFFontEmbedding Property	<ul style="list-style-type: none"> Report Object 	
Permissions Property	<ul style="list-style-type: none"> Report Object 	
PrintOnlyIfDetail Property	<ul style="list-style-type: none"> Dataltem controls 	
ProcessingOnly Property	<ul style="list-style-type: none"> Report Object 	
ReqFilterFields Property	<ul style="list-style-type: none"> Dataltem controls 	
ReqFilterHeading Property	<ul style="list-style-type: none"> Dataltem controls 	
ReqFilterHeadingML Property	<ul style="list-style-type: none"> Dataltem controls 	
ShowPrintStatus Property	<ul style="list-style-type: none"> Report Object 	
SourceExpr Property	<ul style="list-style-type: none"> Column controls 	
TransactionType Property	<ul style="list-style-type: none"> Report Object 	
UsageCategory Property	<ul style="list-style-type: none"> Report Object 	
UseRequestPage Property	<ul style="list-style-type: none"> Report Object 	
UseTemporary Property (Reports)	<ul style="list-style-type: none"> Dataltem controls 	
UseSystemPrinter Property	<ul style="list-style-type: none"> Report Object 	

See Also

[Properties](#)

[Table and Table Extension Properties](#)

[Page and Page Extension Properties Overview](#)

XMLport Properties

6/17/2019 • 3 minutes to read

This topic lists properties of the XMLport object, element, and attribute.

PROPERTY NAME	XMLPORT OBJECT
AutoCalcField Property	<ul style="list-style-type: none">• Field elements• Field attributes
AutoReplace Property	<ul style="list-style-type: none">• Table elements• Table attributes
AutoSave Property	<ul style="list-style-type: none">• Table elements• Table attributes
AutoUpdate Property	<ul style="list-style-type: none">• Table elements• Table attributes
CalcFields Property	<ul style="list-style-type: none">• Table elements• Table attributes
Caption Property	<ul style="list-style-type: none">• XMLport object
CaptionML Property	<ul style="list-style-type: none">• XMLport object
ContextSensitiveHelpPage Property	<ul style="list-style-type: none">• Report Object
CurrentPath Property	<ul style="list-style-type: none">• XMLport object
DefaultFieldsValidation Property	<ul style="list-style-type: none">• XMLport object
DefaultNamespace Property	<ul style="list-style-type: none">• XMLport object
Direction Property	<ul style="list-style-type: none">• XMLport object
Encoding Property	<ul style="list-style-type: none">• XMLport object
FieldValidate Property	<ul style="list-style-type: none">• Field elements• Field attributes

PROPERTY NAME	XMLPORT OBJECT
FileName Property	<ul style="list-style-type: none"> XMLport object
Format Property	<ul style="list-style-type: none"> XMLport object
Format-Evaluate Property	<ul style="list-style-type: none"> XMLport object
ID Property	<ul style="list-style-type: none"> XMLport object
Indentation Property (XMLports)	<ul style="list-style-type: none"> Text elements Table elements Field elements Text attributes Table attributes Field attributes
InlineSchema Property	<ul style="list-style-type: none"> XMLport object
LinkFields Property	<ul style="list-style-type: none"> Table elements Table attributes
LinkTable Property	<ul style="list-style-type: none"> Table elements Table attributes
LinkTableForceInsert Property	<ul style="list-style-type: none"> Table elements Table attributes
MaxOccurs Property	<ul style="list-style-type: none"> Text elements Table elements Field elements
MinOccurs Property	<ul style="list-style-type: none"> Text elements Table elements Field elements
Name Property	<ul style="list-style-type: none"> XMLport object
NamespacePrefix Property	<ul style="list-style-type: none"> XMLport object
Namespaces Property	<ul style="list-style-type: none"> XMLport object

PROPERTY NAME	XMLPORT OBJECT
NodeName Property	<ul style="list-style-type: none"> • Text elements • Table elements • Field elements • Text attributes • Table attributes • Field attributes
NodeType Property	<ul style="list-style-type: none"> • Text elements • Table elements • Field elements • Text attributes • Table attributes • Field attributes
Occurrence Property	<ul style="list-style-type: none"> • Text attributes • Table attributes • Field attributes
Permissions Property	<ul style="list-style-type: none"> • XMLport object
PreserveWhiteSpace Property	<ul style="list-style-type: none"> • XMLport object
ReqFilterFields Property	<ul style="list-style-type: none"> • Table elements • Table attributes
ReqFilterHeading Property	<ul style="list-style-type: none"> • Table elements • Table attributes
ReqFilterHeadingML Property	<ul style="list-style-type: none"> • Table elements • Table attributes
SourceField Property	<ul style="list-style-type: none"> • Field elements • Field attributes
SourceTable Property (XMLports)	<ul style="list-style-type: none"> • Table elements • Table attributes
SourceTableView Property (XMLports)	<ul style="list-style-type: none"> • Table elements • Table attributes

PROPERTY NAME	XMLPORT OBJECT
SourceType Property	<ul style="list-style-type: none"> • Text elements • Table elements • Field elements • Text attributes • Table attributes • Field attributes
TextEncoding Property (XMLports)	<ul style="list-style-type: none"> • XMLport object
TextType Property	<ul style="list-style-type: none"> • Text elements • Text attributes
TransactionType Property	<ul style="list-style-type: none"> • XMLport object
Unbound Property	<ul style="list-style-type: none"> • Text elements • Field elements
UseDefaultNamespace Property	<ul style="list-style-type: none"> • XMLport object
UseLax Property	<ul style="list-style-type: none"> • XMLport object
UseRequestPage Property	<ul style="list-style-type: none"> • XMLport object
UseTemporary Property (XMLports)	<ul style="list-style-type: none"> • Table elements • Table attributes
VariableName Properties	<ul style="list-style-type: none"> • Text elements • Table elements • Text attributes • Table attributes
Width Property (XMLport)	<ul style="list-style-type: none"> • Text elements • Table elements • Field elements • Text attributes • Table attributes • Field attributes

See Also

[Properties](#)

[Table and Table Extension Properties](#)

[Page and Page Extension Properties Overview](#)

Control Add-In Properties

5/28/2019 • 2 minutes to read

The following topic lists properties that apply to the [Control Add-In Object](#).

- [VerticalShrink](#)
- [HorizontalShrink](#)
- [MinimumHeight](#)
- [MinimumWidth](#)
- [MaximumHeight](#)
- [MaximumWidth](#)
- [VerticalStretch](#)
- [HorizontalStretch](#)
- [RequestedHeight](#)
- [RequestedWidth](#)

See also

[Codeunit Properties](#)

[Page Properties](#)

[Query Properties](#)

[Report Properties](#)

[Table Properties](#)

[XMLPort Properties](#)

View Properties

5/28/2019 • 2 minutes to read

The following topic lists properties that apply to [Views](#).

- [Filters](#)
- [OrderBy](#)

See also

[Codeunit Properties](#)

[Page Properties](#)

[Query Properties](#)

[Report Properties](#)

[Table Properties](#)

[XMLPort Properties](#)

Integrating Dynamics 365 for Sales for Extension Development

3/31/2019 • 2 minutes to read

Develop extensions and streamline the workflow by synchronizing the Sales data from Microsoft Dynamics 365 for Sales with Dynamics 365 Business Central.

For developing extensions to integrate with sales data, you simply enable the tables used in Dynamics 365 for Sales. The extension development process includes the following set of properties to enable field mapping. You can enable the field mapping by using the following properties.

IMPORTANT

Extending tables from Dynamics 365 for Sales is currently not supported.

Associated table and field properties

The following properties are used for integrating with Microsoft Dynamics 365 for Sales:

PROPERTIES	APPLIES TO	DESCRIPTION
TableType Property	Tables	Specifies the table type. This enables the table to integrate with the external database. For example, <code>CRM</code> .
ExternalName Property	Tables, Fields	<p>Specifies the name of the original table in the external database when used as a table property.</p> <p>Specifies the field name of the corresponding field specified in the external table when used as a field property.</p>
ExternalAccess Property	Fields	Specifies the access to the underlying CRM entity when CRM tables are generated using the cmdlet.
ExternalType Property	Fields	Specifies the data type of the corresponding field in Dynamics 365 for Sales table.
OptionMembers Property	Fields	Sets the option values for a field, text box or variable.
OptionOrdinalValues Property	Fields	Specifies the list of option values. You can set this property, if the <code>ExternalType</code> is set to <code>Picklist</code> .

Enabling the entity

Typically in Dynamics 365 for Sales, entities handle the internal processes. In order to access to the underlying CRM entity, you use the TableType property and select the value called **CRM**. This enables the table as an integration table for integrating Dynamics 365 Business Central with Dynamics 365 for Sales. The table is mainly based on an entity in Dynamics 365 for Sales, such as the Accounts entity.

Snippet support

Typing the shortcut `ttable` will create the basic layout for a table object when using the AL Language extension in Visual Studio Code.

Example

In the following example, the `SalesIntegration` table uses the TableType and ExternalName properties to link the underlying **CRM** entity for mapping the fields from the `Sales` table with the specified fields.

```
table 50100 SalesIntegration
{
    TableType = CRM;
    ExternalName = 'Sales';

    fields
    {
        field(1; ActualSales; Integer)
        {
            ExternalName = 'ActualSale';
            ExternalAccess = Full;
            ExternalType = 'String';
        }

        field(2; SalesCategories; Option)
        {
            ExternalName='SalesCategory';
            ExternalAccess = Read;
            ExternalType = 'Picklist';
            OptionMembers = Manufacturing, Marketing, Support;
            OptionOrdinalValues = -1, 1, 2;
        }
    }
}
```

See Also

[Table Properties](#)

[TableType Property](#)

Triggers Overview

4/24/2019 • 2 minutes to read

The following sections describe the triggers that are available for the different AL objects:

- [Table and Field Triggers](#)
- [Page and Action Triggers](#)
- [Codeunit Triggers](#)
- [Report and Data Item Triggers](#)
- [XMLport Triggers](#)
- [Query Triggers](#)

See Also

[AL Method Reference](#)

[Properties](#)

Table and Field Triggers

5/21/2019 • 2 minutes to read

Dynamics 365 Business Central recognizes certain actions that happen to a table when you use it, for example, when you insert or modify data. In response, you specify to execute AL code defined in a trigger. Triggers are predefined methods that are executed when certain actions happen. The bodies of these methods are initially empty and must be defined by the developer. Defining AL code in triggers allows you to change the default behavior of Dynamics 365 Business Central.

The triggers in a table can be divided into two categories:

- Table triggers
- Field triggers

Tables have the following triggers.

TABLE TRIGGER	RUNS WHEN
OnInsert Trigger	A new record is inserted into the table.
OnModify Trigger	A record in the table is modified.
OnDelete Trigger	A record in the table is deleted.
OnRename Trigger	A record is modified in a primary key field.

Fields have the following triggers.

FIELD TRIGGER	RUNS WHEN
OnValidate (Fields) Trigger	Data is entered in a field or when the VALIDATE (Record) is executed.
OnLookup (Fields) Trigger	Lookup is activated.
OnBeforeValidate (Fields) Trigger	Before data is entered in a field.
OnAfterValidate (Fields) Trigger	After data is entered in a field.

See Also

[Table Object](#)

[Table Extension Object](#)

[Triggers](#)

[Table and Table Extension Properties](#)

Page and Action Triggers

3/31/2019 • 2 minutes to read

Page triggers allow you to use AL code to control the behavior of the system as a result of an event on the page, such as a page opening or a field changing its value. You typically use page triggers for advanced validation and logic.

Page triggers can be divided into three categories:

- General page triggers that apply to the entire page
- Field page triggers that apply to a field control on a page
- Action triggers that apply to an action on a page.

IMPORTANT

If you define two methods that have the same name, one defined in a page and the other in a table that is referenced by the page, you cannot invoke the method defined in the page directly. By default, a call to the method invokes the method that is defined in the table. This behavior occurs when the method is called from a source expression or a trigger.

General Triggers

The following table lists triggers that apply to the entire page.

PAGE TRIGGER NAME	RUNS
OnInit Trigger	When the page is loaded, but before the controls are available.
OnOpenPage Trigger	When the page is initialized and the controls are available.
OnClosePage Trigger	When the page about to close and after OnQueryClosePage Trigger trigger.
OnFindRecord Trigger	When the page is opened and a record is retrieved from a table.
OnNextRecord Trigger	When the page changes from displaying one record to another record in a table. For example, on a Customer card page, this happens when a user selects Next (Ctrl+Page Down) or Previous (Ctrl+Page Up).
OnAfterGetCurrRecord Trigger	After the current record is retrieved from the table.
OnAfterGetRecord Trigger	When a record has been retrieved but not yet displayed.
OnNewRecord Trigger	When a new record has been initialized but not yet displayed.
OnInsertRecord Trigger	When a new record is about to be inserted in the table.
OnModifyRecord Trigger	When a record is about to be modified in the table.

PAGE TRIGGER NAME	RUNS
OnDeleteRecord Trigger	When a record is about to be deleted from the table.
OnQueryClosePage Trigger	When the page is about to close, but before the OnClosePage Trigger .

Field Triggers

The following table describes the triggers that are available on field controls.

CONTROL TRIGGER	RUNS
OnValidate (Page fields) Trigger	When the user changes the value in a field and then selects away from the field so that the field loses focus.
OnLookup (Page fields) Trigger	When the user requests a lookup by clicking a field's lookup button or pressing F4.
OnDrillDown Trigger	When the user requests a drill-down by choosing the field's drill-down button or pressing Shift+F8.
OnAssistEdit Trigger	When the user requests assist-edit by choosing an AssistEdit button or by pressing Shift+F4.

Action Triggers

The following table lists triggers that apply to actions on a page.

TRIGGERS	RUNS
OnAction Trigger	When an action is initiated on a page.

See Also

[Triggers](#)

Codeunit Triggers

6/17/2019 • 2 minutes to read

The following triggers apply to codeunits.

CODEUNIT TRIGGER NAME	RUNS
OnBeforeTestRun Trigger	Before a test method of a test codeunit is run.
OnAfterTestRun Trigger	After a test method of a test codeunit is run.
OnCheckPreconditionsPerCompany Trigger	Before an extension upgrade of an upgrade codeunit is run.
OnCheckPreconditionsPerDatabase Trigger	Before an extension upgrade of an upgrade codeunit is run.
OnUpgradePerCompany Trigger	When an extension upgrade of an upgrade codeunit is run.
OnUpgradePerDatabase Trigger	When an extension upgrade of an upgrade codeunit is run.
OnValidateUpgradePerCompany Trigger	After an extension upgrade of an upgrade codeunit is run.
OnValidateUpgradePerDatabase Trigger	After an extension upgrade of an upgrade codeunit is run.
OnInstallAppPerCompany Trigger	When an extension installation or reinstallation in an install codeunit is run.
OnInstallAppPerDatabase Trigger	When an extension installation or reinstallation in an install codeunit is run.

See Also

[Triggers](#)

Report and Data Item Triggers

3/31/2019 • 2 minutes to read

In reports, triggers are typically used to perform calculations and verification. Triggers let you control how data is selected and retrieved in a more complex and effective way than you can achieve by using properties.

Report Triggers

The following table lists triggers that apply to the report itself.

TRIGGER	RUNS
OnInitReport Trigger	When the report is loaded.
OnPreReport Trigger	Before the report is run, but after the RequestPage has been run.
OnPostReport Trigger	After the report has run, but not if the report was stopped manually or by the QUIT Method (Report, XMLport) .

Data Item Triggers

The following table lists triggers that apply to each data item on the report.

TRIGGER	RUNS
OnPreDataItem Trigger	Before the data item is processed, but after the associated variable has been initialized.
OnAfterGetRecord (Data Items) Trigger	When a record has been retrieved from the table.
OnPostDataItem Trigger	When the data item has been iterated for the last time.

See Also

[Report Triggers](#)
[Triggers](#)

XMLport Triggers

3/31/2019 • 2 minutes to read

The following triggers apply to XMLports.

XMLport triggers

XMLPORT TRIGGER	RUNS
OnAfterAssignField Trigger	<p>Runs after a field has been assigned a value and before it is validated and imported.</p> <p>This trigger is only used to import data.</p>
OnAfterAssignVariable Trigger	<p>Runs after the value defined in the XML document is assigned to the text variable.</p> <p>This trigger is only used to import data.</p>
OnAfterGetField Trigger	<p>Runs after a field is passed to the XML document.</p> <p>This trigger is only used to export data.</p>
OnAfterGetRecord (XMLports) Trigger	<p>Runs after a record is retrieved from a table and before it is exported to the XML document.</p> <p>This trigger is only used to export data.</p>
OnAfterInitRecord Trigger	<p>Runs after a record is loaded.</p> <p>This trigger is only used to import data.</p>
OnAfterInsertRecord Trigger	<p>Runs after a record has been inserted into a database table.</p> <p>This trigger is only used to import data.</p>
OnAfterModifyRecord Trigger	<p>Runs after a record has been modified.</p> <p>The trigger is used to import data.</p>
OnBeforeInsertRecord Trigger	<p>Runs after a record has been loaded and before it is inserted into a database table.</p> <p>This trigger is only used to import data.</p>
OnBeforeModifyRecord Trigger	<p>Runs before a record is modified.</p> <p>This trigger is used to import data.</p>
OnBeforePassField Trigger	<p>Runs before a field is passed to the XML document.</p> <p>This trigger is only used to export data.</p>

XMLPORT TRIGGER	RUNS
OnBeforePassVariable Trigger	<p>Runs after the source expression has been formatted into a text variable and before the text variable is passed to the XML document.</p> <p>This trigger is only used to export data.</p>
OnInitXMLport Trigger	<p>Executes when the XMLport is loaded and before any table views and filters are set.</p>
OnPreXMLport Trigger	<p>Runs after the table views and filters are set and before the XMLport is run.</p>
OnPostXMLport Trigger	<p>Runs after the XMLport is run.</p>
OnPreXMLItem Trigger	<p>Runs after the table is initialized and before you start exporting data to an XML object. This trigger only applies to XMLport elements that have a source type of Table.</p> <p>This trigger is only used to export data.</p>

See Also

[XMLPort Object](#)

[Triggers](#)

[XMLPort Properties](#)

Query Triggers

3/31/2019 • 2 minutes to read

This topic describes the AL triggers that are available for queries. Triggers are typically used to perform calculations and verification. Triggers let you control how data is selected and retrieved in a more complex and effective way than you can achieve by using properties.

Query object triggers

The following table lists the triggers that apply to the query object.

TRIGGER	RUNS
OnBeforeOpen	Before the query object is run and the dataset is generated. For example, you can use the OnBeforeOpen trigger to apply filters using the SETFILTER method.

See Also

[Query Object](#)

[Triggers](#)

[SETFILTER Method \(Query\)](#)

[Report Triggers](#)

Rules and Guidelines for AL Code

6/25/2019 • 4 minutes to read

This page defines the rules and guidelines to follow when writing AL code in an extension package for Dynamics 365 Business Central. The rules and guidelines are grouped according to two importance levels: critical errors that must be resolved, and important errors that should be resolved. Errors that are not resolved must include an explanation and justification for the error.

Critical errors

- Code uses encryption key functions such as `IMPORTENCRYPTIONKEY`, `EXPORTENCRYPTIONKEY`, `CREATEENCRYPTIONKEY`, and `DELETEENCRYPTIONKEY`. (It is fine to use the `ENCRYPT` and `DECRYPT` methods.)
- Code uses `ASSERTERROR`.
- External data connections do not properly handle sensitive data.
- It does not encrypt sensitive table data. (i.e. credit card info, passwords, etc.).

Important errors

- Temporary files are not cleaned up after use.
- Code uses codeunits that require printers to be selected.
- Code uses a specific time zone or locale.

Common pitfalls

To help you save time, we're sharing a list of the top 15 common pitfalls that regularly lead to app validation failures.

1. Prefix/Suffix missing

One of the app requirements is for you to reserve a prefix/suffix for your app. This is needed to ensure a healthy app ecosystem by avoiding collision amongst apps. This common failure occurs due to not setting your prefix/suffix in some or all required places. For more information, see [Benefits and Guidelines for using a Prefix or Suffix](#).

2. DataClassification missing or set incorrectly

Due to GDPR requirements, fields of field class *Normal* must use the [DataClassification property](#), and its value must be different from *ToBeClassified*. This applies to fields in tables and table extensions. Use the [AppSourceCop](#) tool for detecting this.

3. Required translation files missing

There are many countries today that where Dynamics 365 Business Central is available, and that you can support as well with your app. For specifying additional languages, we no longer support Caption ML. You must use xcliff translation files instead. For more information, see [Working with Translation Files](#).

Microsoft provides a free translation tool that you can access from <https://lcs.dynamics.com> To support a specific country, you must include a translation file per for each language code. For example, to support Switzerland, you must provide fr-CH, de-CH, and it-CH.

4. Missing permission sets

Your app must provide one or more permission sets so that users can use your app's functionality. Your app must never require the SUPER permission set.

5. Permission errors

For your app to be a good citizen in Dynamics 365 Business Central, permission errors must not appear unless it is a necessary reason for showing the error.

It is acceptable to throw an error to a user that does not have your permission set marked and tries to access your page object. It is not acceptable to throw an error to that same user trying to access Business Central pages in the base application, or to throw an error to a user who is not trying to access your app's functionality.

6. Missing application area tagging

Tag in which part your app participates. Pages, controls, actions, and fields will not appear in Dynamics 365 Business Central if the [Application Area property](#) has not been set.

7. Usage Category not set

You enable a page or report to be available through search in Dynamics 365 Business Central by using the [UsageCategory property](#). For more information, see [Using Tell Me to Find Features and Information](#).

8. OnCompany procedure

Due to their performance impact, *OnBeforeCompanyOpen* and *OnAfterCompanyOpen* cannot be used. For more information, see [Replacing OnBeforeCompanyOpen and OnAfterCompanyOpen](#).

9. Upgrade procedures

Make sure that your app can be upgraded properly. For more information, see [Upgrading extensions](#).

10. Profiles

Do not insert into the Profile table. Use the [Profile object](#) instead.

11. App file not properly code signed

Your app file must be digitally signed with a certificate from a third-party certification authority trusted by Windows.

12. You tested your app on an obsolete Dynamics 365 Business Central version (or never even tested it)

Make sure that your app is properly tested on the correct version. For more information, see [Current Build - Developing for Dynamics 365 Business Central](#) on the Collaborate site.

13. You tested using SUPER permissions

You tested your app, but your user had SUPER permissions. This can hide critical errors. You must test with a user that doesn't have the SUPER permissions. The user must have the ESSENTIAL license. For more information, see [Testing your Extension](#).

14. User scenario document unclear

Our validation team is testing your app functionality manually, so we need to be able to understand the core functionality of your app. If your user scenario document is missing important details that are needed for us to properly walk through your app's setup and usage scenarios, we cannot validate your app successfully. For more information, see [User Scenario Documentation](#).

15. The .json file is incorrect

There are many values in the app.json file that may not be mandatory to compile your app, but are mandatory for your app to be in AppSource. For example, your app cannot be published to a production

tenant if the **target** value is set to *Internal*. It must be set to *Extension*. For information, see [JSON Files](#).

See Also

[Best Practices for AL Code](#)

[Checklist for Submitting Your App](#)

Best Practices for AL

4/4/2019 • 4 minutes to read

This page defines some of the best practices to follow when writing AL code for Dynamics 365 Business Central. These best practices are additional to rules and guidelines that are caught during compilation of AL code. We recommend following these best practices when developing extensions in AL to ensure consistency and discoverability on file, object, and method naming, as well as better readability of written code.

NOTE

If a best practice is not mentioned here, the PreCal rules listed [here](#) apply.

Extension structure

An extension is fully contained in a single folder. This folder often contains multiple files, such as app.json and launch.json files, perhaps an image file representing the extension's logo, various folders for source; "\src", other resources; "\res", and a test folder; "\test" folder. The extension does not need to follow a flat structure, which means that, depending on the amount of application files, additional folders can be used in the "src" or "test" folders to group objects based on their functionality, which can help make maintaining a large .al project easier.

File naming

Each file name must start with the corresponding type and ID, followed by a dot for full objects or a dash for extensions. The name of the object is written only with characters [A-Za-z0-9] and dot al is used for the file type.

File naming notation

Follow the syntax for file naming as shown below:

FULL OBJECTS	EXTENSIONS
<code><Type><Id>.<ObjectName>.al</code>	<code><Type><BaseId>-Ext<ObjectId>.<ObjectName>.al</code>

Type map

Use the listed abbreviations for each type of object in the file naming:

OBJECT	ABBREVIATION
Page	Pag
Page Extension	PagExt
Page Customization	PagCust
Codeunit	Cod
Table	Tab
Table Extension	TabExt

OBJECT	ABBREVIATION
XML Port	Xml
Report	Rep
Query	Que
Enum	Enu
Enum Extension	EnuExt
Control Add-ins	ConAddin

Examples of file naming

The following table illustrates how the file naming should look.

OBJECT NAME	FILE NAME
codeunit 1000 "Job Calculate WIP"	Cod1000.JobCalculateWIP.al
page 21 "Customer Card"	Pag21.CustomerCard.al
page 1234 "MyPag" extends "Customer Card"	Pag21-Ext1234.MyPag.al

Formatting

We recommend keeping your AL code properly formatted as follows:

- Use all lowercase letters for reserved language keywords. Built-in methods and types are not included in this rule because they are written using Pascal case.
- Use four spaces for indentation.
- Curly brackets are always on a new line. If there is one property, put it on a single line.

The following example illustrates these formatting rules.

```

page 123 PageName
{
    actions
    {
        area(Processing)
        {
            action(ActionName)
            {
                trigger OnAction()
                begin
                    end;
            }
        }
    }

    var
        TempCustomer: Record Customer temporary;

    [EventSubscriber(ObjectType::Page, Page::"Item Card", 'OnAfterGetCurrRecordEvent', '', false, false)]
    local procedure OnOpenItemCard(var rec: Record Item)
    var
        OnRecord: Option " ", Item, Contact;
    begin
        EnablePictureAnalyzerNotification(rec."No.", OnRecord::Item);
    end;
}

```

The AL Language extension offers users the option to automatically format their source code. For more information on how to use it, see [AL Formatter](#).

Line length

In general, there is no restriction on line length, but lengthy lines can make the code unreadable. We recommend that you keep your code easily scannable and readable.

Object naming

Object names are prefixed. They start with the feature/group name, followed by the logical name as in these two examples:

- Intrastat extension validation codeunit for Denmark
- codeunit 123 "IntrastatDK Validation"

NOTE

The "MS - " prefix is not required.

File structure

Inside an .al code file, the structure for all objects must follow the sequence below:

1. Properties
2. Object-specific constructs such as:
 - Table fields
 - Page layout
 - Actions

3. Global variables

- Labels (old Text Constants)
- Global variables

4. Methods

Referencing

In AL, objects are referenced by their object name, not by their ID.

Example

```
Page.RunModal(Page::"Customer Card", ...)

var
    Customer: Record Customer;
```

Variable naming

For variables they must:

- Be named using PascalCase.
- Have the `Temp` prefix if they are temporary variables.
- Include the object name in the name (for objects).

Example

```
TempCustomer: Record Customer temporary;
Vendor: Record Vendor;
```

Method declaration

To declare a method, follow the guidelines below:

- Include a space after a semicolon when declaring multiple arguments.
- Semicolons can be used at the end of the signature/method header. If you use a snippet, the semicolons are not automatically added.
- Methods are named as variables using Pascal case. However, this is not a mandatory rule.
- There must be a blank line between method declarations. If you format your code using the [AL Formatter](#) tool, the auto-formatter sets the blank line between procedures.

Example

```
local procedure MyProcedure(Customer: Record Customer; Int: Integer)
begin
end;

// space

local procedure MyProcedure2(Customer: Record Customer; Int: Integer)
begin
end;
```

Calling methods

When calling a method, include one space after each command if you are passing multiple parameters. Parentheses must be specified when you are making a method call or system call such as: Init(), Modify(), Insert() etc.

Example

```
MyProcedure();  
MyProcedure(1);  
MyProcedure(1, 2);
```

Type definition (colon)

When declaring a variable or a parameter, the name of that variable or parameter must be immediately followed by a colon, then a single space, and then the type of the variable/parameter as illustrated in the example below.

```
Var  
    Number: Integer;  
  
local procedure MyProcedure(a: Integer; b: Integer): Integer
```

See Also

[Checklist for Submitting Your App](#)

[Rules and Guidelines for AL Code](#)

Benefits and Guidelines for using a Prefix or Suffix

3/31/2019 • 2 minutes to read

In your extension, the name of each new application object (table, page, codeunit), must contain a prefix or suffix. This rule applies to all objects. You can use the [Caption](#) values for what you decide to display to the user. When you modify a core Dynamics 365 object using a Table Extension or Page Extension, the prefix/suffix must be defined at the control/field/action/group level.

Declare your objects with a prefix as shown in the following examples.

Table

```
table 70000000 MyPrefix Salesperson
```

Page

```
page 70000000 MyPrefix Salesperson
```

Page Extension

```
actions
{
    addafter(ApprovalEntries)
    {
        action(MyPrefix Vacation)
```

Codeunit

```
codeunit 70000000 MyPrefix Salesperson
```

Benefits

There are two good reasons to why you may want to proactively use a prefix or suffix:

1. App A and App B both use the same field name (for a native Dynamics 365 table) of FAB Salesperson Code. The partner for App B already has the prefix/suffix reserved. A customer wants to install both apps but cannot due to collision of field name. App A will have to reserve a different unique prefix and submit an updated version of their app.
2. Dynamics 365 developers want to use the name of Salesperson Code. App A (published for months), already has that field name. Microsoft will require the app to prefix its field names by submitting an updated version of their app.

General rules

- The prefix/suffix must be at least 3 characters
- The object/field name must start or end with the prefix/suffix
- If a conflict arises, the one who registered the prefix/suffix always wins
- For your own pages/tables/codeunits, you must set the prefix/suffix at the top object level

- For pages/tables in the base application of BC that you extend, you must set the prefix/suffix at the top object level
- For pages/tables of BC in the base application that you extend, you must also set at the control/field/action level
- Use the [AppSourceCop](#) tool to find all missing prefixes and/or suffixes. Configuration options for this tool can be found [here](#). The Rules section explains the different checks the cop will do. For prefix/suffix detection, refer to the Configuration section. It explains how to set your prefix in the AppSourceCop.json file.

Examples of acceptable prefix/suffix

No Delimiter

- FABSalespersonCode
- SalespersonCodeFAB

Space

- "FAB Salesperson Code"
- "Salesperson Code FAB"

Underscore

- FAB_Salesperson_Code
- Salesperson_Code_FAB

Contact us at d365val@microsoft.com to reserve the prefix/suffix of your choosing

See Also

[Checklist for Submitting Your App](#)

[Rules and Guidelines for AL Code](#)

Testing your Extension

6/4/2019 • 5 minutes to read

Several key things lead to your Dynamics 365 Business Central extension passing the Microsoft validation process. However, one of the most important checks you can do is to take the time and test your extension before submitting it for validation. This allows you to catch some of the basic errors that could lead to validation failures. The following list calls out key points, and the sections below provide more context.

- Always test in a Dynamics 365 Business Central online environment. If you test in an on-premises deployment, you might miss errors that would be seen online.
- Ensure that your extension can be published without code signing errors. You **must not** use the `-skipverification` flag.
- The extension should be able to be installed without errors.
- If you are using the **Assisted Setup**, ensure that you can use your wizard to completion without errors.
- Walk through the setup and usage of your extension to verify it works as expected (**remember to test as a user that does not have SUPER permissions**).
- Check that you can uninstall and unpublish your extension without any errors.
- Make sure you can republish and reinstall your extension without any errors.

Use the correct Dynamics 365 Business Central version

Use Docker for your development and testing. At least, run your full test in Dynamics 365 Business Central online at least once before submitting for validation. We use Docker, and this ensures that you will be testing on the same as what we validate your app on.

If you test in an on-premises deployment, you might miss errors that would be seen online.

And with this, make sure you are using the correct Docker image tag to set up the correct Dynamics 365 Business Central version number. If you want your app to go live as soon as possible with the current production version at the time you submit your app, you must use the image tag mentioned on the Collaborate site. To do so, sign into aka.ms/collaborate, navigate to packages, and locate the build named **Current Build - Developing for Dynamics 365 Business Central**. The image tag never changes and when we roll out a new version to production, the build underneath the image tag automatically changes for you. This means that you are always testing on current production. If you test on a build older or much newer, your app will most likely fail validation.

Use the image tag from the current build link above and make sure you refresh the docker instance each time you want to submit. If you haven't run your Docker script to refresh for months, then you are on a much older build.

Use the correct data when you test your app

When we validate apps, we use the base CRONUS demo data. This of course is there for some countries and you don't have to do anything additional to receive that demo data. However, for countries that are empty and don't include demo data, you must import the CRONUS evaluation demo data. Not the International CRONUS data, the evaluation demo data. We do not use custom data and we do not use any other data. We always use the base evaluation demo data. To get this same data (if you don't have it by default), you follow these instructions:

1. Search for **Configuration Packages**, and then choose the link.
2. Choose **Process > Import Predefined Package**.
3. Click the link for **GB.ENU.EVALUATION**.

That will start processing. You will see the process bar.

4. If there are popups at all, just choose **Yes** or **OK**.
5. Once the process is complete, choose the **Apply Package** button, and then choose **Yes**.
6. Again, if any popups or anything just click through them.
7. Once it completes, sign out of Business Central and then sign back in again.
8. You now have data.

Use the right user for your testing

Do not do your testing with a user that has SUPER permissions marked. The SUPER user can do all without issue and you won't catch your true app bugs. No live customer will have several users marked with this permission set. Therefore, we cannot test with it. You need to setup a user in your test environment that only has the BUS FULL ACCESS permission set, LOCAL, and any of your own permission sets. For information on how to setup this user, see this blog [Enabling Premium Experience in Business Central Sandbox Containers](#).

Testing your app

Now it is time to test your app. The following are all things you must do as part of your testing effort.

- Test your app in its entirety. We expect you to test 100% of the functionality of your app. Testing just a few areas of your app will not suffice. Test everything.
- We are not going to test 100% of the functionality of your app. We expect you to be doing that.
- If the testing works for you, it will most likely work for us.
- Ensure that no permission errors are thrown for any of your app's functionality.
- With the ESSENTIAL user (before you assign your permission sets to it), make sure that the user can still use the core Business Central without facing permission errors. You must allow that user to do things such as accessing the Customer card, posting sales order, and so on.

Maintaining your app

Although we do regular testing of your app when we prepare a new version of Dynamics 365 Business Central, we expect you to do the same on your end. You have access to the same builds that we do through the Collaborate program. You can do more thorough testing than we can because you know your app the best. By doing this testing, you can catch future Dynamics 365 Business Central changes that may impact or break your app. Catching these changes in advance leaves less risk for customers to run into them.

You should be doing regular testing against our release branch that ports into our monthly service updates. To test against these builds, sign into aka.ms/collaborate, navigate to packages, and locate the build named **Daily Builds - Maintaining an app in AppSource for Dynamics 365 Business Central**.

We also recommend that you do regular testing against our release branch that eventually becomes our major release in April or October. For more information, see **Daily Builds - Developing for a future release of Dynamics 365 Business Central** on aka.ms/collaborate.

See Also

[Checklist for Submitting Your App](#)
[Rules and Guidelines for AL Code](#)

User Scenario Documentation

6/17/2019 • 3 minutes to read

One of the keys to a successful extension validation is a document that guides the tester through the setup and usage of the extension. You must include a document that helps Microsoft test the key scenarios of your extension. We want to ensure that we are validating the functionality in the correct manner. Following are some key points to keep in mind when writing the user scenario document.

- Be as detailed as possible. No detail is too small. If a field needs a specific value, include that value in your document.
- Keep the inexperienced user in mind. You know your app well, but other users Microsoft does not.
- Use screenshots as much as possible. They provide a good picture of what you want the user to accomplish.
- Provide all prerequisite and setup steps required for successful test scenario completion.
 - If your app requires setup of its own, include those details.
 - If any setup is extensive, consider using the import of Rapid Start packages.
 - If your app has a dependency on non-standard settings in the core default version of Business Central, include those details. The Microsoft-provided demo data might not have everything that your app needs to work properly.
- Include the most important functionality scenarios of your extension. We are not looking to test your entire extension, but we do want to ensure we are validating the most used aspects of your app.
 - Do not give a summary as to what these scenarios do. List step by step details instead. Again, the tester doing the validation might not have the same product knowledge as you do.

NOTE

This is not the same as the requirement to include Help for your functionality. For more information about getting started with extending and customizing the Business Central user assistance, see [User assistance model](#).

Use the correct Business Central version

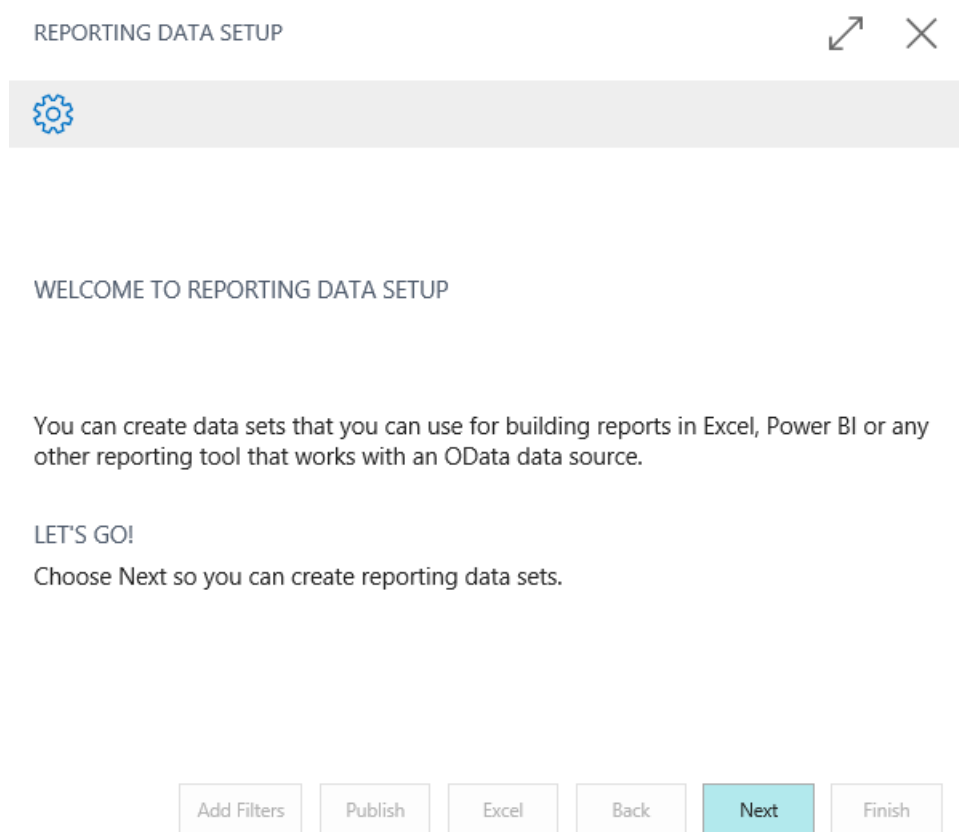
We recommend that you use Docker as a guide for writing your user scenario document. This way, you can take screenshots and other visuals that really help the tester walk through your validation. Keep these things in mind:

- Do not base your user document on an on-premises environment. Business Central on-premises deployments can have different windows, data, and so on. As a result, your document can lead to confusion and differences in our results.
- Use the correct Business Central version. If you are basing your document on a build that is months old, many things could now be different in the latest production environment. This also can lead to much confusion. For more information, see [Current Build - Developing for Dynamics 365 Business Central](#) on the Collaborate site.
- Use the correct data for your document. Do not submit a document based on custom data that our testers will not have access to. You should always base your documents on the base demo CRONUS data and then include Rapid Start packages with any additional data.
- If your app's functionality is different per country, provide that important information. Some of the steps might be different (for your app) between US and DK. If that is the case, mention that in the document.

Example

Here is an example of the level of detail we are looking for, based on running the Assisted Setup wizard:

1. On the Home Page, click the **Setup & Extensions** dropdown
2. Click **Assisted Setup**
3. Click the **Set up reporting data** link
4. This will launch the wizard for this process



5. Click **Next**



I WANT TO...

Create a new data set



Add Filters

Publish

Excel

Back

Next

Finish

6. Click **Next**
7. New Name = TestReport1
8. Data Source Type = Page
9. Data Source Id = 22
10. Data Source Name = Customer List



SELECT THE DATA YOU WOULD LIKE TO USE FOR YOUR REPORTS AND DEFINE A NAME FOR THIS DATA SET.

New Name TestReport1

Data Source Type Page



Data Source Id 22 ...

Data Source Name Customer List

Add Filters

Publish

Excel

Back

Next

Finish

11. Click **Next**

REPORTING DATA SETUP

CHOOSE THE FIELDS TO INCLUDE IN YOUR DATA SET
Changing fields will clear previously set filters.

INC...		REPORT CAPTION	FIELD CAPTION	S
<input checked="" type="checkbox"/>		No	No.	^
<input checked="" type="checkbox"/>		Name	Name	
<input checked="" type="checkbox"/>		Responsibility_Center	Responsibility Center	
<input checked="" type="checkbox"/>		Location_Code	Location Code	
<input type="checkbox"/>		Post_Code	ZIP Code	
<input type="checkbox"/>		Country_Region_Code	Country/Region Code	
<input checked="" type="checkbox"/>		Phone_No	Phone No.	v
<input type="checkbox"/>		IC Partner Code	IC Partner Code	v

Add Filters

Publish

Excel

Back

Next

Finish

12. Click **Publish**

See Also

- [Checklist for Submitting Your App](#)
- [Rules and Guidelines for AL Code](#)
- [User assistance model](#)

Restrictions on UI for Objects Exposed as Web Services

3/31/2019 • 2 minutes to read

Pages and code units that are designed to be exposed as Web services must not generate any UI that would cause an exception in the calling code.

SUMMARY AND INTENT: When writing code for Web services, you must not use end-user confirmation dialog boxes, message boxes, or any other page constructs in the code. Because a Web service runs independently of a user interface, running this type of code causes the code to throw an exception. The exception can be caught and handled, but the Web service will not complete.

RESOURCES: For more information, see [Microsoft Dynamics NAV Web Services](#).

HOW TO COMPLY: Ensure that code for pages and code units that are being exposed as Web services do not use any end-user confirmation dialog boxes or message boxes.

TEST METHODOLOGY: To verify this requirement, the following tests will be performed:

1. Identify the pages and code units that are exposed as Web services during the installation of the extension.
2. Using code inspection, verify that methods from the following table are not used by the pages and code units published by the installation without conditional code that is based on `GUIALLOWED=FALSE` or `CurrFieldNo=0` circumventing their call.

AL METHOD	APPLIES TO
CONFIRM	Codeunit/page
STRMENU	Codeunit/page
(Page RunModal)	Page
Page of type Confirmation Dialog	Page
(Request page)	Page
ERROR	Codeunit/page
BEEP	Codeunit/page
YIELD	Codeunit/page

Additionally, when running the page or code unit as a Web service, the following exception should never occur:

Microsoft.Dynamics.Nav.Types.Exceptions.NavNCLCallbackNotAllowedException: Callback functions are not allowed.

See Also

[Checklist for Submitting Your App](#)
[Rules and Guidelines for AL Code](#)

Replacing OnBeforeCompanyOpen and OnAfterCompanyOpen

6/17/2019 • 2 minutes to read

To improve the login time for Dynamics 365 Business Central, extensions should no longer use the **OnBeforeCompanyOpen** and **OnAfterCompanyOpen** events. Following are some recommended patterns to use in place of these events.

- If the extension is subscribing to **OnBeforeCompanyOpen** or **OnAfterCompanyOpen** in order to complete company setup for a newly created company, we recommend subscribing to `OnCompanyInitialize` from `Codeunit 2` instead.
- If the extension is subscribing to **OnBeforeCompanyOpen** or **OnAfterCompanyOpen** in order to perform some long-running data update, then either call the "update" when the extension gets called in code for the first time or apply the new task scheduler pattern for Update 6 and later.

Task Scheduler example

```
// Add 15s
TASKSCHEDULER.CREATETASK(CODEUNIT::"Job Queue User Handler",0,TRUE,COMPANYNAME,CURRENTDATETIME + 15000);
```

See Also

[Checklist for Submitting Your App](#)

[Rules and Guidelines for AL Code](#)

Building an Advanced Sample Extension

5/3/2019 • 15 minutes to read

It is required to submit tests with your extension in order to pass validation. This walkthrough builds an advanced sample extension which is used as the foundation for writing a test which you can read about in the [Testing the Advanced Sample Extension](#) topic. If you are new to building extensions, we suggest that you get familiar with [Building your first sample extension that uses new objects and extension objects](#).

For information about submitting your app to AppSource, see [Checklist for Submitting Your App](#).

This walkthrough will guide you through all the steps that you must follow to create the sample extension in AL. The final result can be published, installed, and tested on your tenants. After you have built your extension, you must write the test for it.

About this walkthrough

This walkthrough illustrates the following tasks:

- Developing a sample extension that uses codeunits, tables, card pages, list pages, navigate page (Assisted Setup) actions and events, and includes tooltips and links to context-sensitive Help.
- Creating extension objects that can be used to modify page and table objects.
- Initializing the database during the installation of the extension.
- Developing a sample test that tests external calls to a service, events, permissions, actions, navigate page (Assisted Setup), and other modified pages.
- Running the sample test using the Test Tool.

Prerequisites

To complete this walkthrough, you will need:

- The Dynamics 365 Business Central tenant.
- Visual Studio Code.
- The AL Language extension for Visual Studio Code.

For more information on how to get started with your first extension for Dynamics 365 Business Central, see [Getting Started](#).

Customer Rewards extension overview

This sample extension enables the ability to set up any number of reward levels and the minimum number of rewards points required to attain that level. When the sample extension is installed, customers begin to accrue one reward point per sales order. When no reward levels are set up, the customer's reward level is set to 'NONE' even though the customer may have reward points. To begin using the sample extension, the user must accept the extension terms and activate the extension by entering a valid activation code using the **Customer Rewards Assisted Setup Wizard**. Following all the steps of this walkthrough allows you to publish the extension on your tenant and create a possible new feature for your customers.

Developing the sample Customer Rewards extension

In the following section, you will be adding the objects that are needed for the Customer Rewards extension.

Customer Rewards table objects

First, we will get started with the table objects that store the data.

Reward Level table object

The following code adds a new table 50100 **Reward Level** for storing reward level information set up by the user. The table consists of two fields: **Level** and **Minimum Reward Points**.

```
table 50100 "Reward Level"
{
    fields
    {
        field(1; Level; Text[20]) { }

        field(2; "Minimum Reward Points"; Integer)
        {
            MinValue = 0;
            NotBlank = true;

            trigger OnValidate();
            var
                tempPoints: Integer;
                RewardLevel: Record "Reward Level";
            begin
                tempPoints := "Minimum Reward Points";
                RewardLevel.SetRange("Minimum Reward Points", tempPoints);
                if RewardLevel.FindFirst then
                    Error('Minimum Reward Points must be unique');
                end;
            end;
        }
    }

    keys
    {
        key(PK; Level)
        {
            Clustered = true;
        }
        key("Minimum Reward Points"; "Minimum Reward Points") { }
    }

    trigger OnInsert();
    begin
        Validate("Minimum Reward Points");
    end;

    trigger OnModify();
    begin
        Validate("Minimum Reward Points");
    end;
}
```

Activation Code Information table object

The following code adds a new table 50101 **Activation Code Information** for storing activation information for the extension. The table consists of three fields: **ActivationCode**, **Date Activated**, and **Expiration Date**.

```

table 50101 "Activation Code Information"
{
    fields
    {
        field(1; ActivationCode; Text[14])
        {
            Description = 'Activation code used to activate Customer Rewards';
        }

        field(2; "Date Activated"; Date)
        {
            Description = 'Date Customer Rewards was activated';
        }

        field(3; "Expiration Date"; Date)
        {
            Description = 'Date Customer Rewards activation expires';
        }
    }

    keys
    {
        key(PK; ActivationCode)
        {
            Clustered = true;
        }
    }
}

```

Customer Rewards Mgt. Setup table object

The following code adds a new table 50102 **Customer Rewards Mgt. Setup** for storing information about the codeunit that should be used to handle events in the extension. This enables us to mock events in our sample test. The table consists of two fields: **Primary Key** and **Customer Rewards Ext. Mgt. Codeunit ID**.

```

table 50102 "Customer Rewards Mgt. Setup"
{
    fields
    {
        field(1; "Primary Key"; Code[10])
        {
        }

        field(2; "Customer Rewards Ext. Mgt. Codeunit ID"; Integer)
        {
            TableRelation = "CodeUnit Metadata".ID;
        }
    }

    keys
    {
        key(PK; "Primary Key")
        {
            Clustered = true;
        }
    }
}

```

Customer Rewards table extension objects

Customer table extension object

The **Customer** table, like many other tables, is part of the Dynamics 365 Business Central service and it cannot be modified directly by developers. To add additional fields or to change properties on this table, developers must

create a new type of object; a table extension. The following code creates a table extension for the **Customer** table and adds the **RewardPoints** field.

```
tableextension 50100 "CustomerTable Ext." extends Customer
{
    fields
    {
        field(10001; RewardPoints; Integer)
        {
            MinValue = 0;
        }
    }
}
```

Customer Rewards page objects

For each page object, you can specify the target Help page that describes the feature that the page object is part of. The `ContextSensitiveHelpPage` property on the page object works together with the link that is specified in the app.json file. For more information, see [Configure Context-Sensitive Help](#).

Customer Rewards Wizard page object

The following code adds the 50100 **Customer Rewards Wizard** page that enables the user to accept the terms for using the extension as well as activating the extension. The page consists of a welcome step, an activation step, and a finish step. The welcome step has a checkbox for the Terms of Use that must be enabled. The activation step has a text box where the activation code must be entered for validation. A valid activation code for this sample extension is any 14 character alphanumeric code.

```
page 50100 "Customer Rewards Wizard"
{
    // Specifies that this page will be a navigate page.
    PageType = NavigatePage;
    Caption = 'Customer Rewards assisted setup guide';
    ContextSensitiveHelpPage = 'sales-rewards';

    layout
    {
        area(content)
        {
            group(MediaStandard)
            {
                Caption = '';
                Editable = false;
                Visible = TopBannerVisible;

                field("MediaResourcesStandard." "Media Reference"; MediaResourcesStandard."Media Reference")
                {
                    ApplicationArea = All;
                    Editable = false;
                    ShowCaption = false;
                }
            }

            group(FirstPage)
            {
                Caption = '';
                Visible = FirstPageVisible;

                group("Welcome")
                {
                    Caption = 'Welcome';
                    Visible = FirstPageVisible;

                    group(Introduction)
                    {
```

```

        Caption = '';
        InstructionalText = 'This Customer Rewards extension is a sample extension. It adds
rewards tiers support for Customers.';
        Visible = FirstPageVisible;

        field(Spacer1; '')
        {
            ApplicationArea = All;
            ShowCaption = false;
            Editable = false;
            Multiline = true;
        }
    }

    group("Terms")
    {
        Caption = 'Terms of Use';
        Visible = FirstPageVisible;

        group(Terms1)
        {
            Caption = '';
            InstructionalText = 'By enabling the Customer Rewards extension...';
            Visible = FirstPageVisible;
        }
    }

    group(Terms2)
    {
        Caption = '';

        field(EnableFeature; EnableCustomerRewards)
        {
            ApplicationArea = All;
            Multiline = true;
            Editable = true;
            Caption = 'I understand and accept these terms.';

            trigger OnValidate();
            begin
                ShowFirstPage;
            end;
        }
    }
}

group(SecondPage)
{
    Caption = '';
    Visible = SecondPageVisible;

    group("Activation")
    {
        Caption = 'Activation';
        Visible = SecondPageVisible;

        field(Spacer2; '')
        {
            ApplicationArea = All;
            ShowCaption = false;
            Editable = false;
            Multiline = true;
        }

        group(ActivationMessage)
        {
            Caption = '';
            InstructionalText = 'Enter your 14 digit activation code to continue';

```

```

        Visible = SecondPageVisible;

        field(Activationcode; ActivationCode)
        {
            ApplicationArea = All;
            ShowCaption = false;
            Editable = true;
        }
    }
}

group(FinalPage)
{
    Caption = '';
    Visible = FinalPageVisible;

    group("ActivationDone")
    {
        Caption = 'You're done!';
        Visible = FinalPageVisible;

        group(DoneMessage)
        {
            Caption = '';
            InstructionalText = 'Click Finish to setup your rewards level and start using Customer
Rewards.';
            Visible = FinalPageVisible;
        }
    }
}

actions
{
    area(Processing)
    {
        action(ActionBack)
        {
            ApplicationArea = All;
            Caption = 'Back';
            Enabled = BackEnabled;
            Visible = BackEnabled;
            Image = PreviousRecord;
            InFooterBar = true;

            trigger OnAction();
            begin
                NextStep(true);
            end;
        }

        action(ActionNext)
        {
            ApplicationArea = All;
            Caption = 'Next';
            Enabled = NextEnabled;
            Visible = NextEnabled;
            Image = NextRecord;
            InFooterBar = true;

            trigger OnAction();
            begin
                NextStep(false);
            end;
        }

        action(ActionActivate)
    }
}

```

```

    action(ActionActivate)
    {
        ApplicationArea = All;
        Caption = 'Activate';
        Enabled = ActivateEnabled;
        Visible = ActivateEnabled;
        Image = NextRecord;
        InFooterBar = true;

        trigger OnAction();
        var
            CustomerRewardsExtMgt: Codeunit "Customer Rewards Ext. Mgt.";
        begin
            if ActivationCode = '' then
                Error('Activation code cannot be blank.');
            if Text.StrLen(ActivationCode) <> 14 then
                Error('Activation code must have 14 digits.');
            if CustomerRewardsExtMgt.ActivateCustomerRewards(ActivationCode) then
                NextStep(false)
            else
                Error('Activation failed. Please check the activation code you entered.');
        end;
    }

    action(ActionFinish)
    {
        ApplicationArea = All;
        Caption = 'Finish';
        Enabled = FinalPageVisible;
        Image = Approve;
        InFooterBar = true;

        trigger OnAction();
        begin
            FinishAndEnableCustomerRewards
        end;
    }
}

trigger OnInit();
begin
    LoadTopBanners;
end;

trigger OnOpenPage();
begin
    Step := Step::First;
    EnableControls;
end;

local procedure EnableControls();
begin
    ResetControls;

    case Step of
        Step::First :
            ShowFirstPage;

        Step::Second :
            ShowSecondPage;

        Step::Finish :
            ShowFinalPage;
    END;
end;

```

```

local procedure NextStep(backwards: boolean);
begin
    if Backwards then
        Step := Step - 1
    ELSE
        Step := Step + 1;
    EnableControls;
end;

local procedure FinishAndEnableCustomerRewards();
var
    CustomerRewardsExtMgt: Codeunit "Customer Rewards Ext. Mgt.";
begin
    CurrPage.Close;
    CustomerRewardsExtMgt.OpenRewardsLevelPage;
end;

local procedure ShowFirstPage();
begin
    FirstPageVisible := true;
    SecondPageVisible := false;
    FinishEnabled := false;
    BackEnabled := false;
    ActivateEnabled := false;
    NextEnabled := EnableCustomerRewards;
end;

local procedure ShowSecondPage();
begin
    FirstPageVisible := false;
    SecondPageVisible := true;
    FinishEnabled := false;
    BackEnabled := true;
    NextEnabled := false;
    ActivateEnabled := true;
end;

local procedure ShowFinalPage();
begin
    FinalPageVisible := true;
    BackEnabled := true;
    NextEnabled := false;
    ActivateEnabled := false;
end;

local procedure ResetControls();
begin
    FinishEnabled := true;
    BackEnabled := true;
    NextEnabled := true;
    ActivateEnabled := true;
    FirstPageVisible := false;
    SecondPageVisible := false;
    FinalPageVisible := false;
end;

local procedure LoadTopBanners();
begin
    if MediaRepositoryStandard.GET('AssistedSetup-NoText-400px.png', FORMAT(CURRENTCLIENTTYPE))
    then
        if MediaResourcesStandard.GET(MediaRepositoryStandard."Media Resources Ref")
        then
            TopBannerVisible := MediaResourcesStandard."Media Reference".HASVALUE;
        end;
    end;

var
    MediaRepositoryStandard: Record 9400;
    MediaResourcesStandard: Record 2000000182;
    Step: Option First, Second, Finish;

```

```

        ActivationCode: Text;
        TopBannerVisible: Boolean;
        FirstPageVisible: Boolean;
        SecondPageVisible: Boolean;
        FinalPageVisible: Boolean;
        FinishEnabled: Boolean;
        BackEnabled: Boolean;
        NextEnabled: Boolean;
        ActivateEnabled: Boolean;
        EnableCustomerRewards: Boolean;
    }

```

Rewards Level List page object

The following code adds the 50101 **Rewards Level List** page that enables the user to view, edit, or add new reward levels and their corresponding minimum required points. The code example includes tooltips for controls and a relative link to context-sensitive Help.

```

page 50101 "Rewards Level List"
{
    PageType = List;
    ContextSensitiveHelpPage = 'sales-rewards';
    SourceTable = "Reward Level";
    SourceTableView = sorting ("Minimum Reward Points") order(ascending);

    layout
    {
        area(content)
        {
            repeater(Group)
            {
                field(Level; Level)
                {
                    ApplicationArea = All;
                    Tooltip = 'Specifies the level of reward that the customer has at this point.';
                }

                field("Minimum Reward Points"; "Minimum Reward Points")
                {
                    ApplicationArea = All;
                    Tooltip = 'Specifies the number of points that customers must have to reach this level.';
                }
            }
        }
    }

    trigger OnOpenPage();
    begin
        if(not CustomerRewardsExtMgt.IsCustomerRewardsActivated) then
            Error(NotActivatedTxt);
        end;

    var
        CustomerRewardsExtMgt: Codeunit "Customer Rewards Ext. Mgt.";
        NotActivatedTxt: Label 'Customer Rewards is not activated';
    }

```

Customer Rewards page extension objects

Customer card page extension object

A page extension object can be used to add new functionality to pages that are part of the Dynamics 365 Business Central service. The following page extension object extends the **Customer Card** page object by adding two field controls: **RewardLevel** and **RewardPoints** after the **Name** field control on the page. The fields are added in the

layout section.

```
pageextension 50100 "Customer Card Ext." extends "Customer Card"
{
    layout
    {
        addafter(Name)
        {
            field(RewardLevel; RewardLevel)
            {
                ApplicationArea = All;
                Caption = 'Reward Level';
                Description = 'Reward level of the customer.';
                ToolTip = 'Specifies the level of reward that the customer has at this point.';
                Editable = false;
            }

            field(RewardPoints; RewardPoints)
            {
                ApplicationArea = All;
                Caption = 'Reward Points';
                Description = 'Reward points accrued by customer';
                ToolTip = 'Specifies the total number of points that the customer has at this point.';
                Editable = false;
            }
        }
    }

    trigger OnAfterGetRecord();
    var
        CustomerRewardsMgtExt: Codeunit "Customer Rewards Ext. Mgt.";
    begin
        // Get the reward level associated with reward points
        RewardLevel := CustomerRewardsMgtExt.GetRewardLevel(RewardPoints);
    end;

    var
        RewardLevel: Text;
}
```

Customer list page extension object

A page extension object can be used to add new functionality to pages that are part of the Dynamics 365 Business Central service. The following page extension object extends the **Customer List** page object by adding one action control; **Reward Levels** to the **Customer** group on the page.

```

pageextension 50101 "Customer List Ext." extends "Customer List"
{
    actions
    {
        addfirst("&Customer")
        {
            action("Reward Levels")
            {
                ApplicationArea = All;
                Image = CustomerRating;
                Promoted = true;
                PromotedCategory = Process;
                PromotedIsBig = true;
                ToolTip = 'Open the list of reward levels.';

                trigger OnAction();
                begin
                    if CustomerRewardsExtMgt.IsCustomerRewardsActivated then
                        CustomerRewardsExtMgt.OpenRewardsLevelPage
                    else
                        CustomerRewardsExtMgt.OpenCustomerRewardsWizard;
                end;
            }
        }
    }

    var
        CustomerRewardsExtMgt: Codeunit "Customer Rewards Ext. Mgt.";
}

```

Customer Rewards codeunit objects

Customer Rewards Install Logic codeunit object

The following code adds the 50100 **Customer Rewards Install Logic** codeunit that initializes the default codeunit that will be used for handling events. Because this is an install codeunit, it has its **Subtype** property set to **Install**. The **OnInstallAppPerCompany** trigger is run when the extension is installed for the first time and the same version is re-installed.

```

codeunit 50100 "Customer Rewards Install Logic"
{
    // Customer Rewards Install Logic
    Subtype = Install;

    trigger OnInstallAppPerCompany();
    begin
        SetDefaultCustomerRewardsExtMgtCodeunit;
    end;

    procedure SetDefaultCustomerRewardsExtMgtCodeunit();
    var
        CustomerRewardsExtMgtSetup: Record "Customer Rewards Mgt. Setup";
    begin
        CustomerRewardsExtMgtSetup.DeleteAll;
        CustomerRewardsExtMgtSetup.Init;
        // Default Customer Rewards Ext. Mgt codeunit to use for handling events
        CustomerRewardsExtMgtSetup."Customer Rewards Ext. Mgt. Codeunit ID" := Codeunit::"Customer Rewards
        Ext. Mgt.";
        CustomerRewardsExtMgtSetup.Insert;
    end;
}

```

Customer Rewards Ext. Mgt. codeunit object

The 50101 **Customer Rewards Ext. Mgt.** codeunit encapsulates most of the logic and functionality required for

the Customer Rewards extension. This codeunit contains examples of how we can use events to react to specific actions or behavior that occur within our extension. In this sample extension, there is the need to make a call to an external service or API to validate activation codes entered by the user. Typically, you may do this by defining procedures that take in the activation code and then make calls to the API. Instead of using that approach, we use events in AL. Let us look at the following code from the codeunit.

```

// Activates Customer Rewards if activation code is validated successfully
procedure ActivateCustomerRewards(ActivationCode: Text): Boolean;
var
    ActivationCodeInfo: Record "Activation Code Information";
begin
    // raise event
    OnGetActivationCodeStatusFromServer(ActivationCode);
    exit(ActivationCodeInfo.Get(ActivationCode));
end;

// publishes event
[IntegrationEvent(false, false)]
procedure OnGetActivationCodeStatusFromServer(ActivationCode: Text);
begin
end;

// Subscribes to OnGetActivationCodeStatusFromServer event and handles it when the event is raised
[EventSubscriber(ObjectType::Codeunit, Codeunit::"Customer Rewards Ext. Mgt.",
'OnGetActivationCodeStatusFromServer', '', false, false)]
local procedure OnGetActivationCodeStatusFromServerSubscriber(ActivationCode: Text);
var
    ActivationCodeInfo: Record "Activation Code Information";
    ResponseText: Text;
    Result: JsonToken;
    JsonReponse: JsonToken;

begin
    if not CanHandle then
        exit; // use the mock
    // Get response from external service and update activation code information if successful
    if(GetHttpResponse(ActivationCode, ResponseText)) then begin
        JsonReponse.ReadFrom(ResponseText);

        if(JsonReponse.SelectToken('ActivationResponse', Result)) then begin

            if(Result.AsValue().AsText() = 'Success') then begin

                if(ActivationCodeInfo.FindFirst()) then
                    ActivationCodeInfo.Delete;

                ActivationCodeInfo.Init;
                ActivationCodeInfo.ActivationCode := ActivationCode;
                ActivationCodeInfo."Date Activated" := Today;
                ActivationCodeInfo."Expiration Date" := CALCDATE('<1Y>', Today);
                ActivationCodeInfo.Insert;
            end;
        end;
    end;
end;

// Helper method to make calls to a service to validate activation code
local procedure GetHttpResponse(ActivationCode: Text; var ResponseText: Text): Boolean;
begin
    // You will typically make external calls / http requests to your service to validate the activation
code
    // here but for the sample extension we simply return a successful dummy response
    if ActivationCode = '' then
        exit(false);

    ResponseText := DummySuccessResponseTxt;
    exit(true);
end;

```

We define an event publisher method **OnGetActivationCodeStatusFromServer** that accepts the activation code entered by the user as a parameter; and, a subscriber method **OnGetActivationCodeStatusFromServerSubscriber** to listen for and handle the event. When the

ActivateCustomerRewards procedure is run, the **OnGetActivationCodeStatusFromServer** event is raised. Because the **EventSubscriberInstance** property for the codeunit is set to **Static-Automatic** by default, the **OnGetActivationCodeStatusFromServerSubscriber** procedure is called. In this procedure, we handle the raised event by first checking if the current codeunit has been defined for handling this event. If the codeunit can handle the event, the **GetHttpResponse** helper procedure is called to validate the activation code. Depending on the response, Customer Rewards is activated or not.

By using events when the extension makes external calls to a service, we are able to mock the behavior of what happens when events are raised. This becomes particularly useful when writing tests for the extension.

For more information about events, see [Events in Microsoft Dynamics 365 Business Central](#).

Below is the full code for this codeunit.

```
codeunit 50101 "Customer Rewards Ext. Mgt."
{
    var
        DummySuccessResponseTxt: Label '{"ActivationResponse": "Success"}', Locked = true;
        NoRewardlevelTxt: TextConst ENU = 'NONE';

    // Determines if the extension is activated
    procedure IsCustomerRewardsActivated(): Boolean;
    var
        ActivationCodeInfo: Record "Activation Code Information";
    begin
        if not ActivationCodeInfo.FindFirst then
            exit(false);

        if(ActivationCodeInfo."Date Activated" <= Today) and(Today <= ActivationCodeInfo."Expiration Date")
    then
        exit(true);
        exit(false);
    end;

    // Opens the Customer Rewards Assisted Setup Guide
    procedure OpenCustomerRewardsWizard();
    var
        CustomerRewardsWizard: Page "Customer Rewards Wizard";
    begin
        CustomerRewardsWizard.RunModal;
    end;

    // Opens the Reward Level page
    procedure OpenRewardsLevelPage();
    var
        RewardsLevelPage: Page "Rewards Level List";
    begin
        RewardsLevelPage.Run;
    end;

    // Determines the correponding reward level and returns it
    procedure GetRewardLevel(RewardPoints: Integer) RewardLevelTxt: Text;
    var
        RewardLevelRec: Record "Reward Level";
        MinRewardLevelPoints: Integer;
    begin
        RewardLevelTxt := NoRewardlevelTxt;

        if RewardLevelRec.IsEmpty then
            exit;
        RewardLevelRec.SetRange("Minimum Reward Points", 0, RewardPoints);
        RewardLevelRec.SetCurrentKey("Minimum Reward Points"); // sorted in ascending order

        if not RewardLevelRec.FindFirst then
            exit;
        MinRewardLevelPoints := RewardLevelRec."Minimum Reward Points";
```

```

        if RewardPoints >= MinRewardLevelPoints then begin
            RewardLevelRec.Reset;
            RewardLevelRec.SetRange("Minimum Reward Points", MinRewardLevelPoints, RewardPoints);
            RewardLevelRec.SetCurrentKey("Minimum Reward Points"); // sorted in ascending order
            RewardLevelRec.FindLast;
            RewardLevelTxt := RewardLevelRec.Level;
        end;
    end;

// Activates Customer Rewards if activation code is validated successfully
procedure ActivateCustomerRewards(ActivationCode: Text): Boolean;
var
    ActivationCodeInfo: Record "Activation Code Information";
begin
    // raise event
    OnGetActivationCodeStatusFromServer(ActivationCode);
    exit(ActivationCodeInfo.Get(ActivationCode));
end;

// publishes event
[IntegrationEvent(false, false)]
procedure OnGetActivationCodeStatusFromServer(ActivationCode: Text);
begin
end;

// Subscribes to OnGetActivationCodeStatusFromServer event and handles it when the event is raised
[EventSubscriber(ObjectType::Codeunit, Codeunit::"Customer Rewards Ext. Mgt.",
'OnGetActivationCodeStatusFromServer', '', false, false)]
local procedure OnGetActivationCodeStatusFromServerSubscriber(ActivationCode: Text);
var
    ActivationCodeInfo: Record "Activation Code Information";
    ResponseText: Text;
    Result: JsonToken;
    JsonReponse: JsonToken;
begin
    if not CanHandle then
        exit; // use the mock

    // Get response from external service and update activation code information if successful
    if(GetHttpResponse(ActivationCode, ResponseText)) then begin
        JsonReponse.ReadFrom(ResponseText);

        if(JsonReponse.SelectToken('ActivationResponse', Result)) then begin

            if(Result.AsValue().AsText() = 'Success') then begin

                if(ActivationCodeInfo.FindFirst()) then
                    ActivationCodeInfo.Delete;

                ActivationCodeInfo.Init;
                ActivationCodeInfo.ActivationCode := ActivationCode;
                ActivationCodeInfo."Date Activated" := Today;
                ActivationCodeInfo."Expiration Date" := CALCDATE('<1Y>', Today);
                ActivationCodeInfo.Insert;

            end;
        end;
    end;
end;

// Helper method to make calls to a service to validate activation code
local procedure GetHttpResponse(ActivationCode: Text; var ResponseText: Text): Boolean;
begin
    // You will typically make external calls / http requests to your service to validate the activation
code
    // here but for the sample extension we simply return a successful dummy response
    if ActivationCode = '' then
        exit(false);

```

```

        ResponseText := DummySuccessResponseTxt;
        exit(true);
    end;

    // Subscribes to the OnAfterReleaseSalesDoc event and increases reward points for the sell to customer in
    posted sales order
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"Release Sales Document", 'OnAfterReleaseSalesDoc', '',
    false, false)]
    local procedure OnAfterReleaseSalesDocSubscriber(VAR SalesHeader: Record "Sales Header"; PreviewMode:
    Boolean; LinesWereModified: Boolean);
    var
        Customer: Record Customer;
    begin
        if SalesHeader.Status <> SalesHeader.Status::Released then
            exit;

            Customer.Get(SalesHeader."Sell-to Customer No.");
            Customer.RewardPoints += 1; // Add a point for each new sales order
            Customer.Modify;
        end;

        // Checks if the current codeunit is allowed to handle Customer Rewards Activation requests rather than a
        mock.
        local procedure CanHandle(): Boolean;
        var
            CustomerRewardsExtMgtSetup: Record "Customer Rewards Mgt. Setup";
        begin
            if CustomerRewardsExtMgtSetup.Get then
                exit(CustomerRewardsExtMgtSetup."Customer Rewards Ext. Mgt. Codeunit ID" = CODEUNIT::"Customer
                Rewards Ext. Mgt.");
                exit(false);
            end;
        }
    }

```

Conclusion

At this point, the Customer Rewards sample extension can be published and installed on your sandbox. To continue writing tests for the sample extension, see [Testing the Advanced Sample Extension](#).

See Also

[Developing Extensions](#)

[Getting Started with AL](#)

[How to: Publish and Install an Extension](#)

[Converting Extensions V1 to Extensions V2](#)

[Configure Context-Sensitive Help](#)

Testing the Advanced Sample Extension

3/31/2019 • 27 minutes to read

It is required to submit tests with your extension in order to pass validation. This walkthrough builds on the advanced sample extension which you can read about here [Building an Advanced Sample Extension](#). If you are new to building extensions, we suggest that you get familiar with [Building your first sample extension that uses new objects and extension objects](#). This walkthrough goes through how you develop the test for the sample Customer Rewards extension.

For information about submitting your app to AppSource, see [Checklist for Submitting Your App](#).

Prerequisites

To complete this walkthrough, you will need:

- Dynamics 365 Business Central Docker container-based development environment. For more information, see [Get started with the Container Sandbox Development Environment](#) and [Running a Container-Based Development Environment](#).
- [Visual Studio Code](#).
- The [AL Language extension](#) for Visual Studio Code.

Identifying the areas of the extension that need to be tested

Before writing tests for your extension, you need to identify all the areas of the extension that need to be tested.

- Ensure that your tests cover all the setup and usage scenario steps found in the [user scenario document](#). This includes Assisted Setup, pages, fields, actions, events, and other controls and objects used by your extension.
- The CRONUS demo company will be used. If your app requires setup within the core product or any additional data, remember to include that in your tests.
- As part of your tests, remember to include tests that verify that the extension works as expected for **a user that does not have SUPER permissions**.
- Your tests **should not make any requests to an external service**. Mock your external calls to prevent this from happening.

In the sample test we will consider the following:

- Logic in our **Install** codeunit.
- **Assisted Setup - Customer Rewards Wizard** page. We will verify that the wizard behaves as expected. It can be used to completion without errors. The Assisted Setup contains code that mimics making calls to an external service or API. Because our tests cannot make requests to an external service, we will mock the requests and the responses.
- **Reward Level** page. We will verify that the page behaves as expected when the user opens it whether Customer Rewards is activated or not.
- **Customer List** page. We will verify that our new **Reward Levels** action exists on the page and that it behaves as expected whether the extension is activated or not.
- **Customer Card** page. We will verify that the page has the **Reward Level** and **Reward Points** field that we added.
- **New Customer** should have zero reward points and corresponding reward level if defined.

- Different scenarios involving Customers and Sales Orders to verify that **Reward Points** work as expected and that reward levels for reward points work as defined by the user.
- Each test will also verify that the extension works for a user that does not have SUPER permissions.

Writing the tests

We will first create a new project (CustomerRewardsTest) for the tests. You are required to separate the extension and the tests into separate projects.

Before we can start writing the tests for the extension, we need to do the following:

- Specify the dependencies between the extension (CustomerRewards) and the test (CustomerRewardsTest) projects.

Our CustomerRewardsTest project will be referencing objects from the CustomerRewards project and so we will need to specify this in the `dependencies` setting in the CustomerRewardsTest project's app.json file. The `dependencies` setting takes a list of dependencies, where each dependency specifies the `appId`, `name`, `publisher`, and `version` of the base project/package that the current project/package will depend on.

NOTE

Another prerequisite is to update the app.json with a dependency to the test toolkit.

```
{
  ...
  "dependencies": [
    {
      "appId": "c228bdcf-7112-480b-a832-da81971b6feb",
      "name": "CustomerRewards",
      "publisher": "Microsoft",
      "version": "1.0.0.0"
    }
  ],
  "test": "13.0.0.0"
  ...
}
```

For more information, see [JSON Files](#).

After setting the `dependencies` value, you will be prompted to download the symbols from the base project/package if they are not present.

Application Test Toolkit

We will be using the Application Test Toolkit to automate and run the tests that we write. The toolkit includes:

- Codeunits with test functions to test various application areas.
- Codeunits with generic and application-specific functions to reduce duplication of test code.
- Application objects for running application tests such as the **Test Tool** page.

In order to install the Application Test Toolkit:

1. Open the Nav Container Helper prompt found on the Desktop. You will see a list of functions that you can run on the container.
2. Run the `Import-TestToolkitToNavContainer` function with `-containerName` parameter to import the test toolkit

into the application database.

```
Import-TestToolkitToNavContainer -containerName <name-of-container>
```

Alternatively, if you use the `New-NavContainer` function from the NavContainerHelper PowerShell module to create your containers on Docker, you can add the `-includeTestToolkit` flag. This will install the Application Test Toolkit during the creation of your container.

Describing your tests

To help you design the relevant tests for your functionality, you can write scenarios that outline what you want to test, and you can write test criteria in the GIVEN-WHEN-THEN format. By adding comments based on feature, scenario, and GIVEN-WHEN-THEN, you add structure to your test code and make tests readable.

The following sections provide an overview of the tags that we recommend you to use.

FEATURE Tag

```
// [FEATURE] [<FeatureTag1>][<FeatureTagN>]
```

`FeatureTag` represents the name of the feature, application area, functional area, or another aspect of the application. This list of tags must point to an area of your solution that is touched by the test. Order tags in descending importance. Start with the most important tags referring to the WHEN or THEN steps. The `[FEATURE]` tag can be set for the whole test codeunit. This means all tests in this codeunit will inherit the list of tags set there. If a test is supposed to have the same list of tags as the codeunit has, you do not have to add the `[FEATURE]` tag for this test. Add the tags only if the test has something specific to say.

SCENARIO Tag

```
// [SCENARIO <ScenarioID>] <TestDescription>
```

`ScenarioID` links the test to a work item for the functionality. For example, if you use Visual Studio Online or Team Foundation Server, `[SCENARIO 12345]` represents a work item with the ID 12345.

`TestDescription` represents a short description of the purpose of the test, such as *Annie can apply a deferral template to a purchase order*.

GIVEN-WHEN-THEN Tags

The `GIVEN-WHEN-THEN` tags provide a framework for the specific test criteria.

TAG	DESCRIPTION
GIVEN	Describes one step in setting up the test. If you feel a need to add an AND, you should probably add a separate GIVEN. In most of cases, in order to run an action under test, you must prepare the database. Tests can be complex, so you can add more than one GIVEN. They can come in one block or comment particular lines of code. Do not try to repeat code and comment each line. Instead, add information of a higher level that would be valuable when reading without the test code.

TAG	DESCRIPTION
WHEN	Describes the action under test. A test is to test one thing. There should be only one WHEN in a test. It is the line of code that changes the state of something that we are going to verify. If you feel a need to add more than one WHEN followed by different verification, you should split this test in two or more tests.
THEN	Describes what is verified by the test. All tests must have a verification part. If there is no verification, the test does not test anything. You can add more than one THEN tag.

We can now begin writing the tests for the extension.

MockCustomerRewardsExtMgt codeunit object

The 50102 **MockCustomerRewardsExtMgt** codeunit contains all the code that mocks the process of validating the activation code for Customer Rewards. Because we cannot make requests to external services in the tests, we define a subscriber method **MockOnGetActivationCodeStatusFromServerSubscriber** for handling the **OnGetActivationCodeStatusFromServer** event when it is raised in the **Customer Rewards Ext. Mgt.** codeunit. The **EventSubscriberInstance** property for this codeunit is set to **Manual** so that we can control when the subscriber function is called. We want the subscriber method to be called only during our tests. We also define a Setup procedure that modifies the **Customer Rewards Ext. Mgt. Codeunit ID** in the **Customer Rewards Mgt. Setup** table so that the actual **OnGetActivationCodeStatusFromServerSubscriber** will not handle **OnGetActivationCodeStatusFromServer** event when it is raised.

```
codeunit 50102 MockCustomerRewardsExtMgt
{
    // When set to Manual subscribers in this codeunit are bound to an event by calling the BINDSUBSCRIPTION
    // method.
    // This enables you to control which event subscriber instances are called when an event is raised.
    // If the BINDSUBSCRIPTION method is not called, then nothing will happen when the published event is
    // raised.

    EventSubscriberInstance = Manual;
    var
        DummyResponseTxt: Text;
        DummySuccessResponseTxt: Label '{"ActivationResponse": "Success"}', Locked = true;
        DummyFailureResponseTxt: Label '{"ActivationResponse": "Failure"}', Locked = true;

    // Mocks the response text for testing success and failure scenarios

    procedure MockActivationResponse(Success: Boolean);
    begin
        if Success then
            DummyResponseTxt := DummySuccessResponseTxt
        else
            DummyResponseTxt := DummyFailureResponseTxt;
    end;

    // Modifies the default Customer Rewards Ext. Mgt codeunit to this codeunit to prevent the
    // OnGetActivationCodeStatusFromServerSubscriber in Customer Rewards Ext. Mgt from handling
    // the OnGetActivationCodeStatusFromServer event when it is raised

    procedure Setup();
    var
        CustomerRewardsExtMgtSetup: Record "Customer Rewards Mgt. Setup";
    begin
        CustomerRewardsExtMgtSetup.Get;
        CustomerRewardsExtMgtSetup."Customer Rewards Ext. Mgt. Codeunit ID" :=
```

```

Codeunit::MockCustomerRewardsExtMgt;
    CustomerRewardsExtMgtSetup.Modify;
end;

// Subscribes to OnGetActivationCodeStatusFromServer event and handles it when the event is raised

[EventSubscriber(ObjectType::Codeunit, Codeunit::"Customer Rewards Ext. Mgt.",
'OnGetActivationCodeStatusFromServer', '', false, false)]

local procedure MockOnGetActivationCodeStatusFromServerSubscriber(ActivationCode: Text);
var
    ActivationCodeInfo: Record "Activation Code Information";
    ResponseText: Text;
    Result: JsonToken;
    JsonReponse: JsonToken;
begin
    if(MockGetHttpResponse(ActivationCode, ResponseText)) then begin
        JsonReponse.ReadFrom(ResponseText);

        if(JsonReponse.SelectToken('ActivationResponse', Result)) then begin
            if(Result.AsValue().AsText() = 'Success') then begin
                if ActivationCodeInfo.FindFirst then
                    ActivationCodeInfo.Delete;
                    ActivationCodeInfo.Init;
                    ActivationCodeInfo.ActivationCode := ActivationCode;
                    ActivationCodeInfo."Date Activated" := Today;
                    ActivationCodeInfo."Expiration Date" := CALCDATE('<1Y>', Today);
                    ActivationCodeInfo.Insert;
                end;
            end;
        end;
    end;

    // Mocks making calls to external service

local procedure MockGetHttpResponse(ActivationCode: Text; var ResponseText: Text): Boolean;
begin
    if ActivationCode = '' then
        exit(false);

    ResponseText := DummyResponseTxt;

    exit(true);
end;
}

```

Customer Rewards Test codeunit object

A test codeunit must have its **Subtype** property set to **Test** and the test methods must be decorated with the `[Test]` attribute. When a test codeunit runs, it executes the **OnRun** trigger, and then executes each test method in the codeunit. By default, each test function runs in a separate database transaction, but you can use the **TransactionModel** attribute on test methods to control the transactional behavior. The outcome of a test method is either SUCCESS or FAILURE. If any error is raised by either the code that is being tested or the test code, then the outcome is FAILURE and the error is included in the results log file. Even if the outcome of one test method is FAILURE, the next test methods are still executed.

In addition to the Application Test Toolkit, the following features are available to help you test your extension:

Test pages

Test pages mimic actual pages, but do not present any UI on a client computer. Test pages let you test the code on a page by using AL to simulate user interaction with the page. You can access the fields on a page and the properties of a page or a field by using the dot notation. You can open and close test pages, perform actions on the test page, and navigate around the test page by using AL methods.

UI handlers

To create tests that can be automated, you must handle cases when user interaction is requested by code that is being tested. UI handlers run instead of the requested UI. UI handlers provide the same exit state as the UI. For example, a test method that has a ConfirmHandler handles CONFIRM method calls. If code that is being tested calls the CONFIRM method, then the ConfirmHandler method is called instead of the CONFIRM method. You write code in the ConfirmHandler method to verify that the expected question is displayed by the CONFIRM method and you write AL code to return the relevant reply. The following table describes the available UI handlers.

FUNCTION TYPE	SYNTAX EXAMPLE	PURPOSE	
MessageHandler	<pre>[MessageHandler] procedure MessageHandler(Msg : Text[1024]);</pre>	This handler is called when a message function is invoked in the code. The parameter type, Text , contains the text of the function.	
ConfirmHandler	<pre>[ConfirmHandler] procedure ConfirmHandlerNo(Question: Text[1024]; var Reply: Boolean);</pre>	This handler is called when a confirm function is invoked in the code. The parameter type, Text , contains the text of the function and the parameter Reply if the response to confirm is yes or no.	
StrMenuHandler	<pre>[StrMenuHandler] procedure StrMenuHandler(Option: Text[1024]; var Choice: Integer; Instruction: Text[1024]);</pre>	This handler is called when a StrMenu function is invoked in code. The parameter type, Text , contains the text of the function and Choice is the option chosen in the StrMenu. Options is the list of the different option values and Instruction is the leading text.	
PageHandler	<pre>[PageHandler] procedure MappingPageHandler(var MappingPage: TestPage 1214);</pre>	This handler is called when a non-modal page is invoked in the code. TestPage is the specific page in this case.	
ModalPageHandler	<pre>[ModalPageHandler] procedure DevSelectedObjectPageHandler(DevSelectedObjects: TestPage 89015);</pre>	This handler is called when a modal page is invoked in the code. TestPage is the specific page in this case.	
ReportHandler	<pre>[ReportHandler] procedure VendorListReportHandler(var VendorList: Report 301);</pre>	This handler is called when a report is invoked in the code. Report is the specific report in this case.	

FUNCTION TYPE	SYNTAX EXAMPLE	PURPOSE	
RequestPageHandler	<pre>[RequestPageHandler] procedure SalesInvoiceReportRequestPageHandler(SalesInvoice: TestRequestPage, 206);</pre>	This handler is called when a report is invoked in the code. TestRequestPage handles the specific report ID.	

You must create a specific handler for each page that you want to handle. Any unhandled UI in the test methods of the test codeunit causes a failure of the test.

ASSERTERROR statement

When you test your extension, you should test that your code performs as expected under both successful and failing conditions. These are called positive and negative tests. To test how your extension performs under failing conditions, you can use the ASSERTERROR keyword. The ASSERTERROR keyword specifies that an error is expected at run time in the statement that follows the ASSERTERROR keyword. If a simple or compound statement that follows the ASSERTERROR keyword causes an error, then execution successfully continues to the next statement in the test function. If a statement that follows the ASSERTERROR keyword does not cause an error, then the ASSERTERROR statement itself fails with an error, and the test function that is running produces a FAILURE result.

The 50103 **Customer Rewards Test** codeunit contains all the tests for the Customer Rewards extension. For each test method, we follow the following pattern:

- Initialize and set up the conditions for the test.
- Invoke the business logic that you want to test.
- Validate that the business logic performed as expected.

Let us look some of the sample tests.

TestOnInstallLogic Test

This test verifies that the logic we defined in our Install codeunit works as expected. We first call a helper method **Initialize** which initializes and cleans up any objects that will be needed for the test. The Initialize method also binds our mock codeunit **MockCustomerRewardsExtMgt** to our test codeunit so that any events raised during our test can be handled by the subscriber methods specified in our mock codeunit.

Next, we invoke the **SetDefaultCustomerRewardsExtMgtCodeunit** method, which is the method defined in our Install codeunit.

And finally, we verify using the **Assert** codeunit from the Application Test Toolkit, that the **Customer Rewards Mgt. Setup** table contains the expected codeunit ID.

TestCustomerRewardsWizardActivationPageErrorsWhenInvalidActivationCodeEntered Test

This is one of the tests that focus on the **Customer Rewards Assisted Setup Guide**. The test verifies that an error message is displayed when a not valid activation code is entered in the wizard.

First, **Initialize** is called to clean up previous state and bind our mock subscriber methods to the test codeunit. Additionally, we set our MockActivationResponse to return FAILURE since we are mocking a not valid validation of the activation code. We also use the **Library - Lower Permissions** codeunit to restrict the users permission to one that does not have the SUPER permission.

Next, we open the **Customer Rewards Wizard** by using a Customer Rewards Wizard, the TestPage object is used to mimic the actual page. On the page, the activation code is entered and then the Activate action is invoked.

And finally, we verify that an error message is displayed because the validation of the activation code failed. If no other error is reported then we are also able to conclude that the functionality in this test can be run without the

need for a SUPER permission.

TestRewardLevelsActionExistsOnCustomerListPage Test

This test verifies that the new **Reward Levels** action exists on the Customer List page.

TestCustomerHasBronzeRewardLevelAfterPostedSalesOrders Test

This is one of the tests that considers the interaction between Customers, Sales Orders, and Reward Levels. This test verifies that when two sales orders are made for a new customer, that customer accrues two reward points. Consequently, he attains the corresponding reward level for two points, which is the BRONZE reward level.

First, the test is initialized by calling Initialize. The extension is activated and then a BRONZE reward level for two points or more is set up in the **Reward Level** table.

Next, a new **Customer** is created using the **LibrarySales** codeunit from the Application Test Toolkit. And then, the **LibrarySales** codeunit is used again to create and post two sales orders for the previously created customer.

Finally, to verify that the customer got the correct reward points and level, we open the **Customer Card** using its corresponding TestPage and then verify the values in the **Reward Points** and **Reward Level** fields.

There are many more areas that we look at in the sample test. See the full codeunit below for the rest of the tests.

```
codeunit 50103 "Customer Rewards Test"

{
    // [FEATURE] [Customer Rewards]

    Subtype = Test;
    TestPermissions = Disabled;

    var
        Assert: Codeunit Assert;
        LibraryLowerPermissions: Codeunit "Library - Lower Permissions";
        LibrarySales: Codeunit "Library - Sales";
        MockCustomerRewardsExtMgt: Codeunit MockCustomerRewardsExtMgt;
        ActivatedTxt: TextConst ENU = 'Customer Rewards should be activated';
        NotActivatedTxt: TextConst ENU = 'Customer Rewards should not be activated';
        BronzeLevelTxt: TextConst ENU = 'BRONZE';
        SilverLevelTxt: TextConst ENU = 'SILVER';
        GoldLevelTxt: TextConst ENU = 'GOLD';
        NoLevelTxt: TextConst ENU = 'NONE';

    [Test]

    procedure TestOnInstallLogic();
    var
        CustomerRewardsExtMgtSetup: Record "Customer Rewards Mgt. Setup";
        CustomerRewardsInstallLogic: Codeunit "Customer Rewards Install Logic";

    begin
        // [Scenario] Check default codeunit is specified for handling events on install
        // [Given] Customer Rewards Mgt. Setup table

        Initialize;

        // [When] Install logic is run
        CustomerRewardsInstallLogic.SetDefaultCustomerRewardsExtMgtCodeunit;

        // [Then] Default Customer Rewards Ext. Mgt codeunit is specified
        Assert.AreEqual(1, CustomerRewardsExtMgtSetup.Count, 'CustomerRewardsExtMgtSetup must have exactly one record.');
```

```
        CustomerRewardsExtMgtSetup.Get;

        Assert.AreEqual(Codeunit::"Customer Rewards Ext. Mgt.", CustomerRewardsExtMgtSetup."Customer Rewards Ext. Mgt. Codeunit ID", 'Codeunit does not match default');
```

```
END; --get column 25, column does not match column 25;
```

```
end;
```

```
[Test]
```

```
procedure TestCustomerRewardsWizardTermsPage();
```

```
var
```

```
    CustomerRewardsWizardTestPage: TestPage "Customer Rewards Wizard";
```

```
begin
```

```
    // [Scenario] Check Terms Page on Wizard
```

```
    // [Given] The Customer Rewards Wizard
```

```
    Initialize;
```

```
    // Using permissions that do not include SUPER
```

```
    LibraryLowerPermissions.Set0365BusFull;
```

```
    // [When] The Wizard is opened
```

```
    CustomerRewardsWizardTestPage.OpenView;
```

```
    // [Then] The terms page and fields behave as expected
```

```
    Assert.IsFalse(CustomerRewardsWizardTestPage.EnableFeature.AsBoolean, 'Enable feature should be unchecked');
```

```
    Assert.IsFalse(CustomerRewardsWizardTestPage.ActionNext.Visible, 'Next should not be visible');
```

```
    Assert.IsFalse(CustomerRewardsWizardTestPage.ActionBack.Visible, 'Back should not be visible');
```

```
    Assert.IsFalse(CustomerRewardsWizardTestPage.ActionFinish.Enabled, 'Finish should be disabled');
```

```
    CustomerRewardsWizardTestPage.EnableFeature.SetValue(true);
```

```
    Assert.IsTrue(CustomerRewardsWizardTestPage.EnableFeature.AsBoolean, 'Enable feature should be checked');
```

```
    Assert.IsTrue(CustomerRewardsWizardTestPage.ActionNext.Visible, 'Next should be visible');
```

```
    Assert.IsFalse(CustomerRewardsWizardTestPage.ActionFinish.Enabled, 'Finish should be disabled');
```

```
    CustomerRewardsWizardTestPage.Close;
```

```
end;
```

```
[Test]
```

```
procedure TestCustomerRewardsWizardActivationPageErrorsWhenNoActivationCodeEntered();
```

```
var
```

```
    CustomerRewardsExtMgt: Codeunit "Customer Rewards Ext. Mgt.";
```

```
    CustomerRewardsWizardTestPage: TestPage "Customer Rewards Wizard";
```

```
begin
```

```
    // [Scenario] Error message when user tries to activate Customer Rewards without activation code.
```

```
    // [Given] The Customer Rewards Wizard
```

```
    Initialize;
```

```
    Commit;
```

```
    // Using permissions that do not include SUPER
```

```
    LibraryLowerPermissions.Set0365BusFull;
```

```
    Assert.IsFalse(CustomerRewardsExtMgt.IsCustomerRewardsActivated, NotActivatedTxt);
```

```
    // [When] User invokes activate action without entering activation code
```

```
    OpenCustomerRewardsWizardActivationPage(CustomerRewardsWizardTestPage);
```

```
    Assert.IsTrue(CustomerRewardsWizardTestPage.ActionBack.Visible, 'Back should be visible');
```

```
    Assert.IsFalse(CustomerRewardsWizardTestPage.ActionFinish.Enabled, 'Finish should be disabled');
```

```
    // [Then] Error message displayed
```

```
    asserterror CustomerRewardsWizardTestPage.ActionActivate.Invoke;
```

```
    Assert.AreEqual(GETLASTERRORTEXT, 'Activation code cannot be blank.', 'Invalid error message.');
```

```
    Assert.IsFalse(CustomerRewardsExtMgt.IsCustomerRewardsActivated, NotActivatedTxt);
```

```
end;
```

```
[Test]
```

```
procedure TestCustomerRewardsWizardActivationPageErrorsWhenShorterActivationCodeEntered();
```

```
var
```

```
    CustomerRewardsExtMgt: Codeunit "Customer Rewards Ext. Mgt.";
```

```
    CustomerRewardsWizardTestPage: TestPage "Customer Rewards Wizard";
```

```

CustomerRewardsWizardTestPage: TestPage CustomerRewardsWizard;

begin
    // [Scenario] Error message when user tries to activate Customer Rewards with short activation code.
    // [Given] The Customer Rewards Wizard
    Initialize;
    Commit;

    // Using permissions that do not include SUPER
    LibraryLowerPermissions.Set0365BusFull;
    Assert.IsFalse(CustomerRewardsExtMgt.IsCustomerRewardsActivated, NotActivatedTxt);

    // [When] User invokes activate action after entering short activation code
    OpenCustomerRewardsWizardActivationPage(CustomerRewardsWizardTestPage);
    CustomerRewardsWizardTestPage.Activationcode.SetValue('123456');

    // [Then] Error message displayed
    asserterror CustomerRewardsWizardTestPage.ActionActivate.Invoke;
    Assert.AreEqual(GETLASTERRORTEXT, 'Activation code must have 14 digits.', 'Invalid error message.');
```

Assert.IsFalse(CustomerRewardsExtMgt.IsCustomerRewardsActivated, NotActivatedTxt);

```

end;

[Test]
procedure TestCustomerRewardsWizardActivationPageErrorsWhenLongerActivationCodeEntered();
var
    CustomerRewardsExtMgt: Codeunit "Customer Rewards Ext. Mgt.";
    CustomerRewardsWizardTestPage: TestPage "Customer Rewards Wizard";

begin
    // [Scenario] Error message when user tries to activate Customer Rewards with long activation code.
    // [Given] The Customer Rewards Wizard
    Initialize;
    Commit;

    // Using permissions that do not include SUPER
    LibraryLowerPermissions.Set0365BusFull;
    Assert.IsFalse(CustomerRewardsExtMgt.IsCustomerRewardsActivated, NotActivatedTxt);

    // [When] User invokes activate action after entering long activation code
    OpenCustomerRewardsWizardActivationPage(CustomerRewardsWizardTestPage);
    CustomerRewardsWizardTestPage.Activationcode.SetValue('123456789012345');

    // [Then] Error message displayed
    asserterror CustomerRewardsWizardTestPage.ActionActivate.Invoke;
    Assert.AreEqual(GETLASTERRORTEXT, 'Activation code must have 14 digits.', 'Invalid error message.');
```

Assert.IsFalse(CustomerRewardsExtMgt.IsCustomerRewardsActivated, NotActivatedTxt);

```

end;

[Test]
procedure TestCustomerRewardsWizardActivationPageErrorsWhenInvalidActivationCodeEntered();
var
    CustomerRewardsExtMgt: Codeunit "Customer Rewards Ext. Mgt.";
    CustomerRewardsWizardTestPage: TestPage "Customer Rewards Wizard";

begin
    // [Scenario] Error message when user tries to activate Customer Rewards with invalid activation code.
    // [Given] The Customer Rewards Wizard
    Initialize;
    Commit;

    // Using permissions that do not include SUPER
    LibraryLowerPermissions.Set0365BusFull;
    Assert.IsFalse(CustomerRewardsExtMgt.IsCustomerRewardsActivated, NotActivatedTxt);
    MockCustomerRewardsExtMgt.MockActivationResponse(false);

    // [When] User invokes activate action after entering invalid but correct length activation code
    OpenCustomerRewardsWizardActivationPage(CustomerRewardsWizardTestPage);
    CustomerRewardsWizardTestPage.Activationcode.SetValue('12345678901234');
```

```

        // [Then] Error message displayed
        asserterror CustomerRewardsWizardTestPage.ActionActivate.Invoke;
        Assert.AreEqual(GETLASTERRORTEXT, 'Activation failed. Please check the activation code you entered.',
'Invalid error message.');
```

```

        Assert.IsFalse(CustomerRewardsExtMgt.IsCustomerRewardsActivated, NotActivatedTxt);
    end;

[Test]
procedure TestCustomerRewardsWizardActivationPageDoesNotErrorWhenValidActivationCodeEntered();
var
    CustomerRewardsExtMgt: Codeunit "Customer Rewards Ext. Mgt.";
    CustomerRewardsWizardTestPage: TestPage "Customer Rewards Wizard";

begin
    // [Scenario] Customer Rewards is activated when user enters valid activation code.
    // [Given] The Customer Rewards Wizard
    Initialize;
    Commit;

    // Using permissions that do not include SUPER
    LibraryLowerPermissions.Set0365BusFull;
    Assert.IsFalse(CustomerRewardsExtMgt.IsCustomerRewardsActivated, NotActivatedTxt);
    MockCustomerRewardsExtMgt.MockActivationResponse(true);

    // [When] User invokes activate action after entering valid activation code
    OpenCustomerRewardsWizardActivationPage(CustomerRewardsWizardTestPage);
    CustomerRewardsWizardTestPage.Activationcode.SetValue('12345678901234');
    CustomerRewardsWizardTestPage.ActionActivate.Invoke;
    CustomerRewardsWizardTestPage.Close;

    // [Then] Customer Rewards is activated
    Assert.IsTrue(CustomerRewardsExtMgt.IsCustomerRewardsActivated, ActivatedTxt);
end;

[Test]
procedure TestRewardsLevelListPageDoesNotOpenWhenNotActivated();
var
    CustomerRewardsExtMgt: Codeunit "Customer Rewards Ext. Mgt.";
    RewardLevelListTestPage: TestPage "Rewards Level List";

begin
    // [Scenario] Error opening Reward Level Page when Customer Rewards is not activated
    // [Given] Unactivated Customer Rewards
    Initialize;
    Commit;

    // Using permissions that do not include SUPER
    LibraryLowerPermissions.Set0365BusFull;
    Assert.IsFalse(CustomerRewardsExtMgt.IsCustomerRewardsActivated, NotActivatedTxt);

    // [When] User opens Reward Level Page
    // [Then] Error message
    asserterror RewardLevelListTestPage.OpenView;
    Assert.AreEqual(GETLASTERRORTEXT, 'Customer Rewards is not activated', 'Invalid error message.');
```

```

end;

[Test]
procedure TestRewardsLevelListPageOpensWhenActivated();
var
    CustomerRewardsExtMgt: Codeunit "Customer Rewards Ext. Mgt.";
    RewardLevelListTestPage: TestPage "Rewards Level List";

begin
    // [Scenario] Reward Level Page opens when Customer Rewards is activated
    // [Given] Activated Customer Rewards
    Initialize;
    Commit;

    // Using permissions that do not include SUPER

```

```

    LibraryLowerPermissions.Set0365BusFull;
    Assert.IsFalse(CustomerRewardsExtMgt.IsCustomerRewardsActivated, NotActivatedTxt);
    ActivateCustomerRewards;
    Assert.IsTrue(CustomerRewardsExtMgt.IsCustomerRewardsActivated, ActivatedTxt);

    // [When] User opens Reward Level Page
    // [Then] No error
    RewardLevelListTestPage.OpenView;
end;

[Test]
procedure TestRewardLevelsActionExistsOnCustomerListPage();
var
    CustomerListTestPage: TestPage "Customer List";

begin
    // [Scenario] Reward Level action exists on customer list page
    // [Given] Customer List Page

    CustomerListTestPage.OpenView;

    // Using permissions that do not include SUPER
    LibraryLowerPermissions.Set0365BusFull;

    // [Then] Reward levels action exists on custome list page
    Assert.IsTrue(CustomerListTestPage."Reward Levels".Visible, 'Reward Levels action should be visible');
end;

[Test]

[HandlerFunctions('CustomerRewardsWizardModalPageHandler')]

procedure TestRewardLevelsActionOnCustomerListPageOpensCustomerRewardsWizardWhenNotActivated();
var
    CustomerRewardsExtMgt: Codeunit "Customer Rewards Ext. Mgt.";
    CustomerListTestPage: TestPage "Customer List";

begin
    // [Scenario] Reward Levels Action Opens Customer Rewards Wizard When Not Activated
    // [Given] Unactivated Customer Rewards
    Initialize;
    Commit;

    // Using permissions that do not include SUPER
    LibraryLowerPermissions.Set0365BusFull;
    Assert.IsFalse(CustomerRewardsExtMgt.IsCustomerRewardsActivated, NotActivatedTxt);

    // [When] User opens Customer List page and invokes action
    CustomerListTestPage.OpenView;
    CustomerListTestPage."Reward Levels".Invoke;

    // [Then] Wizard opens. Caught by CustomerRewardsWizardModalPageHandler
end;

[Test]

[HandlerFunctions('RewardsLevelList1PageHandler')]
procedure TestRewardLevelsActionOnCustomerListPageOpensRewardsLevelListPageWhenActivated();
var
    CustomerRewardsExtMgt: Codeunit "Customer Rewards Ext. Mgt.";
    CustomerListTestPage: TestPage "Customer List";

begin
    // [Scenario] Reward Levels Action Opens Reward Level Page When Activated
    // [Given] Activated Customer Rewards
    Initialize;
    Commit;

    // Using permissions that do not include SUPER

```

```

LibraryLowerPermissions.Set0365BusFull;
Assert.IsFalse(CustomerRewardsExtMgt.IsCustomerRewardsActivated, NotActivatedTxt);
ActivateCustomerRewards;
Assert.IsTrue(CustomerRewardsExtMgt.IsCustomerRewardsActivated, ActivatedTxt);

// [When] User opens Customer List page and invokes action
CustomerListTestPage.OpenView;
CustomerListTestPage."Reward Levels".Invoke;

// [Then] Wizard opens. Caught by RewardsLevelListPageHandler
end;

[Test]
procedure TestCustomerCardPageHasRewardsFields();
var
    CustomerRewardsExtMgt: Codeunit "Customer Rewards Ext. Mgt.";
    CustomerCardTestPage: TestPage "Customer Card";

begin
    // [Scenario] Customer Card Page Has Reward Fields When Opened
    // [Given] Customer Card Page

    // Using permissions that do not include SUPER
    LibraryLowerPermissions.Set0365BusFull;

    // [When] Customer card page is opened
    CustomerCardTestPage.OpenView;

    // [Then] Reward fields exist
    Assert.IsTrue(CustomerCardTestPage.RewardLevel.Visible, 'Reward Level should be visible');
    Assert.IsFalse(CustomerCardTestPage.RewardLevel.Editable, 'Reward Level should not be editable');
    Assert.IsTrue(CustomerCardTestPage.RewardPoints.Visible, 'Reward Points should be visible');
    Assert.IsFalse(CustomerCardTestPage.RewardPoints.Editable, 'Reward Points should not be editable');
end;

[Test]
procedure TestNewCustomerHasZeroRewardPointsAndNoRewardLevel();
var
    Customer: Record Customer;
    CustomerRewardsExtMgt: Codeunit "Customer Rewards Ext. Mgt.";
    CustomerCardTestPage: TestPage "Customer Card";

begin
    // [Scenario] A new customer Has Zero Reward Points And No Reward Level
    // [Given] Activated Customer Rewards
    Initialize;
    Commit;

    // Using permissions that do not include SUPER
    LibraryLowerPermissions.Set0365BusFull;
    ActivateCustomerRewards;

    // [When] New Customer
    LibrarySales.CreateCustomer(Customer);
    CustomerCardTestPage.OpenView;
    CustomerCardTestPage.GoToRecord(Customer);

    // [Then] No Reward level
    VerifyCustomerRewardLevel(CustomerCardTestPage.RewardLevel.Value, NoLevelTxt);

    // [Then] Reward Point is zero
    VerifyCustomerRewardPoints(CustomerCardTestPage.RewardPoints.AsInteger, 0);
end;

[Test]
procedure TestCustomerHasCorrectRewardPointsAfterPostedSalesOrders();
var
    Customer: Record Customer;
    CustomerRewardsExtMgt: Codeunit "Customer Rewards Ext. Mgt.";

```

```

CustomerCardTestPage: TestPage "Customer Card";

begin
    // [Scenario] Customer Has Correct Reward Points After 4 Posted Sales Orders
    // [Given] Activated Customer Rewards and Customer
    Initialize;
    Commit;

    // Using permissions that do not include SUPER
    LibraryLowerPermissions.Set0365BusFull;
    ActivateCustomerRewards;

    // New Customer
    LibrarySales.CreateCustomer(Customer);

    // [When] 4 Sales Orders
    CreateAndPostSalesOrder(Customer."No.");
    CreateAndPostSalesOrder(Customer."No.");
    CreateAndPostSalesOrder(Customer."No.");
    CreateAndPostSalesOrder(Customer."No.");

    // [Then] Customer has 4 reward points
    CustomerCardTestPage.OpenView;
    CustomerCardTestPage.GoToRecord(Customer);
    VerifyCustomerRewardPoints(CustomerCardTestPage.RewardPoints.AsInteger, 4);
end;

[Test]
procedure TestCustomerHasNoRewardLevelAfterPostedSalesOrders();
var
    Customer: Record Customer;
    CustomerRewardsExtMgt: Codeunit "Customer Rewards Ext. Mgt.";
    CustomerCardTestPage: TestPage "Customer Card";

begin
    // [Scenario] Customer Has 1 Reward Point and No Reward Level After 1 Posted Sales Orders
    // [Scenario] Because Lowest Level requires at least 2 points
    // [Given] Activated Customer Rewards, Customer, Bronze level for 2 points and above
    Initialize;
    Commit;

    // Using permissions that do not include SUPER
    LibraryLowerPermissions.Set0365BusFull;
    ActivateCustomerRewards;
    AddRewardLevel(BronzeLevelTxt, 2); // 2 points required for BRONZE level

    // New Customer
    LibrarySales.CreateCustomer(Customer);
    CustomerCardTestPage.OpenView;
    CustomerCardTestPage.GoToRecord(Customer);

    // Verify 0 points and no reward level before sales order
    VerifyCustomerRewardPoints(CustomerCardTestPage.RewardPoints.AsInteger, 0);
    VerifyCustomerRewardLevel(CustomerCardTestPage.RewardLevel.Value, NoLevelTxt);

    // [When] 1 Sales Order
    CreateAndPostSalesOrder(Customer."No.");

    // [Then] Customer has 1 points and no reward level after sales order
    CustomerCardTestPage.GoToRecord(Customer);
    VerifyCustomerRewardPoints(CustomerCardTestPage.RewardPoints.AsInteger, 1);
    VerifyCustomerRewardLevel(CustomerCardTestPage.RewardLevel.Value, NoLevelTxt);
end;

[Test]
procedure TestCustomerHasBronzeRewardLevelAfterPostedSalesOrders();
var
    Customer: Record Customer;
    CustomerRewardsExtMgt: Codeunit "Customer Rewards Ext. Mgt.";

```

```

CustomerRewardExtMgt: Codeunit "Customer Rewards Ext. Mgt.";
CustomerCardTestPage: TestPage "Customer Card";

begin
    // [Scenario] Customer Has 2 Reward Points and Bronze Reward Level After 2 Posted Sales Orders
    // [Scenario] Because Bronze Level requires at least 2 points
    // [Given] Activated Customer Rewards, Customer, Bronze level for 2 points and above
    Initialize;
    Commit;

    // Using permissions that do not include SUPER
    LibraryLowerPermissions.Set0365BusFull;
    ActivateCustomerRewards;
    AddRewardLevel(BronzeLevelTxt, 2); // 2 points required for BRONZE level

    // New Customer
    LibrarySales.CreateCustomer(Customer);

    // [When] 2 Sales Order
    CreateAndPostSalesOrder(Customer."No.");
    CreateAndPostSalesOrder(Customer."No.");

    // [Then] Customer has 2 points and bronze reward level
    CustomerCardTestPage.OpenView;
    CustomerCardTestPage.GoToRecord(Customer);
    VerifyCustomerRewardPoints(CustomerCardTestPage.RewardPoints.AsInteger, 2);
    VerifyCustomerRewardLevel(CustomerCardTestPage.RewardLevel.Value, BronzeLevelTxt);
end;

[Test]
procedure TestCustomerHasSilverRewardLevelAfterPostedSalesOrders();
var
    Customer: Record Customer;
    CustomerRewardsExtMgt: Codeunit "Customer Rewards Ext. Mgt.";
    CustomerCardTestPage: TestPage "Customer Card";

begin
    // [Scenario] Customer Has 3 Reward Points and Silver Reward Level After 3 Posted Sales Orders
    // [Scenario] Because Silver Level requires at least 3 points
    // [Given] Activated Customer Rewards, Customer, Bronze level from 2 points, Silver level from 3
points
    Initialize;
    Commit;

    // Using permissions that do not include SUPER
    LibraryLowerPermissions.Set0365BusFull;
    ActivateCustomerRewards;
    AddRewardLevel(BronzeLevelTxt, 2); // 2 points required for BRONZE level
    AddRewardLevel(SilverLevelTxt, 3); // 3 points required for SILVER level

    // New Customer
    LibrarySales.CreateCustomer(Customer);

    // 2 Sales Order
    CreateAndPostSalesOrder(Customer."No.");
    CreateAndPostSalesOrder(Customer."No.");

    // Verify 2 points and bronze reward level
    CustomerCardTestPage.OpenView;
    CustomerCardTestPage.GoToRecord(Customer);
    VerifyCustomerRewardPoints(CustomerCardTestPage.RewardPoints.AsInteger, 2);
    VerifyCustomerRewardLevel(CustomerCardTestPage.RewardLevel.Value, BronzeLevelTxt);

    // [When] 3rd Sales Order
    CreateAndPostSalesOrder(Customer."No.");

    // [Then] Customer has 3 points and silver reward level
    CustomerCardTestPage.GoToRecord(Customer);
    VerifyCustomerRewardPoints(CustomerCardTestPage.RewardPoints.AsInteger, 3);
    VerifyCustomerRewardLevel(CustomerCardTestPage.RewardLevel.Value, SilverLevelTxt);

```

```

VerifyCustomerRewardLevel(CustomerCardTestPage.RewardLevel.Value, SilverLevelTxt),
end;

[Test]
procedure TestCustomerHasGoldRewardLevelAfterPostedSalesOrders();
var
    Customer: Record Customer;
    CustomerRewardsExtMgt: Codeunit "Customer Rewards Ext. Mgt.";
    CustomerCardTestPage: TestPage "Customer Card";

begin
    // [Scenario] Customer Has 4 Reward Points and Gold Reward Level After 4 Posted Sales Orders
    // [Scenario] Because Gold Level requires at least 4 points
    // [Given] Activated Customer Rewards, Customer
    // [Given] Bronze level from 2 points, Silver level from 3 points, Gold level from 4 points
    Initialize;
    Commit;

    // Using permissions that do not include SUPER
    LibraryLowerPermissions.Set0365BusFull;
    ActivateCustomerRewards;
    AddRewardLevel(BronzeLevelTxt, 2); // 2 points required for BRONZE level
    AddRewardLevel(SilverLevelTxt, 3); // 3 points required for SILVER level
    AddRewardLevel(GoldLevelTxt, 4); // 4 points required for GOLD level

    // New Customer
    LibrarySales.CreateCustomer(Customer);

    // 3 Sales Order
    CreateAndPostSalesOrder(Customer."No.");
    CreateAndPostSalesOrder(Customer."No.");
    CreateAndPostSalesOrder(Customer."No.");

    // Verify 3 points and silver reward level
    CustomerCardTestPage.OpenView;
    CustomerCardTestPage.GoToRecord(Customer);
    VerifyCustomerRewardPoints(CustomerCardTestPage.RewardPoints.AsInteger, 3);
    VerifyCustomerRewardLevel(CustomerCardTestPage.RewardLevel.Value, SilverLevelTxt);

    // [When] 4th Sales Order
    CreateAndPostSalesOrder(Customer."No.");

    // [Then] Customer has 4 points and gold reward level
    CustomerCardTestPage.GoToRecord(Customer);
    VerifyCustomerRewardPoints(CustomerCardTestPage.RewardPoints.AsInteger, 4);
    VerifyCustomerRewardLevel(CustomerCardTestPage.RewardLevel.Value, GoldLevelTxt);
end;

local procedure OpenCustomerRewardsWizardActivationPage(VAR CustomerRewardsWizardTestPage: TestPage
"Customer Rewards Wizard");
begin
    CustomerRewardsWizardTestPage.OpenView;
    CustomerRewardsWizardTestPage.EnableFeature.SetValue(true);
    CustomerRewardsWizardTestPage.ActionNext.Invoke;
end;

local procedure Initialize();
var
    ActivationCodeInfo: Record "Activation Code Information";
    RewardLevel: Record "Reward Level";
    Customer: Record Customer;

begin
    Customer.ModifyAll(RewardPoints, 0);
    ActivationCodeInfo.DeleteAll;
    RewardLevel.DeleteAll;
    UnbindSubscription(MockCustomerRewardsExtMgt);
    BindSubscription(MockCustomerRewardsExtMgt);
    MockCustomerRewardsExtMgt.Setup;

```

```

end;

local procedure ActivateCustomerRewards();
var
    ActivationCodeInfo: Record "Activation Code Information";

begin
    ActivationCodeInfo.Init;
    ActivationCodeInfo.ActivationCode := '12345678901234';
    ActivationCodeInfo."Date Activated" := Today;
    ActivationCodeInfo."Expiration Date" := CALCDATE('<1Y>', Today);
    ActivationCodeInfo.Insert;
end;

local procedure CreateAndPostSalesOrder(SellToCustomerNo: Code[20]);
var
    SalesHeader: Record "Sales Header";
    SalesLine: Record "Sales Line";
    LibraryRandom: Codeunit "Library - Random";
    SalesOrderTestPage: TestPage "Sales Order";

begin
    LibrarySales.CreateSalesHeader(SalesHeader, SalesHeader."Document Type"::Order, SellToCustomerNo);
    LibrarySales.CreateSalesLine(SalesLine, SalesHeader, SalesLine.Type::Item, '', 1);
    SalesLine.VALIDATE("Unit Price", LibraryRandom.RandIntInRange(5000, 10000));
    SalesLine.MODIFY(TRUE);
    LibrarySales.PostSalesDocument(SalesHeader, true, true);
end;

local procedure AddRewardLevel(Level: Text; MinPoints: Integer);
var
    RewardLevel: Record "Reward Level";

begin
    if RewardLevel.Get(Level) then begin
        RewardLevel."Minimum Reward Points" := MinPoints;
        RewardLevel.Modify;
    end else begin
        RewardLevel.Init;
        RewardLevel.Level := Level;
        RewardLevel."Minimum Reward Points" := MinPoints;
        RewardLevel.Insert;
    end;
end;

local procedure VerifyCustomerRewardLevel(ExpectedLevel: Text; ActualLevel: Text);
begin
    Assert.AreEqual(ExpectedLevel, ActualLevel, 'Reward Level should be the same.');
```

end;

```

local procedure VerifyCustomerRewardPoints(ExpectedPoints: Integer; ActualPoints: Integer);
begin
    Assert.AreEqual(ExpectedPoints, ActualPoints, 'Reward Points should be the same.');
```

end;

```

[ModalPageHandler]
procedure CustomerRewardsWizardModalPageHandler(var CustomerRewardsWizard: TestPage "Customer Rewards
Wizard");
begin
end;

[PageHandler]
procedure RewardsLevelList1PageHandler(var RewardsLevelList: TestPage "Rewards Level List");
begin
end;
}

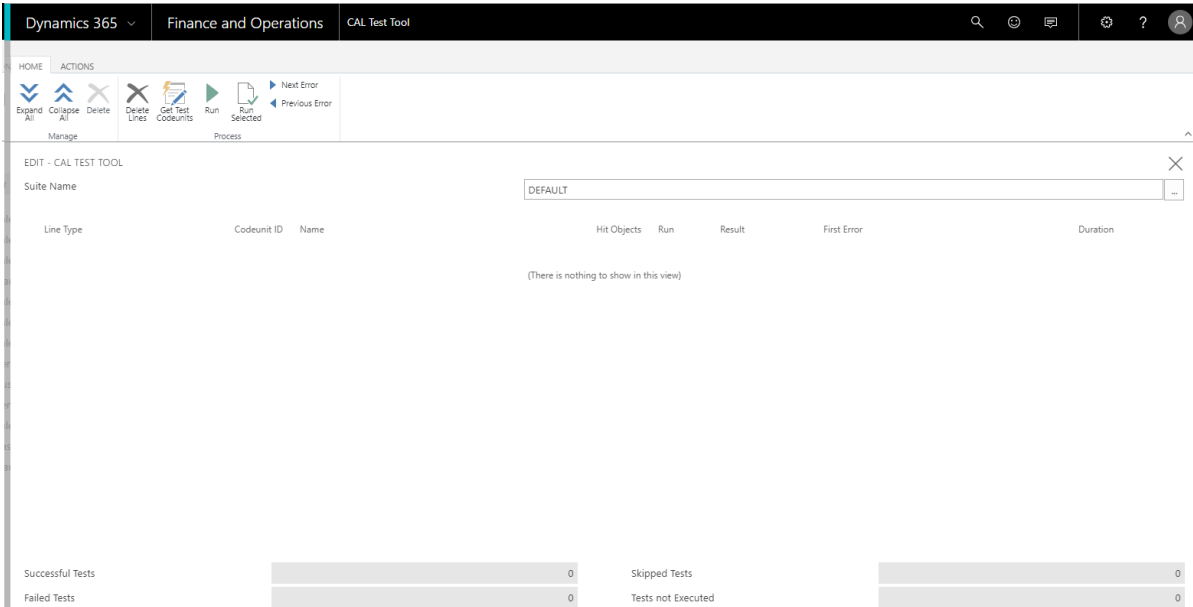
```

At this point you can publish and run your tests on your tenant by pressing Ctrl+F5.

Run the tests

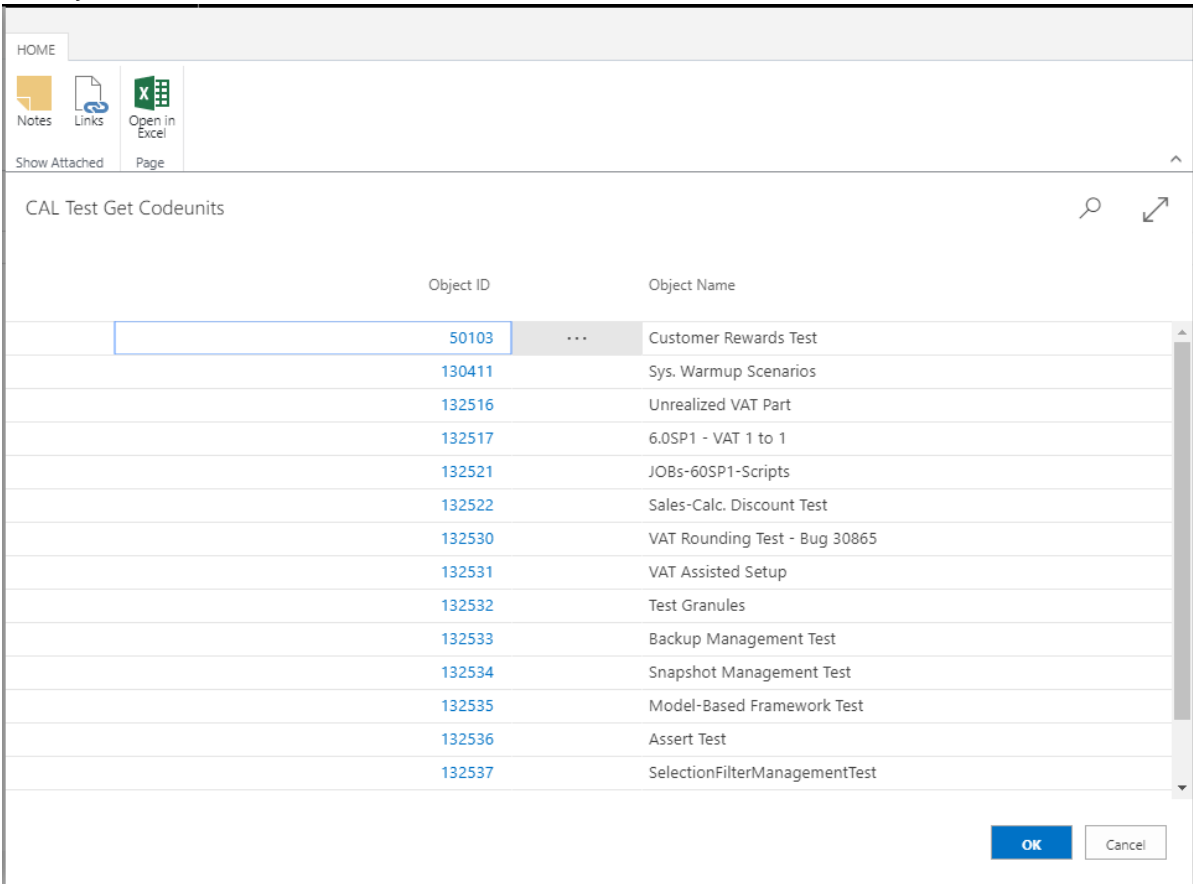
In order to run the tests, follow the steps below.

1. Open the **Test Tool** page (130401).



2. Choose **Get Test Codeunits** and then choose **Select Test Codeunits**.

3. Select your test codeunits and then choose the **OK** button.



You can now see all the test methods from your test codeunits.

4. Now, choose **Run** or **Run Selected** to run all the tests in the test codeunit or only the selected tests. The **Result** column indicates whether a test was a SUCCESS or FAILURE. A summary is also presented at the bottom of the page.

HOME

ACTIONS

Expand All

Collapse All

Delete

Delete Lines

Get Test Codeunits

Run

Run Selected

Next Error

Previous Error

Manage

Process

EDIT - CAL TEST TOOL - DEFAULT

Suite Name

DEFAULT

Line Type	Codeunit ID	Name	Hit Objects	Run	Result	First Error	Duration
Codeunit	50103	Customer Rewards Test		✓	Success		1 minute 8 seconds 436 ...
Function	50103	TestCustomerRewardsWizardActivationPageErrors...		✓	Success		373 milliseconds
Function	50103	TestCustomerCardPageHasRewardsFields		✓	Success		850 milliseconds
Function	50103	TestRewardsLevelListPageOpensWhenActivated		✓	Success		317 milliseconds
Function	50103	TestRewardsLevelListPageDoesNotOpenWhenNot...		✓	Success		414 milliseconds
Function	50103	TestRewardLevelsActionOnCustomerListPageOpen...		✓	Success		890 milliseconds
Function	50103	TestCustomerRewardsWizardActivationPageDoesN...		✓	Success		384 milliseconds
Function	50103	TestCustomerRewardsWizardActivationPageErrors...		✓	Success		357 milliseconds
Function	50103	TestNewCustomerHasZeroRewardPointsAndNoRe...		✓	Success		7 seconds 830 milliseco...
Function	50103	TestCustomerHasGoldRewardLevelAfterPostedSale...		✓	Success		13 seconds 546 millise...
Function	50103	TestCustomerHasNoRewardLevelAfterPostedSales...		✓	Success		11 seconds 793 millise...
Function	50103	TestRewardLevelsActionExistsOnCustomerListPage		✓	Success		323 milliseconds
Function	50103	TestCustomerRewardsWizardActivationPageErrors...		✓	Success		410 milliseconds
Function	50103	TestCustomerRewardsWizardTermsPage		✓	Success		400 milliseconds
Function	50103	TestCustomerHasBronzeRewardLevelAfterPostedSales...		✓	Success		5 seconds 656 millise...
Successful Tests			19	Skipped Tests			0
Failed Tests			0	Tests not Executed			0

Failing Tests

Let us look at what to do if you have a failing test. To create a failing test, we will modify the **SetDefaultCustomerRewardsExtMgtCodeunit** method in codeunit 50100 **Customer Rewards Install Logic** to the following:

```

procedure SetDefaultCustomerRewardsExtMgtCodeunit();
var
    CustomerRewardsExtMgtSetup: Record "Customer Rewards Mgt. Setup";

begin
    CustomerRewardsExtMgtSetup.DeleteAll;
    CustomerRewardsExtMgtSetup.Init;
    // Default Customer Rewards Ext. Mgt codeunit to use for handling events
    // Changing
    // CustomerRewardsExtMgtSetup."Customer Rewards Ext. Mgt. Codeunit ID" := Codeunit::"Customer Rewards
Ext. Mgt.";
    // To
    CustomerRewardsExtMgtSetup."Customer Rewards Ext. Mgt. Codeunit ID" := 0;
    CustomerRewardsExtMgtSetup.Insert;
end;

```

Now, anytime the **SetDefaultCustomerRewardsExtMgtCodeunit** method in the install codeunit is run, the **Customer Rewards Ext. Mgt. Codeunit ID** in the **Customer Rewards Mgt. Setup** table will be set to 0.

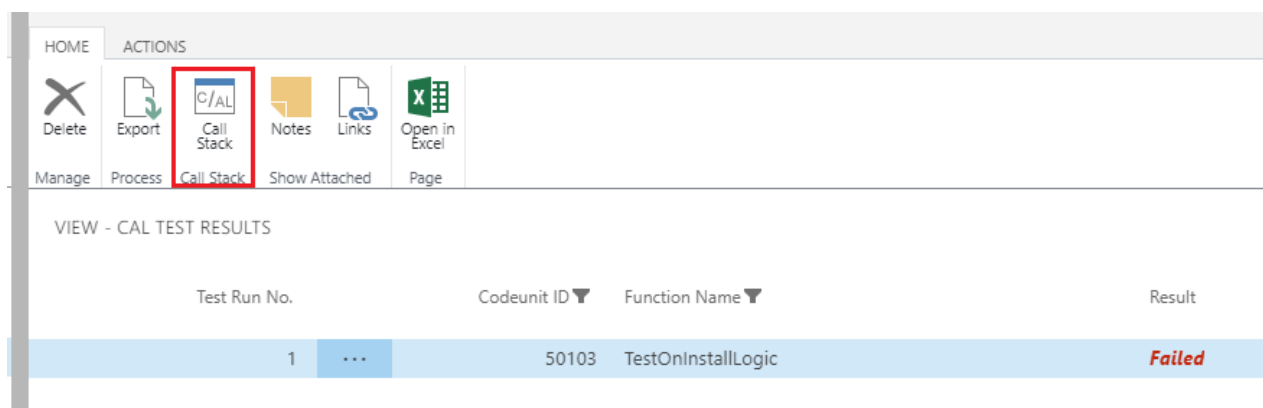
Press Ctrl+F5 to publish the updated tests to your tenant and then run them.

Function	50103	TestCustomerHasBronzeRewardLevelAfterPostedSalesOr...	✓	Success		5 seconds 357 millisecond
Function	50103	TestOnInstallLogic	✗	Failure	Assert.AreEqual failed. Expected:<50101> (Integer). Act...	234 milliseconds
Function	50103	TestCustomerHasCorrectRewardPointsAfterPostedSales...	✓	Success		5 seconds 656 millisecond

The test TestOnInstallLogic should now have a Failure result with the error message:

"Assert.AreEqual failed. Expected:<50101> (Integer). Actual:<0> (Integer). Codeunit does not match default."

The error message shows that the actual result in one of our Assert statements differed from what was expected. According to the error message, the Assert statement was expecting a value of 50101 but actually got a value of 0. We can also tell where in our code this is happening because of the message; "Codeunit does not match default", which we defined earlier when we wrote our tests. If we had no idea where the error occurred, we can click on the error message to open the **Test Results** page and then choose the **Call Stack** action.



Choosing the **Call Stack** action will give you a message alert that contains an ordered list of method calls up to the one that caused the error.

i Assert(CodeUnit 130000).AreEqual line 2
 "Customer Rewards Test"(CodeUnit 50103).TestOnInstallLogic_Scope_1248196953 line 35
 "CAL Test Runner"(CodeUnit 130400).RunTests line 25
 "CAL Test Runner"(CodeUnit 130400).OnRun(Trigger) line 7
 "CAL Test Management"(CodeUnit 130401).RunSuite line 3
 "CAL Test Management"(CodeUnit 130401).RunSelected line 27
 "CAL Test Tool"(Page 130401)."RunSelected - OnAction"(Trigger) line 3

OK

The list of method calls is arranged from the most recent at the top to the oldest at the bottom. In our example, we can tell that the `Assert(CodeUnit 130000).AreEqual` (the first on the list) was the last method to be run, indicating where the error was found. Because we did not modify the Assert codeunit, then the wrong values or results must have been passed to it. The next item on the list,

`"Customer Rewards Test"(CodeUnit 50103).TestOnInstallLogic_Scope_1248196953` line 35 points to the method that was run before the final one that caused the error. This time, it is in the TestOnInstallLogic method of codeunit 50103 Customer Rewards Test after line 35.

```

35 CustomerRewardsExtMgtSetup.Get;
36 Assert.AreEqual(Codeunit::"Customer Rewards Ext. Mgt.", CustomerRewardsExtMgtSetup."Customer Rewards Ext. Mgt. Codeunit ID", 'Codeunit does not match default');
37 end;

```

On line 36 of codeunit 50103 **Customer Rewards Test**, we can see the Assert statement that throws the error. We tested that the result should be `Codeunit::"Customer Rewards Ext. Mgt."` which is 50101, when our install logic is run, however, the result of the test indicated that we got a result of 0. This implies that our install logic is not working as expected. To fix this, we need to examine all the previous lines of code in the method to figure out where we went wrong. This will lead us to line 31, where the **SetDefaultCustomerRewardsExtMgtCodeunit** method call is made.

```

31 CustomerRewardsInstallLogic.SetDefaultCustomerRewardsExtMgtCodeunit;

```

When you go into the **SetDefaultCustomerRewardsExtMgtCodeunit** method, codeunit 50100 **Customer Rewards Install Logic**, you will see the change we made to cause the test to fail. Revert it so that

`CustomerRewardsExtMgtSetup."Customer Rewards Ext. Mgt. Codeunit ID"` now stores `Codeunit::"Customer Rewards Ext. Mgt."`, instead of 0. Publish the updated extension and tests to your tenant and run the tests again. The test **TestOnInstallLogic** should pass now because the actual result matches what is expected.

Conclusion

At this point, the Customer Rewards sample extension can be published and installed on your sandbox.

See Also

[Developing Extensions](#)

[Getting Started with AL](#)

[How to: Publish and Install an Extension](#)

[Converting Extensions V1 to Extensions V2](#)

Business Central Web Services

3/31/2019 • 5 minutes to read

Business Central supports two types of web services: SOAP and OData. Web services are a lightweight, industry-standard way to make application functionality available to a variety of external systems and users. Developers can create and publish functionality as web services, where they expose pages, codeunits, or queries, and even enhance a page web service by using an extension codeunit. When Business Central objects are published as web services, they are immediately available on the network.

Business Central web services are stateless and do not preserve the values of global variables or single-instance codeunits between calls.

Comparing SOAP and OData Web Services

Developers planning to create Microsoft Dynamics NAV web services may need to decide which type of web service is better suited to their needs. The following table shows the types of web service applications that you can create for the web service protocols.

	SOAP WEB SERVICES	ODATA WEB SERVICES
Pages	Yes: Create, Read, Update, and Delete operations (CRUD)	Yes: Create, Read, Update, and Delete operations (CRUD)
Codeunits	Yes	No
Queries	No	Yes: Read-only

Business Central supports OData web services in addition to the SOAP web services that have been available since Microsoft Dynamics NAV 2009.

SOAP Web Services

SOAP web services allow full flexibility for building operation-centric services. They provide industry standard interoperability. Windows Communication Framework (WCF) has supported SOAP services since its initial release in .NET Framework 3.0, and later releases of the .NET Framework have added additional support and default bindings to make it easier to build SOAP services using WCF.

The most common type of messaging pattern in SOAP is the Remote Procedure Call (RPC), where one network node (the client) sends a request message to another node (the server), and the server sends a response message to the client.

OData Web Services

The OData standard is well suited for web service applications that require a uniform, flexible, general purpose interface for exposing create retrieve update delete (CRUD) operations on a tabular data model to clients. OData is less suited for applications that are primarily RPC-oriented or in which data operations are constrained to certain prescribed patterns. OData supports Representational State Transfer (REST)-based data services, which enable resources, identified using Uniform Resource Identifiers (URIs), and defined in an abstract data model (EDM), to be published and edited by web clients within corporate networks and across the Internet using simple Hypertext Transfer Protocol (HTTP) messages. OData services are lightweight, with functionality often referenced directly in the URI.

Whereas SOAP web services expose a WSDL document, OData web services expose an EDMX document

containing metadata for all published web services.

OData is supported in PowerPivot, a data-analysis add-in to Microsoft Excel that provides enhanced Business Intelligence capabilities. PowerPivot supports sharing and collaboration on user-generated business intelligence solutions in a Microsoft SharePoint Server environment. For more information about PowerPivot, see <http://www.powerpivot.com/>.

The extensions to the Atom Publishing Protocol defined in the AtomPub extensions to the OData protocol documentation (which you can download [here](#)) describe how REST-based data services can enable resources, identified using URIs and defined in an abstract data model (EDM), to be published and edited by web clients within corporate networks and across the Internet using simple HTTP messages.

In addition to the AtomPub format, the OData implementation in Business Central also supports the JSON format, a somewhat less verbose format that may perform better in low-bandwidth environments.

Page Web Services

When you expose a page as an OData web service, you can query that data to return a service metadata (EDMX) document, an AtomPub document, or a JavaScript Object Notation (JSON) document. You can also write back to the database if the exposed page is writable. For more information, see [OData Web Services](#).

When you expose a page as a SOAP web service, you expose a default set of operations that you can use to manage common operations such as Create, Read, Update, and Delete. Page-based web services offer built-in optimistic concurrency management. Each operation call in a page-based web service is managed as a single transaction.

For SOAP services, you can also use extension codeunits to extend the default set of operations that are available on a page. Adding an extension codeunit to a page is useful if you want to perform operations other than the standard Create, Read, Update, and Delete operations. The benefit of adding an extension codeunit to a page is that you can make the web service complete by adding operations that are logical to that service. Those operations can use the same object identification principle as the basic page operations.

Codeunit Web Services

For SOAP services only, codeunit web services provide you with the most control and flexibility. When a codeunit is exposed as a web service, all functions defined in the codeunit are exposed as operations.

Query Web Services

When you expose a Business Central query as an OData web service, you can query that data to return a service metadata (EDMX) document or an AtomPub document. For more information about how to create and use Business Central queries, see [Query Object](#).

Web Services and Regional Settings

Data is formatted according to the value of the **Services Language** setting for the relevant Business Central Server instance. The default value is **en-us**. This means that Business Central Server interprets all incoming data as the specified culture, such as dates and amounts.

If you know that the **Services Language** setting is always en-us, for example, your code can be based on that assumption. In a multilanguage environment, you will see more predictable transformations of data if data that is transmitted through web services is in a consistent culture.

Similarly, you can use the **ServicesOptionFormat** setting to specify how Business Central Server must understand option values. If you set the **ServicesOptionFormat** setting to *OptionString*, Business Central Server understands option values as the *name* of the option value, which is always en-us. If you set the setting to *OptionCaption*, web service data will be interpreted in the language specified by the **Services Language** setting.

Web Services in Multitenant Deployments

If your Business Central solution is used in a multitenant deployment architecture, you must make sure that any code that generates or consumes a web service specifies the relevant tenant. Web services are set up in the application, but typically you want to consume company-specific and tenant-specific data.

If you use the GETURL method, the generated URL will automatically apply to the user's tenant ID. For more information, see [GETURL Method](#).

The URL for accessing a web service in a multitenant deployment must specify the tenant ID in one of two ways: As a query parameter, or as a host name. If you use host names for tenants, the host name must be specified as an alternative ID.

For example, the following URL consumes the **Customer** ODATA web service for a specific tenant:

```
http://localhost:7048/BC130/OData/Company('CRONUS-International-Ltd.']/Customer?Tenant=Tenant1
```

For more information, see [Multitenant Deployment Architecture](#).

See Also

[Publish a Web Service](#)

[Web Services Overview](#)

[SOAP Web Service URIs](#)

[Using SystemService to Find Companies](#)

[Basic Page Operations](#)

[Web Services Best Practices](#)

[Configuring Business Central Server](#)


Publishing a Web Service

3/31/2019 • 3 minutes to read

You can set up a web service in the Business Central Web client or Dynamics NAV Client connected to Business Central. You must then publish the web service so that it is available to service requests over the network. Users can discover web services by pointing a browser at the computer that is running Business Central Server and requesting a list of available services. When you publish a web service, it is immediately available over the network for authenticated users. All authorized users can access metadata for Business Central web services, but only users who have sufficient Business Central permissions can access actual data.

Creating and Publishing a Web Service

The following steps explain how to create and publish a web service.

1. Open the client.
2. Choose the  icon, enter **Web Services**, and then choose the related link.
3. In the **Web Services** page, choose **New**.
4. In the **Object Type** column, select **Codeunit**, **Page**, or **Query**.

NOTE

Codeunit and **Page** are valid types for SOAP web services. **Page** and **Query** are valid types for OData web services.

5. In the **Object ID** column, select the object ID of the object that you want to expose. For example, to expose the customer card as a web service, enter **21**.

If the database contains multiple companies, you can choose an object ID that is specific to one of the companies.

6. In the **Service Name** field, assign a name to the web service. For example, if you expose the customer card as a web service, enter **Customers**.

- **Codeunit** and **Page** are valid types for SOAP web services. **Page** and **Query** are valid types for OData web services.
- If the database contains multiple companies, you can choose an object ID that is specific to one of the companies.
- The service name is visible to consumers of your web service and is the basis for identifying and distinguishing web services, so you should make the name meaningful.
- If you are setting up integration with Microsoft Outlook using codeunit 5313, then you must use **DynamicsNAVsynchOutlook** as the service name.

7. Select the check box in the **Published** column.

When you publish the web service, in the **OData URL** and **SOAP URL** fields, you can see the URLs that are generated for the web service. You can test the web service immediately by choosing the links in the **OData URL** and **SOAP URL** fields. Optionally, you can copy the value of the field and save it for later use.

After you publish a web service, it is available on the Business Central Server computer that you were connected to when you published. The web service is available across all Business Central Server instances running on the server computer.

You can verify the availability of that web service by using a browser, or you can choose the link in the **OData URL** and **SOAP URL** fields in the **Web Services** window. The following procedure illustrates how you can verify the availability of the web service for later use.

Verify the availability of a web service

1. In your browser, enter the relevant URL. The following table illustrates the types of URLs that you can enter. For SOAP web services, use the following format for your URI.

WEB SERVICE TYPE	SYNTAX	EXAMPLE
SOAP	<code>https://Server:SOAPWebServicePort/ ServerInstance/WS/CompanyName/ services/</code>	<code>https://localhost:7047/BC140/WS/CR ONUS International Ltd./services/</code>
OData	<code>https://Server:ODataWebServicePort /ServerInstance/OData/Company('Co mpanyName')</code>	<code>https://localhost:7048/BC140/OData /Company('CRONUS International Ltd.')</code>

The company name is case-sensitive.

2. Review the information that is displayed in the browser. Verify that you can see the name of the web service that you have created.

When you access a web service, and you want to write data back to Business Central , you must specify the company name. You can specify the company as part of the URI as shown in the examples, or you can specify the company as part of the query parameters. For example, the following URIs point to the same OData web service and are both valid URIs.

```
https://localhost:7048/<serverinstance>/OData/Company('CRONUS International Ltd.)/Customer
```

```
https://localhost:7048/<serverinstance>/OData/Customer?company='CRONUS International Ltd.'
```

Handling UI Interaction When Working with Web Services

4/4/2019 • 2 minutes to read

Whether you are publishing or consuming web services, exceptions and dialog boxes that may be displayed while code runs must be handled correctly. Exceptions must be handled to prevent the system from ending the web service client execution. You can handle exceptions in the following ways:

- Writing conditional code inside Business Central.
- Writing the code in the web service client application.

The most robust solution is to use both methods.

Publishing Web Services

When publishing a web service, you must make sure that the code that you are publishing does not assume the ability to interact with a user through the UI. You can use the [GUIALLOWED Method](#) to suppress the UI. For example, you can use this method to determine whether a codeunit is being called from the client or from a web service client. You must make sure to suppress errors when a codeunit is called from a web service client.

When implementing a conditional code check in Business Central, you should implement the check only around code that could cause an error. You should not encapsulate the whole business logic.

NOTE

The server returns the following exception when trying to invoke a dialog UI through a web service:

Microsoft.Dynamics.Nav.Types.Exceptions.NavNCLCallbackNotAllowedException: Callback functions are not allowed.

AL Keywords That Can Cause Faults or Exceptions

Variables of the [Dialog Data Type](#) or any of the methods listed as dialog methods can cause callback not allowed exceptions when they are called from a web service application. The [Message Method \(Dialog\)](#) is the only method in this category that does not cause an exception.

Other keywords that you should not use are:

- PAGE.RUN
- PAGE.RUNMODAL
- ACTIVATE
- REPORT.RUN
- REPORT.RUNMODAL
- HYPERLINK
- FILE.UPLOAD
- FILE.DOWNLOAD

You should also avoid operations on client-side Automation and .NET Framework interoperability objects.

Consuming Web Services

You must handle exceptions in client code that calls a Business Central web service. Appropriate exception capturing code should be included around any call to a Business Central web service.

See Also

[Web Services Overview](#)

[Publish a Web Service](#)

Managing Time Zones with Web Services

3/31/2019 • 2 minutes to read

Business Central Server provides a **Services Default Time Zone** setting for defining the time zone in which web service calls run. This setting affects both SOAP and OData web services, in addition to NAS Services.

Time Zone Configuration

You can configure the Services Default Time Zone using the [Server Administration Tool](#), [Business Central Windows PowerShell Cmdlets] (<https://docs.microsoft.com/en-us/powershell/business-central/overview>), or by directly editing CustomSettings.config, the configuration file for the relevant Business Central Server instance. The following table describes the possible values for the **Services Default Time Zone** setting.

VALUE	DESCRIPTION
UTC	Specifies that all business logic for services on the server instance runs in Coordinated Universal Time (UTC). This is the default value. This is how web services business logic was handled in Microsoft Dynamics NAV 2009 SP1 and Microsoft Dynamics NAV 2009.
Server Time Zone	Specifies that services use the time zone of the computer that is running Business Central Server.
<i>ID of any Windows time zone</i>	Specifies that services use a Windows time zone as defined in the system registry under HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Time Zones. For example, Romance Standard Time is a valid Windows time zone value.

When a web service writes data back to Business Central, dates and times are based on the setting of the Services Default Time Zone setting. However, the web service consumer can override the setting and specify a different time zone.

NOTE

Business Central Server stores dates and times as UTC. When a web service receives data from Business Central Server, the time zone is UTC even if the Services Default Time Zone setting is set to a different time zone.

For example, if the Services Default Time Zone setting is set to **UTC+3**, the following table describes two scenarios where a web service consumer modifies Business Central data and sends this back to Business Central Server.

WEB SERVICE CHANGES THE DATETIME FIELD TO	BUSINESS CENTRAL SERVER INTERPRETS THE DATETIME VALUE AS	BUSINESS CENTRAL SERVER SAVES THE DATETIME VALUE AS
01/01/2014 17:00 UTC+1	01/01/2014 17:00 UTC+1	01/01/2014 16:00 UTC
01/01/2014 17:00	01/01/2014 17:00 UTC+3	01/01/2014 14:00 UTC

See Also

Server Administration Tool

Business Central Windows PowerShell Cmdlets

Preserving Data When Working with a Statically Generated Proxy

3/31/2019 • 2 minutes to read

You can lose data if you develop a web service client that interacts with a statically generated proxy. Similarly, a client web service cannot detect if a field has been added to or removed from a page since a proxy was last generated. This topic describes an approach for avoiding this pitfall.

Avoiding Data Loss by Building the Proxy with Your Client Application

To avoid data loss due to a statically generated proxy, automatically generate your proxy whenever you build your client.

Assume you have published page 21, Customer Card, as a SOAP web service with **Customer** as the **Service Name** value. Add a service reference to this web service from a C# console application and insert the following code in the `Main` method:

```
static void Main(string[] args)
{
    CustomerService.Customer_Service svc
        = new CustomerService.Customer_Service();
    svc.UseDefaultCredentials = true;
    Customer c = svc.Read("01121212");
    Console.WriteLine(c.Name);
    Console.ReadKey();
}
```

When you run the application, you will see the following output (assuming the CRONUS International Ltd. demonstration database):

```
Spotsmeyer's Furnishings
```

Then go back to page 21 and set the **Name** property for the control that is bound to the **Name** field to **CustomerName**, and save the page.

Finally, switch back to the console application without updating the web reference and run the code. Instead of getting an error message that indicates that the web request does not match the web service description, you do not get an error message, and `Console.WriteLine` shows an empty line.

See Also

[Web Services Overview](#)

Web Services Authentication

5/3/2019 • 3 minutes to read

When users send a request for a web service, they are authenticated according to the credential type that is configured for Business Central Server. To access a web service, users must provide valid credentials for the credential type being used. If Business Central is configured for Windows credential type, then users are automatically authenticated against the Windows account that their computer is running under, and they are not prompted for their credentials. For other credential types, users are prompted to enter a user name and password.

About NavUserPassword and AccessControlService credential types

If your solution uses NavUserPassword or AccessControlService as the credential type, users can access data through SOAP and OData web services if they specify a password or a web service access key. You set up the user accounts in the Business Central client, based on how they will access Business Central data. For example, if you set up a user account that will allow an external application to read Business Central data through a web service, you can generate a web service access key and specify that key for the relevant user accounts. Then, you add the access key to the configuration of the application that consumes the web service. In contrast, when users access Business Central data through a web service in Microsoft Excel, for example, they specify a password instead of a web service access key.

Business Central also supports OAuth authentication on OData and SOAP endpoints. OAuth is an open standard for authorization that provides client applications with secure delegated access to server resources. OAuth enables you to extend single sign-on with Office 365 to Business Central web services. For more information, see [Using OAuth to Authorize Business Central Web Services \(OData and SOAP\)](#).

IMPORTANT

If the Business Central Server is configured to use NavUserPassword or AccessControlService authentication, then the username, password, and access key can be exposed if the SOAP or OData data traffic is intercepted and the connection string is decoded. To avoid this condition, configure SOAP and OData web services to use Secure Socket Layer (SSL).

Unicode characters in user name or password

When Business Central data is consumed by a web service, users cannot be authenticated if their user name or password contains Unicode characters. This is a limitation in the basic authentication mechanism that is defined in the HTTP/1.1 specification.


The same limitation applies to exposing Business Central data in external products such as a browser or a Microsoft .NET Framework assembly.

How to use an Access Key for SOAP and OData Web Service Authentication

If your solution is configured for NavUserPassword or AccessControlService authentication, then you can configure Business Central user accounts to include an access key that can be used instead of a password to authenticate SOAP and OData web service requests. A web service access key is a random 44 character string that is associated with the user account. Because it can only be used for SOAP and OData web services, it does not require the same level of protection as a password.

Generate a Web Service Access Key

Follow these instructions to generate a web service access key for a user. You perform these steps from the user setup in Business Central client.

- 1. Choose the  icon, enter **Users**, choose the related link, and then open the user account that you want to edit.
- 2. In the **Web Service Access** section, select the **Web Service Access Key** field.
- 3. In the **Set the Web Service Access Key** window, if you do not want the key to expire, select the **Key Never Expires** check box. If you want the key to expire, set the **Key Expiration Date** field to the date.
- 4. Choose the **OK** button.

The access key is automatically generated and appears in the **Web Service Access Key** field.

Implement the Web Service Access Key

Typically, you would create a user account strictly for web services, and then use the account's credentials, which include the user name and access key, in a web service application. For example, if you develop your own web service application, then you can design your application to programmatically pass the credentials to the web service. Some applications let you provide the connection credentials through a user interface. The steps for implementing the web service access key are done in Business Central client.

- 1. Create a user specifically for web services.

For more information, see [Manage Users and Permissions](#).

- 2. Generate a web service access key in the user account.
- 3. Use the access key in the web service application.

TO	SEE
Learn how to use code to pass the user name and web access key to a web service	Passing Credentials for Authentication to Web Services

See Also

- [Web Services Overview](#)
- [SOAP Web Services](#)
- [OData Web Services](#)
- [Authentication and Credential Types](#)

Using Security Certificates with Business Central On-Premises

3/31/2019 • 8 minutes to read

You use certificates to help secure connections over a wide area network (WAN), such as connections from the Business Central Web Server, Dynamics NAV Client connected to Business Central, and web services to the Business Central Server. Implementing security certificates on your deployment environment requires modifications to various components, like the Business Central Server, Business Central Web Server, and clients.

About Security Certificates

A certificate is a file that Business Central Server uses to prove its identity and establish a trusted connection with the client that is trying to connect. Business Central can support the following configurations:

- *Chain trust*, which specifies that each certificate must belong to a hierarchy of certificates that ends in a root authority at the top of the chain.
- *Peer trust*, which specifies that both self-issued certificates and certificates in a trusted chain are accepted.

The implementation in this section describes the chain trust configuration, which is the more secure option.

NOTE

An instance of Business Central Server that has been configured for secure WAN communication always prompts users for authentication when they start the client, even when the client computer is in the same domain as Business Central Server.

Certificates for Production

In a production environment, you should obtain a certificate from a certification authority or trusted provider. Some large organizations may have their own certification authorities, and other organizations can request a certificate from a third-party organization.

Obtaining Certificates

You implement chain trust by obtaining X.509 service certificates from a trusted provider. These certificates and their root certification authority (CA) certificates must be installed in the certificates store on the computer that is running Dynamics NAV Server. The CA certificate must also be installed in the certificate store on computers that are running the Business Central Web Server and Dynamics NAV Client connected to Business Central so that clients can validate the server.

Most enterprises and hosting providers have their own infrastructure for issuing and managing certificates. You can also use these certificate infrastructures. The only requirement is that the service certificates must be set up for key exchange and therefore must contain both private and public keys. Additionally, the service certificates that are installed on Business Central Server instances must have the Service Authentication and Client Authentication certificate purposes enabled.

IMPORTANT

Microsoft recommends against using wildcard SSL certificates in Business Central installations. Wildcard certificates pose security risks because if one server or sub-domain is compromised, all sub-domains may be compromised. Wildcard certificates also introduce a new style of impersonation attack. In this attack, the victim is lured to a fraudulent resource in the certified domain through phishing. Conventional certificates detect this attack, because the user's browser checks that the private key is hosted on a server whose name matches the one displayed in the browser's address window.

Run the Certificates Snap-in for Microsoft Management Console

Some of the following procedures use the Certificates snap-in for Microsoft Management Console (MMC). If you do not already have this snap-in installed, you can add it to the MMC. For information see [Add the Certificates Snap-in to an MMC](#).

Install and Configure the Certificates

You install the security certificates on the computers running Business Central Server, Business Central Web Server, and Dynamics NAV Client connected to Business Central. The root CA certificate and the service certificate are used in the configuration, but client certificates are not.

Install Certificates on components

1. Follow the installation instructions that are available from your certificate provider to install the root CA and service certificates on the following computers:
 - Install the root CA on the computer that is running Business Central Server and all computers that are running Business Central Web Server instances and Dynamics NAV Client connected to Business Central.
 - Install the service certificate on the computer that is running Business Central Server only.
2. Make sure that the **Server Authentication** and **Client Authentication** certificate purposes are enabled for the service certificate.

A certificate can be enabled for several different purposes. The **Server Authentication** and **Client Authentication** purposes must be enabled. You can enable or disable other purposes to suit your requirements.

You enable certificate purposes by using the Certificates Snap-in for MMC. For more information, see [Modify the Properties of a Certificate](#).

Grant access to the Business Central Server service account

After you have installed the root CA and the service certificate on the computer running Business Central Server, you must grant access to the service account that is associated with the server so that the service account can access the service certificate's private key.

1. In the left pane of MMC, expand the **Certificates (Local Computer)** node, expand the **Personal** node, and then select the **Certificates** subfolder.
2. In the right pane, right-click the certificate, select **All Tasks**, and then choose **Manage Private Keys**.
3. In the **Permissions** dialog box for the certificate, choose **Add**.
4. In the **Select Users, Computers, Service Accounts, or Groups** dialog box, enter the name of the dedicated domain user account that is associated with Business Central Server, and then choose the **OK** button.
5. In the **Full Control** field, select **Allow**, and then choose the **OK** button.

6. In the right pane, select the certificate.
7. In the **Certificate** dialog box, choose the **Details** tab, and then select the **Thumbprint** field.
8. Copy the value of **Thumbprint** field.

For example, copy the hexadecimal characters to text editor, such as Notepad. Delete all spaces from the thumbprint string. If the thumbprint is `c0 d0 f2 70 95 b0 3d 43 17 e2 19 84 10 24 32 8c ef 24 87 79`, then change it to `c0d0f27095b03d4317e219841024328cef248779`.

TIP

It is important that the thumbprint does not contain any invisible extra characters; otherwise you will experience problems when using it later. To avoid this, see [Certificate thumbprint displayed in MMC certificate snap-in has extra invisible unicode character](#).

Configure the Business Central Server instance

The Business Central Server instance configuration includes several settings for certificates and enabling remote logins. You can modify a server instance by using Business Central Server Administration tool or Business Central Administration Shell. For details about how to modify a server instance, see [Configuring Business Central Server](#).

1. Run the Business Central Server Administration tool.
2. Under **General**, change the following settings for the Business Central Server instance.

SETTING	NEW VALUE	DESCRIPTION
Credential Type	<code>NavUserPassword</code> , <code>Username</code> , or <code>AccessControlService</code>	The default value is <code>Windows</code> . When you change it to <code>NavUserPassword</code> , <code>Username</code> , or <code>AccessControlService</code> , client users who connect to the server are prompted for user name and password credentials.
Certificate Thumbprint	Value of the Thumbprint field in the previous procedure.	Remove any leading or trailing spaces in the thumbprint.

3. If you want to use secure web services, then under **SOAP Services** and **OData Services**, select the **Enable SSL** check box.
4. Save and the new values for the server instance.
5. Restart the Business Central Server instance.

If there is a problem, see Windows Event Viewer.

Configure the Business Central Web Server and Dynamics NAV Client connected to Business Central

The chain trust configuration allows client users to log on to one or more instances of Business Central Server as long as their login credentials have been associated with user accounts in Business Central. The client validates that the server certificate is signed with the root CA.

After you have installed the root CA on the computer running the Business Central Web Server or Dynamics

NAV Client connected to Business Central, you must modify the client configuration file.

Modify the Business Central Web client configuration file

1. On the computer that is installed the Business Central Web Server, open the [navsetting.json configuration file](#) in a text editor, such as Notepad.
2. Change the following settings:

KEY	NEW VALUE	DESCRIPTION
ClientServicesCredentialType	NavUserPassword , Username , or AccessControlService	The default value is Windows . When you change it to NavUserPassword , Username , or AccessControlService , client users who connect to the server are prompted for user name and password credentials.
DnsIdentity	The subject name of the service certificate	The default value is <identity> . Replace this with the subject name or common name (CN) of the certificate that is used on the computer that is running Business Central Server.

3. Save the navsettings.json configuration file.

Modify the Dynamics NAV Client connected to Business Central configuration file

1. Open the ClientUserSettings.config configuration file.

The location of this file is *Users\<username>\AppData\RoamingLocal\Microsoft\Dynamics 365 Business Central*.

By default, this file is hidden. Therefore, you may have to change your folder options in Windows Explorer to view hidden files.

NOTE

If you want to change default Dynamics NAV Client connected to Business Central settings for all future users, edit the default ClientUserSettings.config file — that is, the one in C:\Program Files (x86)\Microsoft Dynamics 365 Business Central\140. Be sure that you run your text editor with Administrator privileges when you do so.

2. Modify the following settings.

KEY	NEW VALUE	DESCRIPTION
ClientServicesCredentialType	NavUserPassword , Username , or AccessControlService	The default value is Windows . When you change it to NavUserPassword , Username , or AccessControlService , client users are prompted for user name and password credentials.

KEY	NEW VALUE	DESCRIPTION
DnsIdentity	The subject name of the service certificate.	The default value is <identity>. Replace this with the subject name or common name (CN) of the certificate that is used on the computer that is running Business Central Server.

3. Save and close the ClientUserSettings.config file.

When starting the Dynamics NAV Client connected to Business Central, users are prompted for a valid user name and password.

See Also

[Authentication and User Credential Types](#)

Web Services Best Practices

3/31/2019 • 2 minutes to read

This article provides recommendations that you can implement to make your web services applications easier to understand and maintain.

RECOMMENDATION	EXAMPLE
Use the HTTPS protocol to send data between Business Central and the web service consumer. The examples in this section use the HTTP protocol to illustrate the setup, but we recommend that your solution uses transport-level security.	In the application that consumes the Business Central web service, require that URIs are accessed by using HTTPS. For example, a more secure URI for the OData web services on your local computer is <code>https://localhost:7048/BC130/odata/</code> .
Use singular forms of names. This provides meaningful singular entity names in the generated proxy classes.	When publishing page 21, Customer Card, use Customer as the service name instead of Customers or CustomerCard .
Avoid using spaces and other characters because they are transformed to underscores or other characters that may not be displayed as you want and could lead to ambiguity.	When publishing page 42, Sales Order, remove the space and use SalesOrder as the service name.
Use Pascal casing when you combine words. Pascal casing capitalizes the first character of each word, including acronyms and initialisms that are more than two letters long.	Use SalesOrder or ContactPerson as the service name.

See Also

[Web Services Overview](#)

SOAP Web Services

4/4/2019 • 2 minutes to read

SOAP web services enable full flexibility for building operation-centric services. They provide industry-standard interoperability and channel and host pluggability.

You can use SOAP to interact with page or codeunit web services in Business Central .

TO	SEE
Review the different options for creating URLs to interact with SOAP web services.	SOAP Web Service URLs
Review the set of operations that are available when a page is exposed as a web service.	Basic Page Operations
Learn how to write code that provides a list of existing companies in a Business Central database.	Using SystemService to Find Companies
Ensure that field values are updated from web services.	Using Properties to Indicate Field Value Presence

See Also

[Basic Page Operations](#)

SOAP Web Service URIs

3/31/2019 • 2 minutes to read

Web service users can discover published web services by pointing a browser or a tool such as the Web Services Discovery Tool at the computer running Business Central Server and getting a list of available services. For SOAP web services, you typically enter a URI in a browser to view a list of available Business Central web services or to view a schema for a particular web service.

URIs for SOAP Web Services

To display all published SOAP web services that are exposed by a Business Central Server instance, use a URI of the following type:

```
http://<Server>:<Port>/<ServerInstance>/WS/<CompanyName>/services
```

The following example displays all published SOAP web services that are exposed for the CRONUS International Ltd. demonstration database.

```
http://localhost:7047/BC130/WS/CRONUS%20International%20Ltd/services
```

To view the schema for a particular service, use a URI of the following type:

```
http://<Server>:<Port>/<ServerInstance>/WS/<CompanyName>/Page/<servicename>
```

The following example displays the schema for the Customer service for the CRONUS International Ltd. demonstration database.

```
http://localhost:7047/BC130/WS/CRONUS%20International%20Ltd/Page/Customer
```

You can also use a URI for a codeunit web service, as shown in the following example:

```
http://localhost:7047/BC130/WS/CRONUS%20International%20Ltd/Codeunit/Letters
```

Basic Page Operations

4/4/2019 • 2 minutes to read

When you publish a page as a SOAP web service, it has a set of default operations that are exposed to consumers of the web service.

These operations match the actions a user can perform by interacting with a page using the RoleTailored client. The same page- and table-based business logic is executed.

The following table lists the operations and provides links to reference pages.

Pages that are backed by virtual tables are not editable through web services. When such pages are published as SOAP web services, the only operations that are available are **Read**, **ReadMultiple**, and **IsUpdated**.

The term *entity* that is used in the operation signatures describes the data type that is used. The actual entity is defined by the page that is exposed because it contains all controls that are defined on the page. The controls are normally bound to the page's source table. The entity also contains a key that is a special string that uniquely identifies the source table record with a timestamp. This key is used as an argument in many page operations.

All basic page operations are atomic, which means that either all relevant records are affected or no records are affected, even if there was a faulty condition in only one record.

OPERATION	DESCRIPTION AND SIGNATURE
Create Operation	Creates a single record. <code>void Create(ref Entity entity)</code>
CreateMultiple Operation	Creates a set of records. <code>void CreateMultiple(ref Entity[] entity)</code>
Delete Operation	Deletes a single record. <code>bool Delete(string key)</code>
Delete_part Operation	Deletes a subpage of the current page. <code>bool Delete_<part>(string key)</code>
GetRecIdFromKey	Converts a key, which is always part of the page result, to a record ID. <code>string GetRecIdFromKey(string key)</code>
IsUpdated Operation	Checks if an object has been updated since the key was obtained. <code>bool IsUpdated(string key)</code>
Read Operation	Reads a single record. <code>Entity Read(string no)</code>

OPERATION	DESCRIPTION AND SIGNATURE
ReadByRecId Operation	<p>Reads the record that is identified by RecId. You can use GetRecIdFromKey to obtain a record ID. If the record is not found, then the operation returns null.</p> <pre>Entity ReadByRecId(string formattedRecId)</pre>
ReadMultiple Operation	<p>Reads a filtered set of records, paged.</p> <pre>Entity [] ReadMultiple(Entity_Filter[] filterArray, string bookmarkKey, int setSize)</pre>
Update Operation	<p>Updates a single record.</p> <pre>void Update(ref Entity entity)</pre>
UpdateMultiple Operation	<p>Updates a set of records.</p> <pre>void UpdateMultiple(ref Entity[] entity)</pre>

See Also

[Basic Page Operations](#)

Create Operation

3/31/2019 • 2 minutes to read

Creates a single record. The supplied record object is overwritten with the version that is created by the page.

Method Signature

```
void Create(ref Entity entity)
```

Parameters

PARAMETER	DESCRIPTION
<i>entity</i>	Type: Entity A variable of a specific object type that represents the page.

Results

RESULT NAME	DESCRIPTION
<i>entity</i>	Type: Entity A variable of a specific object type that represents the page. Contains the latest values that are present on the page after the record has been inserted into the table.

Faults

SOAP FAULT MESSAGE	DESCRIPTION
The [<i>record name</i>] already exists. Identification fields and values: [<i>field</i>]=[<i>value</i>]	Indicates that the record insertion would violate key constraints.

Other faults are possible if they are generated by the AL code.

Usage Example

```
Customer cust = new Customer();  
cust.Name = "Customer Name";  
service.Create(ref cust);
```

See Also

[Basic Page Operations](#)

CreateMultiple Operation

3/31/2019 • 2 minutes to read

Creates a set of records. The supplied record object is overwritten with the version that is created by the page.

Method Signature

```
void CreateMultiple(ref Entity[] entity)
```

Parameters

PARAMETER	DESCRIPTION
<i>entity[]</i>	Type: An array of Entities An array of a specific object type that represents the page.

Results

RESULT NAME	DESCRIPTION
<i>entity[]</i>	Type: An array of Entities An array of a specific object type that represents the page. Contains the latest values that are present on the page after the records have been inserted into the table.

Faults

SOAP FAULT MESSAGE	DESCRIPTION
The [record name] already exists. Identification fields and values: [field]=[value]	Indicates that the insertion of at least one of the records would violate key constraints.

Other faults are possible if they are generated by the AL code.

The CreateMultiple operation is executed as a single transaction unless the AL code explicitly commits the transaction. Either all or none of the records are inserted unless the application code does not explicitly call COMMIT.

Usage Example

```
Customer[] custArray = new Customer[3];
for (int i = 0; i < custArray.Length; i++)
{
    custArray[i] = new Customer();
    custArray[i].Name = "Customer Name " + i.ToString();
}
service.CreateMultiple(ref custArray);
```

See Also

[Basic Page Operations](#)

Delete Operation

3/31/2019 • 2 minutes to read

Deletes a single record.

Executing the Delete operation in a web service first executes the [OnDeleteRecord Trigger](#) on the designated page. If application code in the OnDeleteRecord trigger for the page returns **true**, then the [OnDelete Trigger](#) for the corresponding table is executed. If no fault occurs, then the record is deleted from the database.

If the application code in the trigger returns **false**, then the OnDelete trigger is not executed. This does not necessarily mean that the record has not been deleted because it may have been deleted explicitly by the application code for the page's OnDeleteRecord trigger.

The return value from the Delete operation is the return value from the page's OnDeleteRecord trigger.

If Delete returns **true**, then the record has been deleted. If Delete has thrown a fault, then the record has not been deleted. But when Delete returns **false**, then this indicates that there was application logic involved, and the record may or may not have been deleted.

Method Signature

```
bool Delete(string key)
```

Parameters

PARAMETER	DESCRIPTION
<i>key</i>	Type: String The bookmark of the record, including both primary key and concurrency information.

Results

RESULT NAME	DESCRIPTION
<i>Delete_Result</i>	Type: Boolean The value that is returned by the OnDeleteRecord page trigger.

Faults

SOAP FAULT MESSAGE	DESCRIPTION
Other user has modified [<i>record name</i>].	Indicates that another user or process has modified the record after it has been retrieved for this delete operation.
[<i>record name</i>] [<i>field</i>] [<i>value</i>] does not exist.	Indicates that the record has been deleted by another user or process after it has been retrieved for this delete operation.

Other faults are possible if they are generated by the AL code.

Usage Example

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ConsoleApplication
{
    // Imports newly generated web service proxy.
    using WebService;

    class Program
    {
        static void Main(string[] args)
        {
            // Creates instance of service and sets credentials.
            Customer_Service service = new Customer_Service();
            service.UseDefaultCredentials = true;
            Customer cust = new Customer();
            cust.Name = "Customer Name";
            service.Create(ref cust);
            cust = service.Read(cust.No);
            service.Delete(cust.Key);
        }
    }
}
```

See Also

[Basic Page Operations](#)

Delete_<part> Operation

3/31/2019 • 2 minutes to read

Deletes records on a subpage of the current page.

This operation is exposed only by pages that have subpages, which are pages that have parts of type Page. For example, the Sales Order page has a part with the name SalesLines, which has PartType equal to Page and PagePartID equal to "Sales Order Subform." The name of the operation that is exposed by the Sales Order page is Delete_SalesLines.

When you call this operation, you delete a record of the subpage. When you call Read on a page that has a subpage, you get the records of the top-level page and the corresponding records of the subpage. To modify them or add to them, you must modify the whole top-level record.

Executing the Delete_<part> operation in a web service first executes the [OnDeleteRecord Trigger](#) on the designated record of the subpage. If application code in the OnDeleteRecord trigger for the record of the subpage returns **true**, then the [OnDelete Trigger](#) for the corresponding table is executed. If no fault occurs, then the record of the subpage is deleted from the database.

If the application code in the trigger returns **false**, then the OnDelete trigger is not executed. This does not necessarily mean that the record of the subpage has not been deleted because it may have been deleted explicitly by the application code for the page's OnDeleteRecord trigger.

The return value from the Delete_<part> operation is the return value from the page's OnDeleteRecord trigger.

If Delete returns **true**, then the record of the subpage has been deleted. If Delete has thrown a fault, then the record of the subpage has not been deleted. When Delete returns **false**, there was application logic involved, and the record of the subpage may or may not have been deleted.

Method Signature

```
bool Delete_<part>(string key)
```

Parameters

PARAMETER	DESCRIPTION
<i>key</i>	Type: String The bookmark of the record of the subpage, including both primary key and concurrency information.

Results

RESULT NAME	DESCRIPTION
Delete_Result	Type: Boolean The value that is returned by the OnDeleteRecord page trigger.

Faults

SOAP FAULT MESSAGE	DESCRIPTION
Other user has modified <i>[record name]</i> .	Indicates that another user or process has modified the record of the subpage after it has been retrieved for this delete operation.
<i>[record name]</i> <i>[field]</i> <i>[value]</i> does not exist.	Indicates that the record of the subpage has been deleted by another user or process after it has been retrieved for this delete operation.

Other faults are possible if they are generated by the AL code.

Usage Example

This example presents a console application that is created in Visual Studio after registering and publishing the Sales Order page. After you publish the page as a web service, you must also add a web reference to it. This task is also described in the walkthrough.

```
using Delete_SalesLinesSample.DynamicsNAVService;

namespace Delete_SalesLinesSample
{
    class Program
    {
        static void Main(string[] args)
        {
            DynamicsNAVService.SalesOrder_Binding so = new SalesOrder_Binding();
            so.UseDefaultCredentials = true;

            SalesOrder salesOrder = so.Read("2001");
            Sales_Order_Line[] salesLines = salesOrder.SalesLines;

            if (salesLines.Length > 0)
            {
                string key = salesLines[0].Key;
                so.Delete_SalesLines(key);
            }
        }
    }
}
```

See Also

[Basic Page Operations](#)

GetRecIdFromKey

3/31/2019 • 2 minutes to read

Converts a key to a record ID. The key is always part of the page result.

Method Signature

```
string GetRecIdFromKey(string key)
```

Parameters

PARAMETER	DESCRIPTION
<i>key</i>	Type: String The bookmark of the record that includes both primary key and concurrency information.

Results

RESULT NAME	DESCRIPTION
<i>String</i>	Type: String The record ID that was obtained from the key.

Faults

SOAP FAULT MESSAGE	DESCRIPTION
<i>[record name] [field] [value]</i> does not exist.	Indicates that the record has been deleted by another user or process after it has been retrieved for this operation.

Usage Example

```
Customer_Service service = new Customer_Service();
Customer cust = new Customer();
service.UseDefaultCredentials = true;
string id = service.GetRecIdFromKey(cust.No);
cust = service.ReadByRecId(id.ToUpper());
```

See Also

[Basic Page Operations](#)

IsUpdated Operation

3/31/2019 • 2 minutes to read

Checks if an object has been updated since the key was obtained. This operation returns **true** if the object has been updated by any user; otherwise, **false**. Concurrency management prevents a record being changed if it has been subsequently updated. This check proactively prevents that failure.

Method Signature

```
bool IsUpdated(string key)
```

Parameters

PARAMETER	DESCRIPTION
<i>key</i>	Type: String The bookmark of the record, including both primary key and concurrency information.

Results

RESULT NAME	DESCRIPTION
IsUpdated_Result	Type: Boolean Returns true if and only if another user has modified the record.

Faults

SOAP FAULT MESSAGE	DESCRIPTION
<i>[record name]</i> <i>[field]</i> <i>[value]</i> does not exist.	Indicates that the record has been deleted by another user or process after it has been retrieved for this operation.

Usage Example

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ConsoleApplication
{
    // Imports newly generated web service proxy.
    using WebService;

    class Program
    {
        static void Main(string[] args)
        {
            // Creates instance of service and sets credentials.
            Customer_Service service = new Customer_Service();
            service.UseDefaultCredentials = true;

            Customer cust = new Customer();
            cust.Name = "Customer Name";
            service.Create(ref cust);
            cust = service.Read(cust.No);
            if (!service.IsUpdated(cust.Key))
            {
                // Add code here to modify record.
            }
        }
    }
}
```

See Also

[Basic Page Operations](#)

Read Operation

3/31/2019 • 2 minutes to read

Reads a single record.

NOTE

The signature of an operation changes if the page contains one or more unbound fields in the root content area. If there are unbound fields, then the names from the `SourceExpr` property for the unbound fields are added as the first parameters to the operation. They are added according to the tab order of the page. For an example, see the Item Journal.

Method Signature

```
Entity Read(string no)
```

Parameters

PARAMETER	DESCRIPTION
<i>no</i>	<p>Type: String</p> <p>The primary key of the requested record.</p> <p>This argument is a filter on the primary key and returns the first record that matches the filter. When special characters, such as an apostrophe, equal sign, or ampersand, are part of the key, you must enclose the argument with quotation marks.</p> <p>For example, when you write C# code, you should use standard quotation marks for strings and also use additional single quotation marks around a value, such as <code>" 'A&B ' "</code>.</p>

Results

RESULT NAME	DESCRIPTION
<i>entity</i>	<p>Type: Entity</p> <p>A variable of a specific object type that represents the page. Contains the latest values that are present on the page.</p> <p>If the record with the given primary key does not exist, then no value is returned. In the .NET Framework, a null reference is returned.</p>

Faults

This operation does not return a fault when a matching record is not present. Instead, it returns no value. In C#, you should test whether the result is a null reference. Otherwise, you may get a `NullReferenceException` exception.

Faults are possible if they are generated by the AL code.

Usage Example

```
Customer customer = new Customer();  
customer.Name = "Customer Name";  
service.Create(ref customer);  
customer = service.Read(customer.No);
```

See Also

[Basic Page Operations](#)

ReadByRecId Operation

3/31/2019 • 2 minutes to read

Reads the record that is identified by the record ID.

Method Signature

```
Entity ReadByRecId(string formattedRecId)
```

Parameters

PARAMETER	DESCRIPTION
<i>formattedRecId</i>	Type: String The RecId that is used to read the record.

Results

RESULT NAME	DESCRIPTION
<i>entity</i>	Type: Entity A variable of a specific object type that represents the page. Contains the latest values that are present on the page. If the record is not found, then the operation returns null .

Faults

This operation does not return a fault when a matching record is not present. Instead, it returns **null**. In C#, you should test whether the result is a null reference. Otherwise, you may get a `NullReferenceException` exception. Faults are possible if they are generated by the AL code.

Usage Example

```
public void ReadById(String id)
{
    Customer_Service service = new Customer_Service();
    Customer cust = new Customer();
    service.UseDefaultCredentials = true;
    cust = service.ReadByRecId(id.ToUpper());
}
```

See Also

[Basic Page Operations](#)

ReadMultiple Operation

3/31/2019 • 2 minutes to read

Reads a filtered set of records. This operation returns an array of entities. The ReadMultiple operation allows the consumer of a web service to specify the number of records to be returned at one time. This can reduce load on the server.

NOTE

Records on a page that were inserted after the page was retrieved are not read. Records on a page may be incorrectly included in the retrieved dataset if they were deleted after the page was retrieved.

Method Signature

```
Entity [] ReadMultiple(Entity_Filter[] filterArray, string bookmarkKey, int setSize)
```

Parameters

PARAMETER	DESCRIPTION
<i>filterArray</i>	Type: Entity_Filter[] An array of record filters.
<i>bookmarkKey</i>	Type: String The last record bookmark of the page that was previously read. To return the first page of results, set <i>bookmarkKey</i> to NULL.
<i>setSize</i>	Type: Integer The size of the set to be returned. To return the complete set of results, set <i>setSize</i> to zero. To reverse the order of the results, set <i>setSize</i> to negative.

Results

RESULT NAME	DESCRIPTION
-------------	-------------

RESULT NAME	DESCRIPTION
<i>entity[]</i>	<p>Type: An array of Entities</p> <p>An array of a specific object type that represents the page. Contains the latest values that are present on the page.</p> <p>The server will return at most <i>setSize</i> records. If all records have been already returned, then subsequent calls will return no records (a 0-element array in C#). You should keep calling the ReadMultiple method until no records are returned.</p>
<i>entity.Key</i>	<p>Type: String</p> <p>The key of the last record read. In C#, you can access it with <code>Entity[Entity.Length-1].Key</code>. Pass this as <i>bookmarkKey</i> for the next ReadMultiple call.</p>

Faults

This operation does not throw faults when no matching records are present. Instead, it returns an empty record list.

Faults are possible if they are generated by the AL code.

Remarks

Retrieving Last Page

To retrieve the last record of a page, set *setSize* to -1. Using negative numbers in *setSize* sorts the records in descending order.

Entity_Filter

You use the *Entity_Filter* parameter with the ReadMultiple operation. It describes a filter that can be applied on a specific field in a record. The *Entity_Filter* parameter contains two members: Field and Criteria.

- Field contains the name of the field that the filter is applied to. This name comes from the Entity_Fields enum.
- Criteria is of type string and can contain any valid Business Central style filter that is specified in a standard Business Central filter format.

Usage Examples

The following example returns the first 100 customer names that start with an S.

```
List<Customer_Filter> filterArray = new List<Customer_Filter>();
Customer_Filter nameFilter = new Customer_Filter();
nameFilter.Field = Customer_Fields.Name;
nameFilter.Criteria = "S*";
filterArray.Add(nameFilter);
Customer[] custList = service.ReadMultiple(filterArray.ToArray(), null, 100);
```

This example uses the *bookmarkKey* argument to read customer records in batches of 10:

```
Customer_Service service = new Customer_Service();
service.UseDefaultCredentials = true;

const int fetchSize = 10;
string bookmarkKey = null;
List<Customer> customerList = new List<Customer>();

// Reads customer data in pages of 10.
Customer[] results = service.ReadMultiple(new Customer_Filter[] { }, bookmarkKey, fetchSize);
while (results.Length > 0)
{
    bookmarkKey = results.Last().Key;
    customerList.AddRange(results);
    results = service.ReadMultiple(new Customer_Filter[] { }, bookmarkKey, fetchSize);
}

// Prints the collected data.
foreach (Customer customer in customerList)
{
    Console.WriteLine(customer.Name);
}
```

See Also

[Basic Page Operations](#)

Update Operation

3/31/2019 • 2 minutes to read

Updates a single record. The updated record is passed as a reference and is updated with the latest version.

Method Signature

```
void Update(ref Entity entity)
```

Parameters

PARAMETER	DESCRIPTION
<i>entity</i>	Type: Entity A variable of a specific object type that represents the page.

Results

RESULT NAME	DESCRIPTION
<i>entity</i>	Type: Entity A variable of a specific object type that represents the page. Contains the latest values that are present on the page after the record has been updated by the business logic or another concurrent process.

Faults

SOAP FAULT MESSAGE	DESCRIPTION
Other user has modified [<i>record name</i>].	Indicates that another user or process has modified the record after it has been retrieved for this update operation.
[<i>record name</i>] [<i>field</i>] [<i>value</i>] does not exist.	Indicates that the record has been deleted by another user or process after it has been retrieved for this update operation.
The [<i>record name</i>] already exists. Identification fields and values: [<i>field</i>]=[<i>value</i>]	Indicates that the renaming of the record would violate key constraints.

Other faults are possible if they are generated by the AL code.

Usage Example

```
Customer cust = new Customer();  
cust.Name = "Customer Name ";  
service.Create(ref cust);  
cust.Name = cust.Name + "Updated";  
service.Update(ref cust);
```

See Also

[Basic Page Operations](#)

UpdateMultiple Operation

3/31/2019 • 2 minutes to read

Updates a set of records. The updated array of records is passed as a reference and is updated with the latest version.

NOTE

The UpdateMultiple operation attempts to update one field at a time. This can cause errors if the updated value in one field prevents another field's value from being valid. You can resolve this by deleting and inserting lines instead of updating fields.

Method Signature

```
void UpdateMultiple(ref Entity[] entity)
```

Parameters

PARAMETER	DESCRIPTION
<i>entity[]</i>	Type: An array of Entities. An array of a specific object type that represents the page.

Results

RESULT NAME	DESCRIPTION
<i>entity[]</i>	Type: An array of Entities An array of a specific object type that represents the page. Contains the latest values that are present on the page after the records have been updated by the business logic or another concurrent process.

Faults

SOAP FAULT MESSAGE	DESCRIPTION
Other user has modified <i>[record name]</i> .	Indicates that another user or process has modified the record after it has been retrieved for this update operation.
<i>[record name]</i> <i>[field]</i> <i>[value]</i> does not exist.	Indicates that the record has been deleted by another user or process after it has been retrieved for this update operation.
The <i>[record name]</i> already exists. Identification fields and values: <i>[field]=[value]</i>	Indicates that the renaming of the record would violate key constraints.

Other faults are possible if they are generated by the AL code.

The UpdateMultiple operation is executed as a single transaction unless the AL code explicitly commits the

transaction. Either all or none of the records are updated provided that the application code does not explicitly call COMMIT.

Usage Example

The following example adds the string "Updated" to the name of the first 100 customer names that start with S.

```
List<Customer_Filter> filterArray = new List<Customer_Filter>();
Customer_Filter nameFilter = new Customer_Filter();
nameFilter.Field = Customer_Fields.Name;
nameFilter.Criteria = "S*";
filterArray.Add(nameFilter);
Customer[] custList = service.ReadMultiple(filterArray.ToArray(), null, 100);
for (int i = 0; i < custList.Length; i++)
{
    custList [i].Name = custList [i].Name + " Updated";
}
service.Update(ref custList)
```

See Also

[Basic Page Operations](#)

Using SystemService to Find Companies

3/31/2019 • 2 minutes to read

You can use the SystemService service in a SOAP web service application to retrieve a list of companies available in a specific database. A company name is typically part of the URI when you access a Business Central web service, and the system service lets you retrieve names of available companies. If you do not specify a company name in a URI, then the default company is used.

In this procedure, you use the SystemService service to retrieve and print a list of companies in Visual Studio.

Use the SystemService service to find companies

1. In Visual Studio, on the **File** menu, point to **New**, and then choose **Project**.
2. Expand the **Visual C#** node, select **Windows**, and then select **Console Application**.

Caution
Do not double-click or otherwise dismiss the **New Project** dialog box.
3. Enter **FindingCompanies** as the **Name** for the application, and then choose **OK**.
4. In Solution Explorer, right-click the **References** node in the project, and then choose **Add Service Reference**.
5. In the **Add Service Reference** dialog box, choose the **Advanced** button, choose the **Add Web Reference** button, type or paste the URL that you used when checking the WSDL, such as `http://localhost:7047/BC130/WS/Services`, and then choose the green arrow to visit the URL.
6. When the **SystemService** service is displayed, choose **View Service**, wait for the service to be displayed, and then choose **Add Reference**. Rename the Web reference name from **localhost** to **NavSOAPService**.
7. On the **Program.cs** tab, replace the stub code with the following.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace FindingCompanies
{
    using System;
    using BCSOAPService;

    public class Program
    {
        static void Main(string[] args)
        {
            // Creates instance of service and set credentials.
            var systemService = new SystemService
            {
                UseDefaultCredentials = true
            };

            // Loads all companies into an array.
            var companies = systemService.Companies();

            // Runs through and print all companies.
            // Also prints company name in encoded form.
            foreach (string company in companies)
            {
                Console.WriteLine(company);
                Console.WriteLine((Uri.EscapeDataString(company)));
            }
            Console.ReadLine();
        }
    }
}

```

8. Save (press Ctrl+F6) and compile (press F6) the **FindingCompanies** application.
9. Press F5 to run the application in debug mode.

A list of all companies in the current database is presented in a command session.

See Also

[SOAP Web Services](#)

Using Properties with Visual Studio to Indicate the Presence of a Value in a Field

4/4/2019 • 2 minutes to read

When you set a field to a value in the code for a web service client and then send the value back to the web service, such as when you use the [Create Operation](#) or the [Update Operation](#), the field value is not actually updated to anything other than its default value. This issue can occur if you are using proxies that are generated by Visual Studio or the tools in the .NET Framework SDK. This topic describes the issue and shows how to avoid the problem.

Using Properties to Indicate the Presence of Values

The problem occurs because of the way proxies are auto-generated and the way the .NET Framework handles values in XML documents that are exchanged between the web service and the client. The issue is not specific to Business Central.

For example, the following code shows the schema for the two fields on this Customer Card web service:

```
<xsd:element minOccurs="0" maxOccurs="1" name="Credit_Limit_LCY" type="xsd:decimal" />
<xsd:element minOccurs="0" maxOccurs="1" name="Salesperson_Code" type="xsd:string" />
```

Both fields are optional, which means that they do not have to be present in the XML document. One field is declared as a Decimal data type in Business Central. The other field is a string because it is declared as a Text data type in Business Central.

If both values are present, then the XML document should contain the following elements:

```
<Credit_Limit_LCY>1000</Credit_Limit_LCY>
<Salesperson_Code>JR</Salesperson_Code>
```

But if there is no information about the credit limit, then the document should contain the following element:

```
<Salesperson_Code>JR</Salesperson_Code>
```

If the credit limit is zero, then the document should contain the following line:

```
<Credit_Limit_LCY>0</Credit_Limit_LCY>
```

For the decimal type and all .NET Framework value types, you can handle this state in the proxy objects by using a **Boolean *Specified** property:

```
if (salesOrder.Credit_Limit_LCYSpecified)
    there is a value present in salesOrder.Credit_Limit_LCY
else
    there is no useful value in salesOrder.Credit_Limit_LCY
```

If you assign a non-default value in the value type, then you should confirm this by setting the accompanying **Boolean *Specified** property to `true`:

```
salesOrder.Credit_Limit_LCY = 1000;  
salesOrder.Credit_Limit_LCYSpecified = true;
```

To specify that there is no meaningful value in the `salesOrder.Credit_Limit_LCY` value type, set the accompanying **Boolean *Specified** property to `false`:

```
salesOrder.Credit_Limit_LCYSpecified = false;
```

The value in the `salesOrder.Credit_Limit_LCY` value type will now be disregarded.

.NET Framework reference types, such as the String class, are handled differently because .NET Framework declarations that are based on those types can be null, which means that they can have an explicit expression of no value present:

```
if (salesOrder.Salesperson_Code != null)  
    there is a value present in salesOrder.Salesperson_Code  
else  
    there is no value in salesOrder.Salesperson_Code
```

If you assign a value, then you implicitly also define it as present:

```
salesOrder.Salesperson_Code = "JR";
```

To specify that there is no useful value in `salesOrder.Salesperson_Code`, you should set it to null:

```
salesOrder.Salesperson_Code = null;
```

Example

```
SalesOrder salesOrder = new SalesOrder();  
salesOrder.Order_Date = DateTime.Today;  
salesOrder.Order_DateSpecified = true;  
salesOrder.Currency_Code = "SEK";  
  
salesOrderService.Create(ref salesOrder);
```

OData Web Services

3/31/2019 • 2 minutes to read

The Open Data Protocol (OData) is a web protocol that is designed for querying tabular data and provides you with an alternative to SOAP-based web services. OData builds on web technologies such as HTTP, the Atom Publishing Protocol (AtomPub), and JavaScript Object Notation (JSON) to provide access to information from different applications, services, and stores. OData uses URIs for resource identification and commits to an HTTP-based, uniform interface for interacting with resources. This commitment to core Web principles allows for OData to enable a new level of data integration and interoperability across a broad range of clients, servers, services, and tools.

You can use OData web services to show Business Central data, and you can update data in a Business Central database using OData web services.

OData can be found in other Microsoft products and technologies, including the following:

- Microsoft Excel implements OData for its PowerPivot add-in.
- Microsoft SharePoint can expose its list-oriented data with OData.
- Microsoft Azure Table Services are based on OData.

The topics in this section describe the key concepts and techniques for accessing Business Central data from OData applications that are supported by Business Central .

TO	SEE
Use OData to obtain an AtomPub document.	Using OData to Return or Obtain an AtomPub Document
Use OData to obtain a service metadata (EDMX) document.	Using OData to Return or Obtain a Service Metadata (EDMX) Document
Use OData to obtain a JavaScript Object Notation (JSON) document.	Using OData to Return-Obtain a JSON Document
Use filter expressions in OData URIs.	Using Filter Expressions in OData URIs
Use FlowFilters in OData URIs.	Using FlowFilters in OData URIs
Use server-driven paging in OData URIs.	Server-Driven Paging in OData Web Services
Navigate in an OData web service application by using resource properties.	Using Containments and Associations
Write to the database through an OData web service that exposes a writable page.	Using OData Web Services to Modify Data

Enabling and Configuring OData on the Business Central Server

The Business Central Server instance has several configurations settings that enable and control OData services. For more information, see [OData Services Settings](#).

Known Limitations

You can specify filters in OData web services in general that are not supported in Business Central , such as using an OR operator to filter on two different fields. In those cases, you will see the following error:

An error occurred while processing this request.
The 'OR' operator is not supported on distinct fields on an OData filter.

See Also

[SOAP Web Services](#)

Using OData to Return-Obtain an AtomPub Document

3/31/2019 • 3 minutes to read

When you register an OData web service, you expose an OData service that can be accessed from a uniform resource identifier (URI) by using a web browser or any other HTTP client. OData clients can use Atom Publishing Protocol (AtomPub) documents to interact with Business Central data. AtomPub is a simple HTTP-based protocol for creating and updating web resources. It is related to the Atom Syndication Format, which is XML for web feeds. In these procedures, you obtain different kinds of AtomPub documents or feeds from a Business Central OData web service. AtomPub documents and feeds are XML.

NOTE

To use the URIs in this topic, you must have access to the CRONUS International Ltd. demonstration database.

Obtain an AtomPub Document or Feed

Depending on how you construct your URI, you can return an AtomPub document or an AtomPub feed. A feed is a request for data that can change over time. For example, this can be news content or other kinds of information. In the case of Business Central, the information is database content.

1. Register and publish a page web service by using the Business Central Web client. See [Publishing a Web Service](#).

The AtomPub documents that are shown in this article are based on the page 21, the **Customer Card** page, with **Customer** as the service name. The concepts and steps are the same for any Business Central Web client page that you register and publish as a web service.

NOTE

You can also register and publish a Business Central query as a web service.

2. Start Windows Internet Explorer. In the **Address** field, enter a URI in this format:

```
http://<Server>:<WebServicePort>/<ServerInstance>/OData
```

If Business Central Server is running on the local computer with the default Business Central Server instance and OData port, then the address is:

```
http://localhost:7048/<server instance>/OData
```

The browser should now show the web service that you have published in the format of an AtomPub document:

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <service xmlns="http://www.w3.org/2007/app" xmlns:app="http://www.w3.org/2007/app"
  xmlns:atom="http://www.w3.org/2005/Atom" xml:base="http://localhost:7048/DynamicsNAV/OData/">
  - <workspace>
    <atom:title>Default</atom:title>
    - <collection href="Customer">
      <atom:title>Customer</atom:title>
    </collection>
    - <collection href="Company">
      <atom:title>Company</atom:title>
    </collection>
  </workspace>
</service>
```

3. If you have multiple companies, then you can modify your URI to return a feed that enumerates all available companies:

```
http://localhost:7048/<server instance>/OData/Company
```

IMPORTANT

You must modify your Internet Explorer settings to display the actual XML for a feed instead of the feed content that has changed. Choose **Internet Options**, choose **Content**, choose **Feeds and Web Slices**, and then clear the **Turn on feed reading view** check box. Restart Internet Explorer to enable the new setting.

Obtain a Keyed Service Entry

With a keyed service entry, you specify content from a particular row in a Business Central table. The AtomPub document will contain information that is specific to that row. This procedure assumes that you have registered and published a page web service in the previous procedure.

1. Start Windows Internet Explorer. In the **Address** field, enter a URI in the following format to get the entry for the CRONUS International Ltd. company:

```
http://localhost:7048/<server instance>/OData/Company('CRONUS-International-Ltd.')
```

2. To get the data feed for the Customer table in the CRONUS International Ltd. company database, enter a URI in the following format:

```
http://localhost:7048/<server instance>/OData/Company('CRONUS-International-Ltd.)/Customer
```

3. To additionally constrain data to a specific keyed customer in the Customer table, enter a URI in the following format, using the customer no. for the record you want. The example uses customer no. 01121212:

```
http://localhost:7048/<server instance>/OData/Company('CRONUS-International-Ltd.)/Customer('01121212')
```

Obtain a Filtered Data Feed

With a filtered data feed, you use special syntax in the URI to define a query on the available data. For details on the specific filters available for Business Central OData web service applications and the syntax for using them, see [Using Filter Expressions in OData URIs](#).

1. Start Windows Internet Explorer. In the **Address** field, enter a URI in the following format to get the entry for the CRONUS International Ltd. company:

```
http://localhost:7048/<server instance>/OData/Company('CRONUS-International-Ltd.']/Customer?
$filter=City-eq-'Birmingham'
```

See Also

[OData Web Services](#)

Use OData to Return and Obtain a Service Metadata (EDMX) Document

3/31/2019 • 2 minutes to read

The Entity Data Model (**EDM**) is a specification for defining the data that is used by applications that are built on the Entity Framework. **EDMX** is an XML-based file format that is the packaging format for the service metadata of a data service. When you interact with an OData service that is published from Business Central, you can request EDM-based proxies and then use tools such as LINQ to create data access logic. LINQ is a programming model that developers can use to query data from a variety of data sources, including OData. For more information, see [LINQ \(Language-Integrated Query\)](#)

The Business Central implementation of EDM follows the [.NET 4.0 WCF Data Service Framework implementation](#).

The following guidelines have been implemented for EDM.

- Business Central field names are mapped to EDMX property names by replacing spaces with underscores.
- Primary key fields in tables are automatically defined as properties in the service metadata document even if they are not exposed on a page as controls.

Obtain a service metadata (EDMX) document

1. You can obtain service metadata documents for either page or query web services. This example uses a page web service. Register and publish a page web service by using the Business Central Web client. See [Publishing a Web Service](#).
2. Start Windows Internet Explorer. In the **Address** field, enter a URI in this format:

```
http://<Server>:<WebServicePort>/<ServerInstance>/OData/$metadata
```

If Business Central Server is running on the local computer and is using the default Business Central Server instance and OData port, then the address is:

```
http://localhost:7048/<server instance>/OData/$metadata
```

The browser should now show the complete metadata for the page web service that you have published. The beginning of this document looks like this:

```

<?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <edmx:Edmx xmlns:edmx="http://schemas.microsoft.com/ado/2007/06/edmx" Version="1.0">
  - <edmx:DataServices m:DataServiceVersion="1.0"
    xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata">
    - <Schema xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
      xmlns="http://schemas.microsoft.com/ado/2007/05/edm"
      xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices" Namespace="NAV">
      - <EntityType Name="Customer">
        - <Key>
          <PropertyRef Name="No"/>
        </Key>
        <Property Name="No" Nullable="false" Type="Edm.String"/>
        <Property Name="Name" Nullable="true" Type="Edm.String"/>
        <Property Name="Address" Nullable="true" Type="Edm.String"/>
        <Property Name="Address_2" Nullable="true" Type="Edm.String"/>
        <Property Name="Post_Code" Nullable="true" Type="Edm.String"/>
        <Property Name="City" Nullable="true" Type="Edm.String"/>
        <Property Name="Country_Region_Code" Nullable="true" Type="Edm.String"/>
        <Property Name="Phone_No" Nullable="true" Type="Edm.String"/>
        <Property Name="Primary_Contact_No" Nullable="true" Type="Edm.String"/>
        <Property Name="Contact" Nullable="true" Type="Edm.String"/>
        <Property Name="Search_Name" Nullable="true" Type="Edm.String"/>
        <Property Name="Balance_LCY" Nullable="false" Type="Edm.Decimal"/>
        <Property Name="Credit_Limit_LCY" Nullable="false" Type="Edm.Decimal"/>

```

See Also

[OData Web Services](#)

Using OData to Return or Obtain a JSON Document

3/31/2019 • 2 minutes to read

You can publish a page as a web service and consume it using JavaScript Object Notation (JSON).

Obtain a document based on JSON

1. You can build applications that consume and display Business Central data using JSON. This example assumes that you have registered and published a page web service in Business Central.
2. Start Windows Internet Explorer. In the **Address** field, enter a URI in this format:

```
http://<Server>:<WebServicePort>/<ServerInstance>/OData/<web service>?$format=json
```

If Business Central Server is running on the local computer and is using the default Business Central Server instance and OData port, and you have published a web service that is based on page 21 that is called **Customer**, then the address is:

```
http://localhost:7048/BC130/OData/Customer?$format=json
```

This generates a text file that contains metadata and data from the web service. You can open the file from the browser, or you can save it to disk.

NOTE

The value of the format attribute must be lowercase: `?$format=json`.

If you want to consume the web service as JSON-P, you can add the `?$callback=<callback function name>` parameter.

You can use a similar URI to return the web service as an AtomPub document, in which case the attribute is `?$format=atom`. For more information, see [Using OData to Return-Obtain an AtomPub Document](#).

See Also

[Using OData Web Services to Modify Data](#)

Using Filter Expressions in OData URIs

4/4/2019 • 3 minutes to read

You can use filter expressions in OData URIs to limit the results that are returned in an AtomPub document. This topic identifies the filter expressions that you can use, describes the equivalent field or table filter that you can use in AL, and presents examples to show the syntax for using filter expressions in OData web service URIs and applications.

Filter Expressions

To add a filter to an OData URI, add `$filter=` to the end of the name of the published web service. For example, the following URI filters the **City** field in the **Customer** page to return all customers who are located in Miami:

```
http://localhost:7048/BC130/OData/Company('CRONUS International Ltd.']/Customer?$filter=City eq 'Miami'
```

The following table shows the filters that are supported in Business Central OData web services and the equivalent AL filter expressions. All examples are based either on page 21, Customer (published as **Customer**), or on page 20, General Ledger Entry (published as **GLEntry**).

NOTE

Filters that do not have equivalent AL expressions might take longer to process compared to filters that do have equivalent AL expressions. The reason is that filters that do not have equivalent AL expressions are processed on the Business Central Server tier, while filters that do have equivalent AL expressions are processed on the Business Central database tier.

DEFINITION	EXAMPLE AND EXPLANATION	EQUIVALENT AL EXPRESSION
Select a range of values	<pre>filter=Entry_No gt 610 and Entry_No lt 615</pre> <p>Query on GLEntry service. Returns entry numbers 611 through 614.</p>	..
And	<pre>filter=Country_Region_Code eq 'ES' and Payment_Terms_Code eq '14 DAYS'</pre> <p>Query on Customer service. Returns customers in Spain where Payment_Terms_Code= 14 DAYS.</p>	&

DEFINITION	EXAMPLE AND EXPLANATION	EQUIVALENT AL EXPRESSION
Or	<pre>filter= Country_Region_Code eq 'ES' or Country_Region_Code eq 'US'</pre> <p>Query on Customer service. Returns customers in Spain and the United States.</p> <p>Alert: You can use OR operators to apply different filters on the same field. However, you cannot use OR operators to apply filters on two different fields.</p>	
Less than	<pre>filter=Entry_No lt 610</pre> <p>Query on GLEntry service. Returns entry numbers that are less than 610.</p>	<
Greater than	<pre>filter= Entry_No gt 610</pre> <p>Query on GLEntry service. Returns entry numbers 611 and higher.</p>	>
Greater than or equal to	<pre>filter=Entry_No ge 610</pre> <p>Query on GLEntry service. Returns entry numbers 610 and higher.</p>	>=
Less than or equal to	<pre>filter=Entry_No le 610</pre> <p>Query on GLEntry service. Returns entry numbers up to and including 610.</p>	<=
Different from (not equal)	<pre>filter=VAT_Bus_Posting_Group ne 'EXPORT'</pre> <p>Query on Customer service. Returns all customers with VAT_Bus_Posting_Group not equal to EXPORT.</p>	<>
endswith	<pre>filter=endswith(VAT_Bus_Posting_Group, *RT')</pre> <p>Query on Customer service. Returns all customers with VAT_Bus_Posting_Group values that end in RT.</p>	
startswith	<pre>filter=startswith(Name, 'S')</pre> <p>Query on Customer service. Returns all customers names beginning with "S".</p>	
substringof	<pre>filter=substringof(Name, 'urn')</pre> <p>Query on Customer service. Returns customer records for customers with names containing the string "urn".</p>	

DEFINITION	EXAMPLE AND EXPLANATION	EQUIVALENT AL EXPRESSION
indexof	<pre>filter=indexof(Location_Code, 'BLUE') eq 0</pre> <p>Query on Customer service. Returns customer records for customers having a location code beginning with the string BLUE.</p>	
replace	<pre>filter=replace(City, 'Miami', 'Tampa') eq 'CODERED'</pre>	
substring	<pre>filter=substring(Location_Code, 5) eq 'RED'</pre> <p>Query on Customer service. Returns true for customers with the string RED in their location code starting as position 5.</p>	
tolower	<pre>filter=tolower(Location_Code) eq 'code red'</pre>	
toupper	<pre>filter=toupper(FText) eq '2ND ROW'</pre>	
trim	<pre>filter=trim(FCode) eq 'CODE RED'</pre>	
concat	<pre>filter=concat(concat(FText, ', '), FCode) eq '2nd row, CODE RED'</pre>	
round	<pre>filter=round(FDecimal) eq 1</pre>	
floor	<pre>filter=floor(FDecimal) eq 0</pre>	
ceiling	<pre>filter=ceiling(FDecimal) eq 1</pre>	

Referencing Different Data Types in Filter Expressions

You must use the appropriate notation for different data types with filter expressions.

- String values must be delimited by single quotation marks.
- Numeric values require no delimiters.

For more information about data types and other information about conventions and standards for OData URIs, see [Atom Publishing Protocol: URI Conventions](#). Conventions for data types are addressed in section 2.2.2, "Abstract Type System."

See Also

Using FlowFilters in OData URIs

3/31/2019 • 2 minutes to read

You can set FlowFilters on the data that your OData web service extracts from the Business Central database.

FlowFilters are a special kind of filter that you use to set ranges on calculations that are shown in FlowFields. For more information, see [FlowFilter Overview](#). FlowFilters for a page are included in the metadata for that page when it is published as a web service. You can then use FlowFilters as filters in a URI that specifies a query against page data. However, only those FlowFilters that are required to calculate the FlowFields that are exposed on the page as controls are included.

Use FlowFilters to Query Data on the Item Card Page

In this procedure, you create and publish a web service from the **Item Card** page in Business Central and then query the data in that web service by using a FlowFilter.

1. Register and publish a page web service by using the Business Central Web client. See [Publishing a Web Service](#).

Register and publish page 30, Item Card, and name the service **ItemCard**.

2. Start Windows Internet Explorer, and then in the **Address** field, enter a URI in this format:

```
http://<Server>:<WebServicePort>/<ServerInstance>/OData/$metadata
```

If Business Central Server is running on the local computer and uses the default Business Central Server instance and the default OData port, then the address is:

```
http://localhost:7048/BC130/OData/$metadata
```

3. Examine the metadata that is returned by this URI. At the end of the list is a set of parameters that end in the word `Filter`. This is the list of FlowFilters for the page:

```
<Property Type="Edm.String" Name="Location_Filter" Nullable="true"/>
<Property Type="Edm.String" Name="Drop_Shipment_Filter" Nullable="true"/>
<Property Type="Edm.String" Name="Variant_Filter" Nullable="true"/>
<Property Type="Edm.String" Name="Lot_No_Filter" Nullable="true"/>
<Property Type="Edm.String" Name="Serial_No_Filter" Nullable="true"/>
<Property Type="Edm.String" Name="Date_Filter" Nullable="true"/>
```

NOTE

The set of FlowFilters that is listed in the page metadata may not match the set of FlowFilters on the equivalent page in the Business Central Web client. This is because the Business Central Web client shows all FlowFilter fields that are defined on the table on which the page is based. The metadata only shows the FlowFilters that are used to calculate the FlowField controls that are exposed on the page.

4. Create a URI that returns information for a single item card. For example:

```
http://localhost:7048/BC130/OData/Company('CRONUS-International-Ltd.']/ItemCard('1906-S')
```

This is the "ATHENS Mobile Pedestal" item. The value for the *Qty_on_Sales_Order* parameter is 33:

```
<d:Qty_on_Sales_Order m:type="Edm.Decimal">33</d:Qty_on_Sales_Order>
```

5. Apply a FlowFilter to that item and specify **GREEN** as the value for the **Location_Filter**:

```
http://localhost:7048/BC130/OData/Company('CRONUS-International-Ltd.']/ItemCard('1906-S')?
$filter=Location_Filter eq 'GREEN'
```

The item is returned as before, the value of the FlowField that has changed. The value for the *Qty_on_Sales_Order* parameter is now 27:

```
<d:Qty_on_Sales_Order m:type="Edm.Decimal">27</d:Qty_on_Sales_Order>
```

This indicates that there are 27 ATHENS Mobile Pedestals on sales orders designated for the GREEN location.

See Also

[OData Web Services](#)

Server-Driven Paging in OData Web Services

4/4/2019 • 2 minutes to read

Server-driven paging ensures that the quantity of data that is returned by an OData URI does not overwhelm Business Central Server or client program that you use to capture data, while optimizing performance.

NOTE

The term *page* in this topic refers only to a page that contains OData results and is not related to Business Central page objects.

Configuring Server-Driven Paging

You configure server-driven paging with the **Max Page Size** setting in the configuration for the Business Central Server instance that you are using for OData services. To modify the setting, you can use [Server Administration Tool](#) or [Business Central Windows PowerShell Cmdlets]((<https://docs.microsoft.com/en-us/powershell/business-central/overview>)). For more information about **Max Page Size** and other Business Central Server parameters, see [Configuring Business Central Server](#).

The **Max Page Size** setting specifies the maximum number of entities returned per page of OData results. The default value is 1000. You can consider a page to be a chunk of data. A large data feed is divided into chunks of data. Each chunk contains no more entities than the value of **Max Page Size**. An increase in the value of **Max Page Size** creates fewer chunks (or pages) per request, which in turn, decreases the processing time. However, an increase in the **Max Page Size** will increase the memory consumption on the Business Central Server or client. If the value is too large, it can overload the memory on Business Central Server. For performance reasons, you should try to set the value of the **Max Page Size** as large as possible without overloading Business Central Server. If the computer that is running Business Central Server is returning out of memory exceptions, then you should reduce the value of **Max Page Size** until the errors stop.

When using OData with queries that are set with a top number of rows by either the [TopNumberOfRows Property](#) and [TopNumberOfRows Method](#), you should set the **Max Page Size** value greater than the value of the **TopNumberOfRows** property and **TopNumberOfRows** method. For more information, see [Using OData with Queries That are Set with a Top Number of Rows](#).

NOTE

In the CustomSettings.config file for Business Central Server, the **Max Page Size** setting is called **ODataServicesPageMaxSize**.

See Also

[OData Web Services](#)

Using Containments and Associations

3/31/2019 • 3 minutes to read

Containments and associations are relationships between pages in Business Central. OData web services support navigation between pages using containments and associations.

- **Containments:** Some pages in Business Central contain subpages. When you publish such a page, the subpages are automatically available in the web service as containments.
- **Associations:** When a field on a page has a **TableRelation** property, the specified table has a **LookupPageId** property that points to a different page. When you publish a page containing such a field as a web service, you must also publish the page that is pointed to by **LookupPageId** property. You can then link from the first page to the second page in a single URI.

Identifying Containments and Associations in Metadata

OData metadata shows containments and associations with the **NavigationProperty** tag. For example, if you published page 42, Sales Order, and page 22, Customer List, and then obtained the service metadata document for the server instance, you would see this tag in the metadata:

```
<NavigationProperty Name="SalesOrderSalesLines" ToRole="SalesOrderSalesLines" FromRole="SalesOrder"
Relationship="NAV.SalesOrder_SalesOrderSalesLines"/>
```

This tag is followed by metadata that specific to the **SalesOrderSalesLines** subpage. This page metadata indicates that this **NavigationProperty** tag is a containment:

```
<EntityType Name="SalesOrderSalesLines">
  <Key>
    <PropertyRef Name="Document_No"/>
    <PropertyRef Name="Document_Type"/>
    <PropertyRef Name="Line_No"/>
  </Key>
  <Property Name="Document_Type" Nullable="false" Type="Edm.String"/>
  <Property Name="Document_No" Nullable="false" Type="Edm.String"/>
  .....many additional properties.....
  <Property Name="ShortcutDimCode_x005B_7_x005D_" Nullable="true" Type="Edm.String"/>
  <Property Name="ShortcutDimCode_x005B_8_x005D_" Nullable="true" Type="Edm.String"/>
</EntityType>
```

Elsewhere in the metadata, you would see additional **NavigationProperty** lines and a series of **Association** tags. Two of these tags have a **Multiplicity** parameter with a value of **0..1**:

```
<Association Name="SalesOrder_Sell_to_Customer_No_Link">
  <End Type="NAV.SalesOrder" Multiplicity="*" Role="SalesOrder"/>
  <End Type="NAV.CustomerList" Multiplicity="0..1" Role="Sell_to_Customer_No_Link"/>
</Association>
<Association Name="SalesOrder_Bill_to_Customer_No_Link">
  <End Type="NAV.SalesOrder" Multiplicity="*" Role="SalesOrder"/>
  <End Type="NAV.CustomerList" Multiplicity="0..1" Role="Bill_to_Customer_No_Link"/>
</Association>
```

These tags describe two associations from the Sales Order page to the Customer List page:

- The Sell-to Customer No. field.
- The Bill-to Customer No. field.

Using Containments

When you publish a page that has a subpage, you can identify that subpage in the AtomPub document that is returned for the published page. For example, when you publish page 42, Sales Order, you can access a single record on the page using a URI such as the following:

```
http://localhost:7048/<server instance>/OData/Company('CRONUS-International-Ltd. ')/SalesOrder(Document_Type='Order',No='101005')/
```

The following line in the returned AtomPub document for the record provides link information for a containment:

```
<link rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/SalesOrderSalesLines"
      type="application/atom+xml;type=feed" title="SalesOrderSalesLines"
      href="SalesOrder(Document_Type='Order',No='101005')/SalesOrderSalesLines" />
```

Notice the `type=feed` in the line above. To access the subpage data feed, use a URI that incorporates the link that was identified in the previous document:

```
http://localhost:7048/<server instance>/OData/Company('CRONUS-International-Ltd. ')/SalesOrder(Document_Type='Order',No='101005')/SalesOrderSalesLines
```

Using Associations

Associations are possible when two published pages are linked. Here is an example:

- Page 42, Sales Order, has its **SourceTable** property set to table 36, Sales Header. The source expression for the **Sell_to_Customer_No** control on page 42 is field 2, Sell-to Customer No., in table 36.
- Field 2, Sell-to Customer No., in table 36 has a **TableRelation** property set to table 18, Customer, field No.
- Table 18, Customer, has a **LookupPageId** property set to page 22, Customer List.

Thus if both page 42, Sales Order, and page 22, Customer List, are published as web services, then an OData URI can link from the **Sell_to_Customer_No** control on page 42 to the related entity on page 22.

Because of this association, you can create OData URIs to access data on the Customer List page as you work with data on the Sales Order page.

If you publish pages 42 and 22 as web services, then you can return an AtomPub document for the Sales Order page. The following URI returns data for a single record on the page, which is order number 101005:

```
http://localhost:7048/<server instance>/OData/Company('CRONUS-International-Ltd. ')/SalesOrder(Document_Type='Order',No='101005')/
```

A set of three tags near the top of the returned document show one containment (**SalesOrderSalesLines**) and two associations (**Sell_to_Customer_No** and **Bill_to_Customer_No**) on the page. Notice the `type=entry` in the following lines:

```
<link rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/SalesOrderSalesLines"
      type="application/atom+xml;type=feed" title="SalesOrderSalesLines"
      href="SalesOrder(Document_Type='Order',No='101005')/SalesOrderSalesLines" />
<link rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/Sell_to_Customer_No_Link"
      type="application/atom+xml;type=entry" title="Sell_to_Customer_No_Link"
      href="SalesOrder(Document_Type='Order',No='101005')/Sell_to_Customer_No_Link" />
<link rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/Bill_to_Customer_No_Link"
      type="application/atom+xml;type=entry" title="Bill_to_Customer_No_Link"
      href="SalesOrder(Document_Type='Order',No='101005')/Bill_to_Customer_No_Link" />
```

This information provides the necessary information to create a URI to access a record on the Customer List page by using an association:

```
http://localhost:7048/<server instance>/OData/Company('CRONUS-International-Ltd.')
/SalesOrder(Document_Type='Order',No='101005')/Sell_to_Customer_No_Link
```

The following URI returns the same information with direct access to the Customer List page:

```
http://localhost:7048/<server instance>/OData/CustomerList('30000')
```

See Also

[OData Web Services](#)

Using OData with Queries That are Set with a Top Number of Rows

4/4/2019 • 2 minutes to read

Business Central queries include the [TopNumberOfRows Property](#) and [TOPNUMBEROFROWS Method](#) that can be used to specify the maximum number of rows to include in the resulting dataset. The OData configuration includes the **Max Page Size** setting that specifies the maximum number of entities returned per page of OData results. The default value is 1000.

To ensure that the OData results include the correct number of entities when you are using a query that is set with a top number of rows, you should set the **Max Page Size** value greater than the value that is set by the **TopNumberOfRows** property and **TopNumberOfRows** method. Otherwise, the **TopNumberOfRows** property and **TopNumberOfRows** method are ignored and the query dataset will be returned in the OData results.

NOTE

Typically, the **TopNumberOfRows** property or **TopNumberOfRows** method are used to return a relatively small number of entities, such as the top five, ten, or 100 entities. Therefore, in most cases, the value of the **TopNumberOfRows** property and **TopNumberOfRows** method will be less than the **Max Page Size**, so that you will not have to change the **Max Page Size** setting.

For information about how to change the **Max Page Size** setting, see [Configuring Business Central Server and Server-Driven Paging in OData Web Services](#).

See Also

[Basic Page Operations](#)

Using OData Web Services to Modify Data

3/31/2019 • 2 minutes to read

You can write to the Business Central database using an OData web service that exposes a writable page. For example, you can expose a page as an OData web service and implement it in a portal that is based on Microsoft SharePoint Online. Users of the portal can then modify the data.

Modifying Data Using OData Web Services

If an editable page is exposed as a web service, the data in the underlying table can be accessed and modified by an OData call. Business Central supports the following OData operations for modifying data.

ODATA CALL	DATA IMPACT	TRIGGERS RUN ON PAGE AND TABLE IN BUSINESS CENTRAL
POST	Creates a new entity.	OnNewRecord and OnInsert
PUT and MERGE	Modifies the specified existing entity.	OnModify
DELETE	Deletes the specified existing entity.	OnDelete

All calls fail if the user does not have the relevant permissions, and if the relevant property on the page, **InsertAllowed**, **ModifyAllowed**, or **DeleteAllowed**, is set to **No**.

You can use an OData web service in applications where you want users to be able to modify Business Central data the Business Central Web client. For example, you can show fields from the **Customer** table on a mobile device or in a browser so that a user can create, update, or delete customers in the Business Central database.

Company-Specific and Tenant-Specific OData Calls

In your implementation of the web service, you can specify which company in the database that a user can write to in the URIs that expose the web services. Similarly, you can specify the specific tenant that the change applies to if the database handles more than one tenant.

If you do not specify a company, Business Central will identify a default company. The default company is found in the following order of sequence:

1. The ServicesDefaultCompany setting in the Tenants.config file.
2. The ServicesDefaultCompany setting in the CustomSettings.config file for Business Central Server.
3. The company in the current tenant when there is only one company.

If the OData request is for modifying metadata, Business Central will return the first company in the tenant database because metadata applies to all companies in the database.

If no default company can be found based on the criteria, an error message appears.

See Also

[OData Web Services](#)

[Using OData to Return-Obtain a JSON Document](#)

Walkthrough: Creating and Interacting With an OData V4 Bound Action

3/31/2019 • 4 minutes to read

This walkthrough illustrates how you can publish a Business Central function as an OData V4 web service action.

About This Walkthrough

This walkthrough provides an overview of how to expose a function as a web service action and how to verify that the service is working as expected. The walkthrough illustrates the following tasks:

- Publishing a Business Central function as a web service.
- Verifying web service availability from a browser.

Prerequisites

To complete this walkthrough, you will need:

- Microsoft Dynamics NAV 2017 CTP 10 with a developer license.
- CRONUS International Ltd. demonstration database.
- The **Postman** app for testing the web service URI.

Publishing a Function as a Web Service

You publish a function as a Web service action by using the Dynamics NAV Development Environment to create the function and the Business Central Windows client or the Business Central Web client for publishing the objects the function is for. The tutorial provides an example that will return a location header. A location header would be used to later issue a get request for the resulting object. Refer to the **Return a Value** section for an example that will return a value specified in your function.

To create the function

Create a Copy action on the Sales Invoice page.

1. Open the Dynamics NAV Development Environment and then connect to the CRONUS International Ltd. company.

Object Designer opens automatically in the development environment.

2. Open page 43, **Sales Invoice**.
3. Create a new function on the page named `Copy`.
4. Open the properties for the function and set the properties to the following values.

PROPERTY	VALUE
Local	No
FunctionVisibility	External
ServiceEnabled	Yes

5. Open **Locals** for the function and set the parameters to the following values.

PARAMETER	VALUE
VAR	Yes
Name	ActionContext
DataType	DotNet
SubType	Microsoft.Dynamics.Nav.Runtime.WebServiceActionContext.'Microsoft.Dynamics.Nav.Ncl, Culture=neutral, PublicKeyToken=31bf3856ad364e35'

6. Select the **Variables** tab and add the following variables.

NAME	DATATYPE	SUBTYPE
ToSalesHeader	Record	36
FromSalesHeader	Record	36
SalesSetup	Record	311
ODataActionManagement	Codeunit	6711
CopyDocMgt	Codeunit	6620
DocType	Option	OptionString = Quote,Blanket Order,Order,Invoice,Return Order,Credit Memo,Posted Shipment,Posted Invoice,Posted Return Receipt,Posted Credit Memo

7. Add the code that copies the sales document, for example.

```
SalesSetup.GET;
CopyDocMgt.SetProperties(
TRUE,FALSE,FALSE,FALSE,FALSE,SalesSetup."Exact Cost Reversing Mandatory",FALSE);

FromSalesHeader.GET(Rec."Document Type",Rec."No.");
ToSalesHeader."Document Type" := FromSalesHeader."Document Type";
ToSalesHeader.INSERT(TRUE);
CopyDocMgt.CopySalesDoc(DocType::Invoice,FromSalesHeader."No.",ToSalesHeader);

// Add the necessary keys of the newly created entity using that entity's backing table to identify the
// field no. and it's value.
ODataActionManagement.AddKey(Rec.FIELDNO(Id),ToSalesHeader.Id);

// Depending on what the result was of the action
// - Created: SetCreatedPageResponse(ActionContext, PAGE::"Sales Invoice");
// - Updated: SetUpdatedPageResponse(ActionContext,PAGE::"Sales Invoice");
// - Deleted: SetDeleteResponse(ActionContext);
ODataActionManagement.SetCreatedPageResponse(ActionContext,PAGE::"Sales Invoice");
```

8. Save and compile the **SalesInvoice** page.

To register and publish a page as a Web service

1. Open Business Central and connect to the CRONUS International Ltd. company.
2. In the **Search** box, enter **Web services**, and then choose the related link.
3. In the **Web Services** page, on the **Home** tab, choose **New**.
4. In the **Object Type** column, select **Page**. In the **Object ID** column, enter **43**, and in the **Service Name** column, enter **SalesInvoice**.
5. Select the check box in the **Published** column.
6. Choose the **OK** button.

Verifying the Web Service Availability

After publishing a web service, verify that the port that web service applications will use to connect to your web service is open. The default port for OData V4 web services is 7047. You can configure this value by using the [Server Administration Tool](#).

To verify availability of a Microsoft Dynamics NAV Web service action

1. Start **Postman** or another tool that can execute a POST command against the web service URI.
2. In the **Address** field, enter a URI in this format:

```
http://<Server>:
<WebServicePort>/<ServerInstance>/api/beta/companies(<companyid>)/salesInvoices(<invoiceid>)/Microsoft.NAV.Copy)
```

- **<Server>** is the name of the computer that is running Business Central Server.
- **<WebServicePort>** is the port that OData V4 is running on. The default port is 7047.
- **<ServiceInstance>** is the name of the Business Central Server instance for your solution. The default name is DynamicsNAV90.

Example if the default Business Central Server is running on your local computer.

```
http://localhost:7047/BC130/api/beta/companies(b9248a6e-966d-478c-a25d-
d91d28610397)/salesInvoices(8cc52602-3aa4-4256-b2c7-fdfe5248cbf)/Microsoft.NAV.Copy)
```

3. Postman should now show the web service function that you have published, and perform the action of copying an invoice.

Return a value

1. Open the Dynamics NAV Development Environment and connect to the application database.
2. Open page 43, **Sales Invoice**.
3. Create a new function called **Example**.
4. Open **Properties** for the function and set the properties to the following values.

PROPERTY	VALUE
Local	No
FunctionVisibility	External
ServiceEnabled	Yes

5. Open **Locals** for the function parameters to the following values.
- 6.

PARAMETER	VALUE	TEST
inParam	Text	test

7. Select the **Return Value** tab and then add the following values.

8.

NAME	RETURN TYPE
outParam	Text

9. Then add the following code for the **Example** function:

```
outParam := inParam + ' Completed';
```

10. You can now issue a post request:

```
http://localhost:7047/Navision_NAV/ODataV4/Company('CRONUS International Ltd.)/SalesInvoice('Invoice',
'1004')/NAV.Example
```

with a JSON body of:

```
{ "inParam": "Hello World" }
```

11. The returned value will be returned in the body of the message.

```
{
  "@odata.context":
  "http://farpedro.northamerica.corp.microsoft.com:7047/Navision_NAV/ODataV4/$metadata#Edm.String",
  "value": "Hello World Completed"
}
```

You have now published a Business Central function as an OData V4 web service action and verified that the service works as expected. To read more about web services, see the **See Also** section below.

See Also

[Web Services](#)

[SOAP Web Services](#)

[Publish a Web Service](#)

[Securing Web Service Connections Using Certificates](#)

Security and Protection in Business Central

3/31/2019 • 2 minutes to read

An enterprise business solution must have a built-in security system that helps protect your database and the information that it contains from unauthorized access. It must also allow you to specify what authorized users are allowed to do in the database, such as what data they can read and modify. The following sections help you understand and improve the security of Business Central.

[Application Security](#)

[Online Security](#)

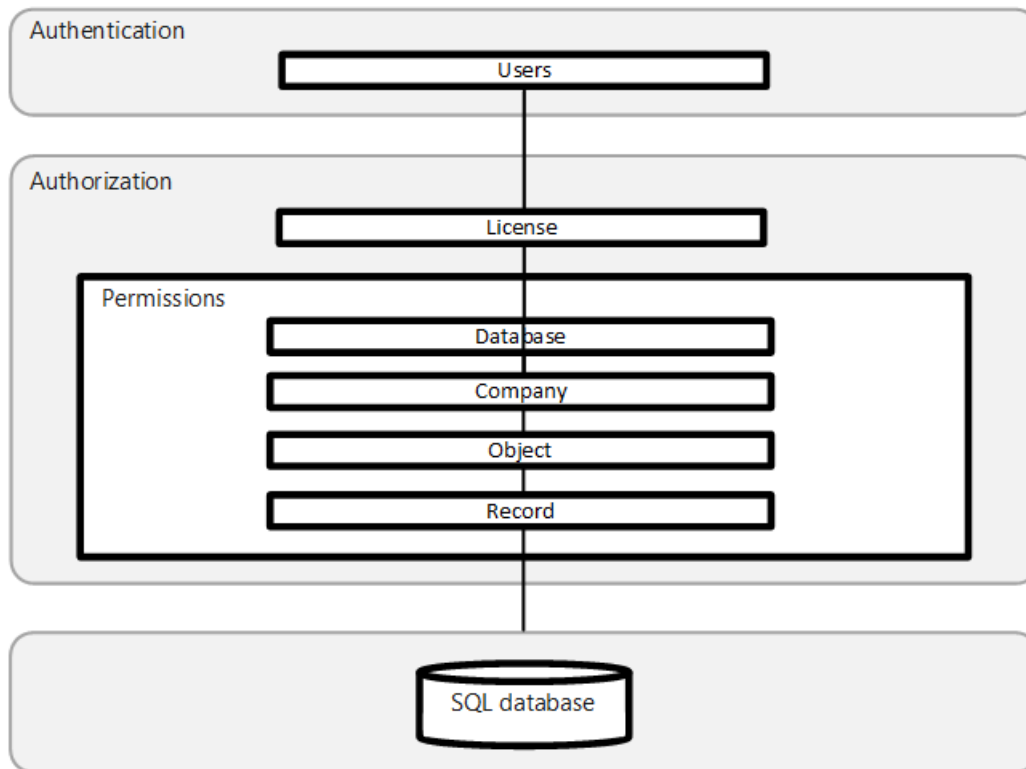
[On-Premises Security](#)

Application Security in Business Central

3/31/2019 • 2 minutes to read

This section helps you understand and improve the security of your Business Central application regardless of where it is hosted. In the articles listed below, you will find guidance and recommended practices related to authentication, authorization, and auditing, as well as data encryption and secure development practices that can be applied to any Business Central environment.

Business Central uses a layered approach to application security, as outlined in the following diagram.



Authentication

Before users can sign-in to the Business Central application, they must be authenticated as valid user in the system. Business Central On-Premises supports several authentication methods, such as Windows and Azure Active Directory. Business Central Online uses strictly Azure Active Directory (Azure AD). For more information, see the following articles:

[Managing Users and Permissions](#)

[Authentication and Credential Types](#)

[Multi-factor Authentication](#)

Authorization

Once authenticated, authorization determines which areas a user can access, such as the pages and reports that they can open, and the permissions that they have on associated data. For more information, see the following articles:

[User Permissions in the Application](#)

Auditing

Business Central includes several auditing features that help you track information about who is signing-in, what their permissions are, what data have they changed, and more. For more information, see the following articles:

[Authorization Assessment](#)

[Data Auditing](#)

[Security Auditing](#)

[Data Classification](#)

Data Encryption

You can encrypt data on the Business Central server by generating new or importing existing encryption keys that you enable on the Business Central server instance that connects to the database. For more information, see [Encrypting Data in Dynamics 365 Business Central](#).

Security Development Lifecycle

Microsoft's Security Development Lifecycle (SDL) is a software development process that helps developers build more secure software and address security compliance requirements while reducing development cost. For more information, see [Security Development Lifecycle](#).

See Also

[Security and Protection](#)

[Online Security](#)

[On-Premises Security](#)

Business Central Online Security

6/26/2019 • 2 minutes to read

This section helps you understand and improve the security of your Dynamics 365 Business Central tenant. In the links below you will find information, guidance and recommended practices related to authentication, data encryption and safely integrating with other services. You will also find information on Business Central's certifications and regulatory compliance.

Authentication

Business Central Online uses Azure Active Directory (Azure AD) as the authentication method, which is automatically set up and managed for you.

Data isolation and encryption

Data belonging to a single tenant is stored in an isolated database and is never mixed with data from other tenants. This ensures complete isolation of data in day-to-day use as well as in backup-restore scenarios. Furthermore, Business Central Online uses encryption to help protect tenant data:

- Data is encrypted at-rest by using Transparent Data Encryption (TDE) and backup encryption.
- Data backups are always encrypted.
- All network traffic inside the service is encrypted by using industry standard encryption protocols.

Service integration

We recommend that you use encrypted network protocols to connect to the PowerBI server and Business Central web services. For more information, see the following articles:

[Connect to Business Central with Power BI](#)

[Using Security Certificates with Business Central On-Premises](#)

See Also

[Microsoft Trust Center \(what we do to make the service secure\)](#)

[Microsoft Dynamics 365 Cloud Services Compliance](#)

[Security and Protection](#)

Business Central On-Premises Security

3/31/2019 • 2 minutes to read

This section helps you understand and improve the security of Business Central hosted on-premises. In the links below you will find information, hardening guidance and recommended best practices addressing client, database, server and network security.

Authentication

Before users can sign-in to the Business Central application, they must be authenticated as valid user in the system. Business Central supports several authentication methods. You configure the authentication method on the server-tiers of Business Central.

For more information, see [Authentication and Credential Types](#).

Server Security

Business Central Server handles communication between clients and databases, controlling authentication, event logging, scheduled tasks, reporting and more. The following articles explain how to improve the security of Business Central Server instances.

[Hardening Business Central Server Security](#)

[Locking Down Server Communication settings](#)

Client Security

The following articles explain how to improve the security of connections from the clients to the Business Central Server.

[Configuring SSL to Secure the Client Connections](#)

[Using Security Certificates with Business Central On-Premises](#)

Database Security

The articles in this section explain how to improve database security in Business Central.

The following articles discuss configurations that you can perform on the Business Central Server:

[Configuring the Database](#)

[Encrypt Traffic](#)

The following are general articles about SQL Server security that can also help secure the database:

[Upgrade to TLS 1.2](#)

[Data Encryption at Rest](#)

[SQL Server Hardening](#)

[SQL Server Auditing](#)

[Backup Encryption](#)

Network Security

The following articles explain to secure client, web service, and PowerBI connections over a wide area network using HTTPS and security certificates.

[Configuring SSL to Secure the Client Connections](#)

[Using Security Certificates with Business Central On-Premises](#)

[Connect to Business Central with Power BI](#)

See Also

[Security and Protection](#)

[Data Security](#)

Upgrading to Dynamics 365 Business Central

3/31/2019 • 2 minutes to read

This section provides an overview of how to upgrade to Business Central. The upgrade process depends on different factors, including on your decision to deploy Business Central on-premises or move your solution online.

TIP

Before you decide how to upgrade your solution, make sure that you read the upgrade considerations at [Important Information and Considerations for Before Upgrading to Dynamics 365 Business Central](#).

Depending on your decision to upgrade to either Business Central online or on-premises, different scenarios are supported. Read the upgrade considerations, and then revise the supported scenarios in the respective sections. If you cannot find guidance for your migration scenario, you can check at the Ideas site if someone else has already suggested support for the same migration path. For more information, see <https://aka.ms/businesscentralideas>.

See Also

[Upgrading to Dynamics 365 Business Central Online](#)
[Upgrading to Dynamics 365 Business Central On-Premises](#)
[Before You Upgrade](#)
[Migrate Legacy Help to the Business Central Format](#)
[Product and Architecture Overview](#)
[Migrating to Multitenancy](#)
[Deployment](#)

Upgrading to Dynamics 365 Business Central Online

3/31/2019 • 2 minutes to read

If you want to move your current solution to Business Central online, the path depends on your current solution. In all cases, it's a matter of migration rather than upgrade due to the nature of online services. The core scenario is to import existing data to an empty company in a Business Central online tenant. For more information, see [Take prospects and customers online](#).

Use the following table to determine the procedures that you must complete for your migration scenario:

SCENARIO	PROCEDURES
Upgrade from Dynamics NAV	Upgrading from Dynamics NAV to Business Central online is only partially supported. For more information, see Upgrading from Dynamics NAV to Business Central online .
Set up a company based on questionnaires	Use RapidStart Services. For more information, see Setting Up a Company With RapidStart Services .
Import data from any system	Use Excel or configuration packages to import data. For more information, see Importing Business Data from Other Finance Systems .
Import data from Dynamics GP	Use the data migration wizard to import master data. For more information, see The Dynamics GP Data Migration Extension .
Import data from Intuit QuickBooks	Use the data migration wizard to import master data. For more information, see The QuickBooks Data Migration Extension .

See Also

[Upgrading to Dynamics 365 Business Central](#)
[Upgrading to Dynamics 365 Business Central On-Premises](#)
[The Lifecycle of Apps and Extensions for Business Central](#)
[Upgrading Extensions](#)
[Product and Architecture Overview](#)
[Deployment](#)

Upgrading to Dynamics 365 Business Central On-Premises

5/6/2019 • 2 minutes to read

The upgrade process depends on different factors, such as the version of Dynamics NAV that you are upgrading from, and the degree to which your solution differs from the standard version of Dynamics NAV. The main tasks range from converting the database to upgrading application code and data.

Use the following table to determine the procedures that you must complete for your upgrade scenario:

SCENARIO	TASKS
Full upgrade from one of the following versions: <ul style="list-style-type: none">• Microsoft Dynamics NAV 2015• Microsoft Dynamics NAV 2016• Microsoft Dynamics NAV 2017• Microsoft Dynamics NAV 2018• Business Central October 2018	From these versions, you can upgrade directly to the latest version of Business Central by following these tasks: <ol style="list-style-type: none">1. Upgrade the Application Code2. Upgrade the Data
Full upgrade from one of the following versions: <ul style="list-style-type: none">• Microsoft Dynamics NAV 2013• Microsoft Dynamics NAV 2013 R2	<ol style="list-style-type: none">1. Upgrade to Microsoft Dynamics NAV 2018. For more information, see Upgrading to Microsoft Dynamics NAV 2018.2. Upgrade to Business Central.<ol style="list-style-type: none">a. Upgrade the Application Codeb. Upgrade the Data: Single-Tenant Deployment or Upgrade the Data: Multitenant Deployment

SCENARIO	TASKS
<p>Full upgrade from one of the following versions:</p> <ul style="list-style-type: none"> • Microsoft Dynamics NAV 2009 SP1 • Microsoft Dynamics NAV 2009 R2 • Microsoft Dynamics NAV 5.0 • Microsoft Dynamics NAV 4.0 	<p>There are two different upgrade paths to Business Central, depending on the version you are upgrading from. For Microsoft Dynamics NAV 5.0 and Microsoft Dynamics NAV 4.0, you must go through Microsoft Dynamics NAV 2013. For Microsoft Dynamics NAV 2009 SP1 and Microsoft Dynamics NAV 2009 R2, you can choose to go through Microsoft Dynamics NAV 2013 or Microsoft Dynamics NAV 2015</p> <p>Through Microsoft Dynamics NAV 2013</p> <ol style="list-style-type: none"> 1. Upgrade to Microsoft Dynamics NAV 2013. For more information, see Upgrading to Microsoft Dynamics NAV 2013 in the MSDN Library. 2. Upgrade to Microsoft Dynamics NAV 2018. For more information, see Upgrading to Microsoft Dynamics NAV 2018. 3. Upgrade to Business Central. <ol style="list-style-type: none"> a. Upgrade the Application Code b. Upgrade the Data: Single-Tenant Deployment or Upgrade the Data: Multitenant Deployment <p>Through Microsoft Dynamics NAV 2015</p> <ol style="list-style-type: none"> 1. Upgrade to Microsoft Dynamics NAV 2015. For more information, see Dynamics NAV Team Blog. 2. Upgrade to Business Central. <ol style="list-style-type: none"> a. Upgrade the Application Code b. Upgrade the Data: Single-Tenant Deployment or Upgrade the Data: Multitenant Deployment <p>After the upgrade, links between interaction records and logged email messages is lost. To resolve this issue, the administrator has to log all mails again to restore the links. For more information, see Logging Interaction Links are Lost When You Upgrade from Microsoft Dynamics NAV 2009 R2.</p>
<p>Platform-only upgrade of Dynamics NAV or Business Central to a new platform version, such as with a cumulative update</p>	<ul style="list-style-type: none"> • Technical Upgrade <p>You can also use this procedure to convert a database to Business Central technical requirements, and then upgrade the application and data later.</p>

Before you begin the upgrade process, see [Important Information and Considerations for Before Upgrading](#) for tips about things to consider when you prepare to upgrade to Business Central.

See Also

[Upgrading to Dynamics 365 Business Central](#)

[Upgrading to Dynamics 365 Business Central Online Product and Architecture Overview](#)

[Migrating to Multitenancy](#)

[Deployment](#)

Transitioning from Codeunit 1 to System Codeunits

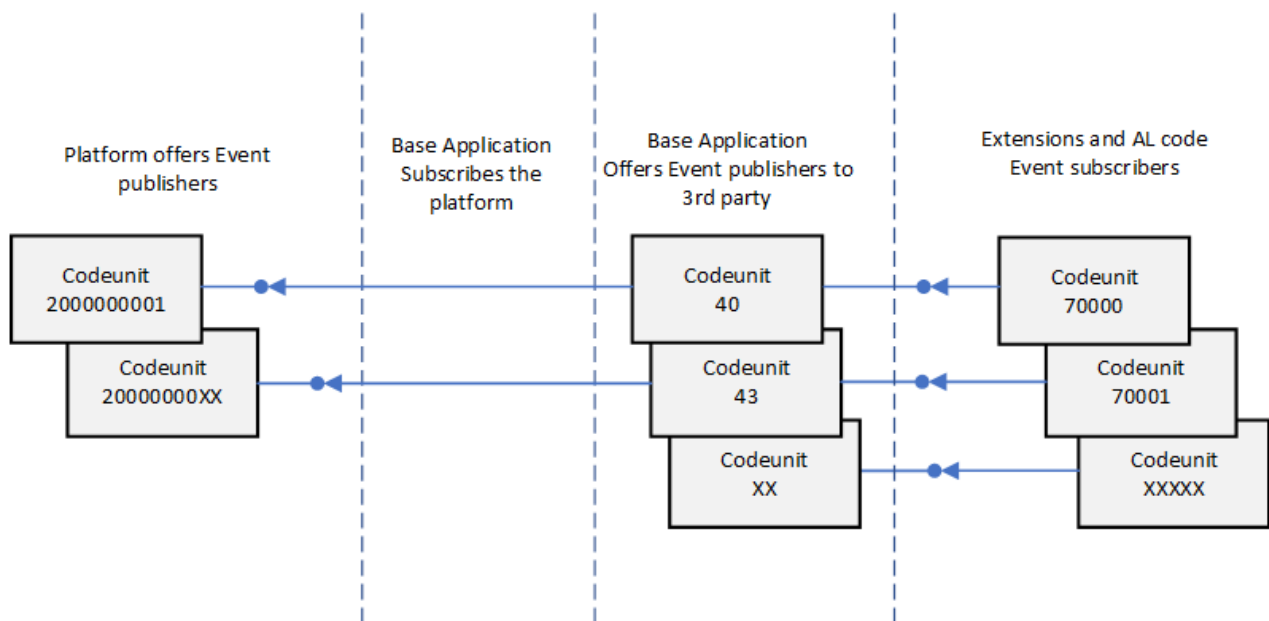
3/31/2019 • 3 minutes to read

With Business Central, codeunit 1 **Application Management** has been removed and replaced with new System codeunits. No functionality has been removed by this change. All system method triggers, event publishers, and their code have been moved to other codeunits.

However, this change will affect the upgrade process from Dynamics NAV and how you develop going forward.

Overview

The foundation of this change is events - publishers and subscribers. System codeunits do not contain code. They only contain event publishers. Instead of running codeunit 1 and calling respective functions, Business Central Server runs system codeunits. The system codeunits will in turn raise published events. There are various management codeunits that subscribe to these events. Like codeunit 1, these subscriber codeunits contain method triggers and integration event publishers, which means that they can call application functionality and raise events. The following figure illustrates the process:



About system codeunits

- They have IDs in the 2 billion range.
- You cannot modify them.
- Currently, we do not recommend that code subscribes to the events in the new system codeunits 2000000001..2000000010 directly. Although this is not blocked, it might be in a future release. Instead, you should subscribe to one of the integration events which now reside next to the business logic.

Mapping Codeunit 1 method triggers to events

The following table lists the mappings between the codeunit 1 triggers and event publishers and the new method trigger and publishers in the management codeunits.

CODEUNIT 1 TRIGGER	NEW CODEUNIT ID	NEW METHOD
CompanyClose	40	CompanyClose

CODEUNIT 1 TRIGGER	NEW CODEUNIT ID	NEW METHOD
--------------------	-----------------	------------

CompanyOpen	40	CompanyOpen
GetSystemIndicator	40	GetSystemIndicator
OnAfterCompanyClose	40	OnAfterCompanyClose
OnAfterCompanyOpen	40	OnAfterCompanyOpen
OnBeforeCompanyClose	40	OnBeforeCompanyClose
OnBeforeCompanyOpen	40	OnBeforeCompanyOpen
FindPrinter	44	GetPrinterName
ApplicationVersion	9015	ApplicationVersion
CustomApplicationVersion	N/A	N/A
ReleaseVersion	9015	ReleaseVersion
ApplicationBuild	9015	ApplicationBuild
CustomApplicationBuild	N/A	N/A
ApplicationLanguage	43	ApplicationLanguage
DefaultRoleCenter	9170	DefaultRoleCenterID
MakeDateTimeText	41	MakeDateTimeText
GetSeparateDateTime	41	GetSeparateDateTime
MakeDateText	41	MakeDateText
MakeTimeText	41	MakeTimeText
MakeText	41	MakeText
MakeDateTimeFilter	41	MakeDateTimeFilter
MakeDateFilter	41	MakeDateFilter
MakeTextFilter	41	MakeTextFilter
MakeCodeFilter	41	MakeTextFilter
MakeTimeFilter	41	MakeTimeFilter

CODEUNIT 1 TRIGGER	NEW CODEUNIT ID	NEW METHOD
AutoFormatTranslate	45	AutoFormatTranslate
ReadRounding	45	ReadRounding
CaptionClassTranslate	42	CaptionClassTranslate
GetCueStyle	9701	GetCueStyle
SetGlobalLanguage	43	SetGlobalLanguage
ValidateApplicationLanguage	43	ValidateApplicationLanguage
LookupApplicationLanguage	43	LookupApplicationLanguage
GetGlobalTableTriggerMask	49	GetGlobalTableTriggerMask
OnGlobalInsert	49	OnGlobalInsert
OnGlobalModify	49	OnGlobalModify
OnGlobalDelete	49	OnGlobalDelete
OnGlobalRename	49	OnGlobalRename
GetDatabaseTableTriggerSetup	49	GetDatabaseTableTriggerSetup
OnDatabaseInsert	49	OnDatabaseInsert
OnDatabaseModify	49	OnDatabaseModify
OnDatabaseDelete	49	OnDatabaseDelete
OnDatabaseRename	49	OnDatabaseRename
OnDebuggerBreak	9500	ProcessOnDebuggerBreak
LaunchDebugger	9500	OpenDebugger
OpenSettings	9170	OpenSettings
OpenContactMSSales	50	OpenContactMSSales
InvokeExtensionInstallation	2501	InvokeExtensionInstallation
CustomizeChart	9180	CustomizeChart
HasCustomLayout	44	HasCustomLayout
MergeDocument	44	MergeDocument

CODEUNIT 1 TRIGGER	NEW CODEUNIT ID	NEW METHOD
ReportGetCustomRdlc	44	ReportGetCustomRdlc
ReportScheduler	44	ScheduleReport
OnBeforeOpenSettings	9170	OnBeforeOpenSettings
OnAfterGetApplicationVersion	9015	OnAfterGetApplicationVersion
OnRoleCenterOpen	9170	OnRoleCenterOpen
OnAfterGetSystemIndicator	79	OnAfterGetSystemIndicator
OnAfterFindPrinter	44	OnAfterGetPrinterName
OnAfterGetDefaultRoleCenter	9170	OnAfterGetDefaultRoleCenter
OnAfterMakeDateText	N/A	N/A
OnAfterMakeTimeText	N/A	N/A
OnAfterMakeText	N/A	N/A
OnAfterMakeDateTimeFilter	41	OnAfterMakeDateTimeFilter
OnAfterMakeDateFilter	41	OnAfterMakeDateFilter
OnAfterMakeTextFilter	41	OnAfterMakeTextFilter
OnAfterMakeCodeFilter	N/A	N/A
OnAfterMakeTimeFilter	41	OnAfterMakeTimeFilter
OnAfterAutoFormatTranslate	45	OnAfterAutoFormatTranslate
OnAfterCaptionClassTranslate	42	OnAfterCaptionClassTranslate
OnAfterGetGlobalTableTriggerMask	49	OnAfterGetGlobalTableTriggerMask
OnAfterOnGlobalInsert	49	OnAfterOnGlobalInsert
OnAfterOnGlobalModify	49	OnAfterOnGlobalModify
OnAfterOnGlobalDelete	49	OnAfterOnGlobalDelete
OnAfterOnGlobalRename	49	OnAfterOnGlobalRename
OnAfterGetDatabaseTableTriggerSetup	49	OnAfterGetDatabaseTableTriggerSetup
OnAfterOnDatabaseInsert	49	OnAfterOnDatabaseInsert

CODEUNIT 1 TRIGGER	NEW CODEUNIT ID	NEW METHOD
OnAfterOnDatabaseModify	49	OnAfterOnDatabaseModify
OnAfterOnDatabaseDelete	49	OnAfterOnDatabaseDelete
OnAfterOnDatabaseRename	49	OnAfterOnDatabaseRename
OnAfterHasCustomLayout	44	OnAfterHasCustomLayout
OnAfterReportGetCustomRdlc	9650	OnAfterReportGetCustomRdlc
OnBeforeOnDatabaseInsert	49	OnBeforeOnDatabaseInsert
OnBeforeOnDatabaseModify	49	OnBeforeOnDatabaseModify
OnBeforeOnDatabaseDelete	49	OnBeforeOnDatabaseDelete
OnBeforeOnDatabaseRename	49	OnBeforeOnDatabaseRename
OnEditInExcel	6710	OnEditInExcel
OnInstallAppPerDatabase	N/A	N/A
OnInstallAppPerCompany	N/A	N/A
OnCheckPreconditionsPerDatabase	9900	OnCheckPreconditionsPerDatabase
OnCheckPreconditionsPerCompany	9900	RaiseOnCheckPreconditionsPerCompany
OnUpgradePerDatabase	9900	OnUpgradePerDatabase
OnUpgradePerCompany	9900	OnUpgradePerCompany
OnValidateUpgradePerDatabase	9900	OnValidateUpgradePerDatabase
OnValidateUpgradePerCompany	9900	OnValidateUpgradePerCompany

What does this mean for upgrade?

For a full upgrade (application code and data), you have to move any custom logic that was included in the old codeunit 1 into the management codeunits and methods described in the previous section. Any custom code that called into codeunit 1 will also have to be changed to call or subscribe to the new methods. You can this as part of the application code upgrade.

For a technical upgrade, after you convert your old database to the Business Central platform, you must import and compile a replacement codeunit 1 object, which you can get from [Codeunit 1 Replacement](#).

See Also

[Converting a Database](#)

Converting a Database to Business Central - Technical Upgrade

5/28/2019 • 13 minutes to read

[See print-friendly quick reference](#)

This article describes how to convert a database from one of the following versions to the latest Business Central platform:

- Microsoft Dynamics NAV 2015
- Microsoft Dynamics NAV 2016
- Microsoft Dynamics NAV 2017
- Microsoft Dynamics NAV 2018

This article can also be used to update your current Business Central database to the latest cumulative update (CU) platform.

About technical upgrade and database conversion

Converting a database, which is often referred to as a *technical upgrade*, changes the database so that it works on the latest Business Central platform. The conversion updates the system tables of the old database to the new schema (data structure), and upgrades of all reports to support Report Viewer 2015. It provides you with the latest platform features and performance enhancements.

The process is slightly different when you have multitenant deployment compared to a single-tenant deployment. The steps that follow indicate the differences where applicable.

IMPORTANT

Before you begin, read the article [Important Information and Considerations for Before Upgrading](#). This article contains information about limitations in a technical upgrade, such as using V1 extensions or Dynamics 365 for Sales integration.

If you are upgrading a single-tenant database to Business Central Cumulative Update 02, 03, 04, or 05, read [Tenant synchronization issue with technical upgrade to Business Central Cumulative Updates 02–05](#) on the Business Central for Partners blog before starting the upgrade.

Tools

To complete the steps in the article, you will use the following tools:

- Dynamics NAV Development Environment (the version that matches your old database and the new version)
- Dynamics NAV Server Administration tool and/or Business Central Server Administration tool
- SQL Server Management Studio

Task 1: (Dynamics NAV upgrade only) Preparation

1. Transition from the use of codeunit 1

With Business Central, codeunit 1 Application Management is no longer used and has been replaced. For more information, see [Transitioning from Codeunit 1](#). To prepare for this change when doing a technical upgrade, do the following:

- a. If you have any custom code in codeunit 1, export the existing codeunit 1 as a .fob or .txt file.
 - b. Go to [Codeunit 1 Replacement](#), and make a .txt file that includes the replacement code for codeunit 1. You will use this file later.
2. Convert V1 extensions to V2 extensions

Business Central does not support V1 extensions. If you are updating a Dynamics NAV database that includes custom V1 extensions, and you want to continue to use them, you have to convert them to V2 extensions. For more information, see [Converting Extensions V1 to Extensions V2](#).

V1 extensions that are produced by Microsoft (first-party extensions, publisher=Microsoft) are now available as V2 extensions on the Business Central installation media (DVD), so you do not have to convert these. If you want to keep the functionality provided and data collected by these V1 extensions, you will have to publish the V2 versions as part of the technical upgrade later in Task 4.

You will have to uninstall all V1 extension to successfully completes the technical upgrade.

Task 2: Preparing the Old Database

Before you convert the old database to Business Central, complete the following steps.

1. Make a copy of the old database or create full database backup.

For more information, see [Create a Full Database Backup \(SQL Server\)](#).

2. For single-tenant mode, uninstall all extensions. For multitenant mode, uninstall all V1 extensions.

You can do this from **Extension Management** page in the Dynamics NAV client or by using the [Uninstall-NAVApp](#) cmdlet of the Dynamics NAV Administration Shell.

To get a list of the extensions that are installed, run this command:

```
Get-NAVAppInfo -ServerInstance <ServerInstanceName> -Tenant <TenantID>
```

For a single-tenant mode, set the `-Tenant` parameter to `default`. V1 extensions are indicated by `ExtensionType: CSIDE`.

For each extension, run this command to uninstall it:

```
Uninstall-NAVApp -ServerInstance <ServerInstanceName> -Name <Name> -Version <N.N.N.N>
```

Alternately, to remove them all at once, you can run this command:

```
Get-NAVAppInfo -ServerInstance <ServerInstanceName> -Tenant default | %{ Uninstall-NAVApp -  
ServerInstance <ServerInstanceName> -Name $_.Name -Version $_.Version }
```

3. Unpublish extensions versions that you do not want to use in the upgraded database. This is optional and typically done for V1 extensions because they are no longer supported.

For example:

```
Unpublish-NAVApp -ServerInstance dynamicsnav110 -Name System -Version 11.0.12925.0
```

4. Open the Dynamics NAV Development Environment that matches the Dynamics NAV or Business Central version of the old database, and then connect to the old application database.

For more information, see [Open Databases](#).

5. In Object Designer, verify that all objects are compiled and no objects are locked.

For more information about compiling objects, see [Compiling Objects](#).

If one or more objects are locked, the conversion process cannot update the database version number. As a result, the conversion does not complete. For more information, see [Locking and Unlocking Objects](#).

6. On the **Tools** menu, choose **Build Server Application Objects**, and then choose the **Yes** button.
7. If any errors occur, they are shown in the **Error List** window. Make sure that you address all compilation errors before you continue.
8. Run the schema synchronization with validation to synchronize the database schema changes.

For more information, see [Synchronizing the Tenant Database and Application Database](#).

9. Upload the Business Central Partner license to the database.

For more information, see [Uploading a License File for a Specific Database](#).

IMPORTANT

The license that you upload must be a developer license. During the conversion, the development environment will convert the report objects that are stored in the old database to the RDL format.

10. (Multitenant only) Dismount tenants.

Use the Dynamics NAV Server Administration tool or [Dismount-NAVTenant](#) cmdlet of the Dynamics NAV Administration Shell to dismount all tenants from the Dynamics NAV Server instance.

```
Dismount-NAVTenant -ServerInstance <serverinstance> -Tenant <tenantID>
```

11. Stop the Dynamics NAV Server instance, and close the development environment.

You can use the Dynamics NAV Server Administration tool or [Set-NAVServerInstance](#) cmdlet of the Dynamics NAV Administration Shell.

To use the Set-NAVServerInstance cmdlet, run the following command:

```
Set-NAVServerInstance -ServerInstance <ServerInstanceName> -Stop
```

12. Clear all records from the **dbo.Server Instance** and **dbo.Debugger Breakpoint** tables in the old application database in SQL Server.

Using SQL Server Management Studio, open and clear the **dbo.Server Instance** and **dbo.Debugger Breakpoint** tables of the old database. For example, you can run the following SQL query:

```
DELETE FROM [<My NAV Database Name>].[dbo].[Server Instance]
DELETE FROM [<My NAV Database Name>].[dbo].[Debugger Breakpoint]
```

13. Close all to connections to the database.

This includes but is not limited to Dynamics NAV client tools and SQL Server Management Studio.

Task 3: Run Technical Upgrade on the Old Database

Next, you will convert the old database so that it can be used with Business Central.

TIP

If you want to write a script that helps you convert databases, you can use the `Invoke-NAVDatabaseConversion` function in the Dynamics NAV Development Shell.

1. If the database is on Azure SQL Database, add your user account to the **dbmanager** database role on the master database.

This membership is only required for converting the database, and can be removed afterwards.

2. Install Business Central.

Run the Business Central Setup, and install the following components as a minimum:

- Server
- SQL Server Database Components
- Administration Tool
- Dynamics NAV Development Environment

IMPORTANT

For a multitenant installation, configure the Business Central Server instance to be a multitenant instance.

3. Run the newly installed Dynamics NAV Development Environment as an administrator.
 - If the Dynamics NAV Development Environment is already connected to the old application database, a dialog box about converting the database appears. Go to the next step.
 - Otherwise, connect to the old application database that you prepared in the previous task, and then go to the next step.

For more information, see [Open Databases](#).

4. In the dialog box that appears, read the instructions about converting the database carefully because this action cannot be reversed. When you are ready, choose the **OK** button, and then choose the **OK** button to confirm that you want to convert the database.

Dynamics NAV Development Environment will now convert the database. This includes an upgrade of system tables and reports.

5. When you are notified that the conversion was successful, choose the **OK** button.
6. If the database references any assemblies (such as client control add-ins) that are not included on the Business Central installation media (DVD), then add the assemblies to the Add-ins folder on Business Central Server.

For Business Central Server, the default path is the `C:\Program Files\Microsoft Dynamics 365 Business Central\140\Service\Add-ins` folder.

7. Connect a Business Central Server instance to the converted database.

Use the Business Central Server Administration tool or the [Set-NAVServerConfiguration cmdlet](#) to connect

a Business Central Server instance to the converted database.

IMPORTANT

The service account that is used by the Business Central Server instance must be a member of the **db_owner** role in the Dynamics NAV database on SQL Server or Azure SQL Database.

For more information, see [Connect a Server Instance to a Database](#) and [Giving the account necessary database privileges in SQL Server](#).

8. Go to the development environment, and set it to use the Business Central Server instance that connects to the database.

For more information, see [Change the Server Instance](#).

9. If upgrading from Dynamics NAV, import the codeunit 1 replacement text file you created earlier.
10. Compile all objects without table schema synchronizing (**Synchronize Schema** set to **Later**); you will do this later.

For more information, see [Compiling Objects](#).

11. Fix compilation errors.

If any errors occur, they are shown in the **Error List** window. For help on resolving the errors, see the following:

- [Resolving Compilation Errors](#)

You can find all objects which did not compile in the **Object Designer** window, by setting a field filter on the **Compiled** field.

12. Recompile V2 extensions that you uninstalled previously.

Use the [Repair-NAVApp cmdlet](#) of the Business Central Administration Shell to compile the published extensions to make sure they are work with the new platform.

For example, you can run the following command to recompile all extensions:

```
Get-NAVAppInfo -ServerInstance <ServerInstanceName> | Repair-NAVApp
```

13. (Multitenant only) Mount the tenant.

Use the [Mount-NAVTenant cmdlet](#).

```
Mount-NAVTenant -ServerInstance <serverinstance> -Tenant <tenantID> -DatabaseName <tenantdatabasename>
```

`-AllowAppDatabaseWrite` is optional but is required for some post-upgrade tasks, like upgrading the control add-ins. When you are done upgrading, you can dismount and mount the tenant without this parameter as needed.

14. Run the schema synchronization with validation to complete the database conversion.

For more information, see [Synchronizing the Tenant Database and Application Database](#).

Task 4: Post-upgrade

1. Upgrade Javascript-based control add-ins to new versions.

The Business Central Server installation includes new versions of Microsoft-provided Javascript-based control add-ins, such as the Business Chart control add-in. If your application is using any of these add-ins, you must upgrade them to the new versions as follows:

- a. Open the Business Central client.
- b. Search for and open the **Control Add-ins** page.
- c. Choose **Actions > Control Add-in Resource > Import**.
- d. Locate and select the .zip file for the control add-in and choose **Open**.

The .zip files are located in the **Add-ins** folder of the Business Central Server installation. There is a sub-folder for each add-in. For example, the path to the Business Chart control add-in is

```
C:\Program Files\Microsoft Dynamics 365 Business Central\140\Service\Add-ins\BusinessChart\Microsoft.Dynamics.Nav.Client.BusinessChart.zip
```

- e. After you have imported all the new control add-in versions, restart Web Server instance.

2. (Single tenant only) Install the V2 extensions that you uninstalled previously.

Use the [Install-NAVApp cmdlet](#) to compile the published extensions to make sure they work with the new platform.

For each V2 extension, run the following command to install it:

```
Install-NAVApp -ServerInstance <ServerInstanceName> -Name <Name> -Version <N.N.N.N>
```

3. (Dynamics NAV 2017 or 2018 upgrade only) If the old database used first-party V1 extensions (publisher Microsoft), then you should publish and install the corresponding V2 extensions that are found on the installation media (DVD). For example, **Sales and Inventory Forecast** and **PayPal Payment Standards** were available as V1 extensions. Now, they are available as V2 extensions.

- a. Publish the system.app and test.app symbol files.

If you installed the **AL Development Environment**, you can find the symbol files where you installed the environment, which by default is C:\Program Files (x86)\Microsoft Dynamics 365 Business Central\140. Otherwise, you can find the files in the **ModernDev** folder on the installation media.

To publish the symbols, open the Business Central Administration Shell as an administrator, and run the following command for each of the symbol files:

```
Publish-NAVApp -ServerInstance <ServerInstanceName> -Path <SymbolFilePath> -PackageType SymbolsOnly
```

- b. Generate the application symbol references by using the finsql.exe file.

Open a command prompt as an administrator, change to the directory where the `finsql.exe` file has been installed as part of Dynamics NAV Development Environment, and then run the following command:

```
finsql.exe Command=generatesymbolreference, ServerName=<DatabaseServerName>\<DatabaseInstance>, Database="<MyDatabaseName>"
```

Replace values for the `Database` and `ServerName` settings to suit.

If the application database contains test objects (ID 130000-139999), then make sure to exclude these objects when generating symbols. You can do this by using the `-Filter` parameter and running the command twice:

```
finsql.exe command=generatesymbolreference, ServerName=<DatabaseServerName>\<DatabaseInstance>, Database="<MyDatabaseName>, filter="Object ID=1..129999"
```

```
finsql.exe command=generatesymbolreference, ServerName=<DatabaseServerName>\<DatabaseInstance>, Database="<MyDatabaseName>, filter="Object ID=140000..1999999999"
```

NOTE

This command does not generate a file. It populates the **Object Metadata** table in the database.

When you run the command, the console returns to an empty command prompt, and does not display or provide any indication about the status of the run. However, the `finsql.exe` may still be running in the background. It can take several minutes for the run to complete, and the symbols will not be generated until such time. You can see whether the `finsql.exe` is still running by using Task Manager and looking on the **Details** tab for **finsql.exe**.

When the process ends, a file named **navcommandresult.txt** is saved to the Dynamics NAV Client connected to Business Central installation folder. If the command succeeded, the file will contain text like `[0] [06/12/17 14:36:17] The command completed successfully in '177' seconds.` If the command failed, another file named **naverrorlog.txt** will be generated. This file contains details about the error(s) that occurred.

For more information about generation symbols, see [Running C/SIDE and AL Side-by-Side](#).

- c. Configure the **Enable loading application symbol references at server startup** (EnableSymbolLoadingAtServerStartup) setting on the Business Central Server instance, and restart the instance.

For more information, see [Configuring Business Central Server](#).

- d. Publish the new V2 extension by running the [Publish-NAVApp](#) cmdlet for each extension:

```
Publish-NAVApp -ServerInstance <ServerInstanceName> -Path <ExtensionFileName>
```

- e. Synchronize the schema with the database by running the [Sync-NAVApp](#) cmdlet for each extension:

```
Sync-NAVApp -ServerInstance <ServerInstanceName> -Name <Name> -Version <N.N.N.N>
```

- f. For each V2 extension, run the following command to install it:

```
Install-NAVApp -ServerInstance <ServerInstanceName> -Name <Name> -Version <N.N.N.N>
```

4. (Dynamics NAV upgrade only) Transition the custom code in the old codeunit 1 to use the new system event implementation.

For more information, see [Transitioning from Codeunit 1](#).

5. (Microsoft Dynamics NAV 2016 upgrade only) Modify C/AL code to ensure that the **My Settings** page

works properly in the Business Central Web client.

For more information, see [Resolving My Settings Page Implementation After a Database Conversion](#).

6. (Dynamics NAV upgrade only) Configure pages and reports included in the MenuSuite to be searchable in the Web client.

The MenuSuite is no longer used to control whether a page or report can be found in the search feature of the Web client. This is now determined by specific properties on the page and report objects. For more information, see [Making Pages and Reports Searchable in Web client After an Upgrade](#).

7. Build the object search index to make objects able to be searched from **Tell Me** in the client. If you completed step 6, you can skip this step.

In the **Tools** menu of the Dynamics NAV Development Environment, select **Build Object Search Index**.

8. Upload the customer license to the converted database.

For more information, see [Uploading a License File for a Specific Database](#).

You have now completed the conversion of the database to be accessed from Business Central. To test the converted database, you can connect it to the Business Central Server instance that is used by Dynamics NAV clients, and then open a client.

Database and Windows collations

Starting from SQL Server 2008, SQL Server collations are fully aligned with the collations in Windows Server. If you upgrade to Business Central from Microsoft Dynamics NAV 2009, the step to convert the database includes upgrading the database from using SQL collations to using Windows collation. This collation change provides users with the most up-to-date and linguistically accurate cultural sorting conventions. For more information, see [Collation and Unicode Support](#).

See Also

[Upgrading the Application Code](#) [Upgrading the Data](#)

[Upgrading to Business Central](#)

Business Central Technical Upgrade Quick Reference

3/31/2019 • 2 minutes to read

This article provides an overview of the technical upgrade process for Business Central. For more detailed steps, see [Technical Upgrade](#).

Prerequisites

STEP	MORE INFO	DONE
Convert custom V1 extensions to V2 extensions.	See...	
Prepare for transitioning from codeunit 1.	See...	

Prepare the old application database

STEP	MORE INFO	DONE
Backup the database(s).	See...	
(Single-tenant mode only) Uninstall all extensions. (Multitenant mode) Uninstall all V1 extensions.	See...	
(Optional) Unpublish unwanted extension versions .	See...	
Ensure all objects are compiled, unlocked, and tables are synchronized.	See...	
Upload a Business Central partner license.	See...	
(Multitenant mode only) Dismount the tenant	See...	
Stop the Dynamics NAV or old Business Central Server Instance		
Clear the dbo.Server Instance and dbo.Debugger Breakpoint tables in SQL Server.	See...	
Close all connections to the database.		

Run the technical upgrade

STEP	MORE INFO	DONE
Install Business Central.	See...	
Open the Dynamics NAV Development Environment as an administrator.		
Connect to and convert the application database.	See...	
Add custom control add-ins to the server instance.	See...	
Connect a Business Central Server instance to the converted application database.	See...	
Connect development environment to the server instance.	See...	
Import codeunit 1 replacement.	See...	
Compile all objects. Important: Choose to synchronize schema later .	See...	
Fix compilation errors.	See...	
Repair published V2 extensions.	See...	
(Multitenant mode only) Mount the tenant database.	See...	
Synchronize the tenant/database.	See...	

Post-upgrade tasks

STEP	MORE INFO	DONE
Upgrade Javascript-based control add-ins to new versions available on Business Central Server.	See...	
(Single-tenant mode only) Install the V2 extensions that were previously uninstalled.	See...	
If the old database used first-party V1 extensions, publish and install the V2 extensions that replace them.	See...	
Transition custom code from old codeunit 1 to management codeunits. (Dynamics NAV 2018 and earlier)	See...	

STEP	MORE INFO	DONE
Configure pages and reports included in the MenuSuite to be searchable in the Web client (Dynamics NAV 2018 and earlier)	See...	
Build object search index.		
Upload the customer license.	See...	

Upgrading the Application Code in Dynamics 365 Business Central

6/4/2019 • 18 minutes to read

Typically, customers want all the customizations that have been implemented in their existing databases to be migrated to their new Business Central databases. Depending on the version of Business Central that a database is being upgraded from, the amount of code changes between the two versions can vary. To upgrade the application code, you must merge code from different versions of the application. This merge process is known as a *code upgrade* or *application upgrade*. You must upgrade the application before you upgrade the data.

IMPORTANT

Before you begin, read the article [Important Information and Considerations for Before Upgrading](#). This article contains information about limitations and things that might require you to perform extra tasks before you upgrade, such as the use of extensions V1 and the deprecation of codeunit 1.

Application Upgrade Overview

During an upgrade, you have to first identify which changes you have to make, and then you'll have to upgrade the application objects and the application code, and finally, you might have to upgrade data so that it fits the new database schema.

For the application portion of the upgrade, you must analyze and process code changes by comparing and merging three separate versions of the database:

VERSION	DESCRIPTION
<i>Original version</i>	This is the baseline version of the solution that you want to upgrade, such as the original release of Business Central October 2018 or Microsoft Dynamics NAV 2018.
<i>Modified version</i>	This is the version that you want to upgrade, such as a customer's Business Central October 2018 or Microsoft Dynamics NAV 2018 database with customizations and add-on solutions.
<i>Target version</i>	This is the target of the merge process that you want to upgrade your application to, such as the standard version of the Business Central database.

When you merge the application objects from these three versions, you can import the result into a new Business Central database that then contains the upgraded application. At the end of the process, you export the merged Business Central objects from this database to a .fob file that you will use during the data upgrade.

Single-tenant and multitenant deployments

The process for upgrading the application code is basically the same for a single-tenant and multitenant deployment. However, there are some inherent differences because with a single-tenant deployment, the application and business data is included in the same database, while with a multitenant deployment application code is in a separate database than the business data (tenants). Here is the general process for each deployment type. In the tasks that follow this section, tasks are marked as *Single-tenant only* or *Multitenant only* where

applicable.

Single-tenant

1. Upgrade the application code.
2. Create a new database on the new platform.
3. Import the upgraded application to the new database.
4. Export the application to a .fob file.
5. Upgrade the data. Here you will import the .fob file.

Multitenant

1. Upgrade the application code.
2. Create a new database on the new platform.
3. Import the upgraded application to the new database.
4. Upgrade the data by mounting the tenant on the application database.

With a multitenant deployment, you will perform the steps in this article on the application database, that is, the database that contains the application object definitions.

Different ways of upgrading application code

You can use any tool or set of tools to help you compare and merge code. Dynamics NAV and Business Central include Windows PowerShell cmdlets and sample scripts that can help you upgrade your application. The cmdlets are available through the Microsoft Dynamics NAV Development Shell and Dynamics NAV Development Shell, or by importing the Microsoft.Dynamics.NAV.Model.Tools.psd1 module into the Windows PowerShell Integrated Scripting Environment (ISE). You can find the sample scripts on the product installation media, in the *WindowsPowerShellScripts\ApplicationMergeUtilities* folder. We recommend that you use these cmdlets and sample scripts because they can make it faster to merge most changes. For example, you can combine several steps in a command that uses a cmdlet such as the [Merge-NAVApplicationObject](#). The sections in this article describe how you can use the Merge-NAVApplicationObject cmdlet and other Windows PowerShell cmdlets. For more information, see [Comparing and Merging Application Object Source Files](#).

Task 1: Install the Prerequisites and Tools

To complete the tasks in this article, you will use various tools and components of the old Dynamics NAV version and Business Central. Ensure that you have the following installed:

Dynamics NAV to Business Central upgrade

	TOOL/COMPONENT
Old Dynamics NAV version	<ul style="list-style-type: none">• Dynamics NAV Development Environment• Dynamics NAV Development Shell
Business Central	<ul style="list-style-type: none">• Business Central Server• Dynamics NAV Development Shell• Business Central Administration Shell• Dynamics NAV Development Environment• Dynamics NAV Development Environment

Business Central to Business Central upgrade

	TOOL/COMPONENT
Old Business Central version	<ul style="list-style-type: none"> • Dynamics NAV Development Environment • Dynamics NAV Development Shell
New Business Central version	<ul style="list-style-type: none"> • Business Central Server • Dynamics NAV Development Shell • Business Central Administration Shell • Dynamics NAV Development Environment

Task 2: Prepare the Application Object Text Files

You must prepare text files that contain the application objects for the different application versions previously described (original, modified, and target). The text files provide the input for the application merge process.

There are three ways to export application objects to text files:

- Use the Dynamics NAV Development Environment version that matches the application database version.

For more information see [To export objects by using the development environment UI](#).

- Use the finsql.exe that matches the application database version to run the ExportObjects command.

For more information, see [To export objects by running finsql.exe with the ExportObjects command](#).

- Use the Microsoft Dynamics NAV Development Shell or Dynamics NAV Development Shell version that matches the application database version.

This is the way that is described in the tasks of this article. Note that the Microsoft Dynamics NAV Development Shell is not available for Microsoft Dynamics NAV 2013 and Microsoft Dynamics NAV 2013 R2. For these versions, you must use development environment or finsql.exe.

Create the application text files

NOTE

Be sure to upload a valid developer license to the database before doing the following steps.

1. Create four folders on the computer, and name them as follows:

- **ORIGINAL**

This folder will be used to store the application object text file(s) from the baseline version, such as the original release of Business Central October 2018, Microsoft Dynamics NAV 2018, or Microsoft Dynamics NAV 2017.

- **MODIFIED**

This folder will be used to store the application object text file(s) from the modified version, such as the customer's database.

- **TARGET**

This folder will be used to store the application object text file(s) from the latest Business Central version.

- **RESULT**

This folder will be used to store the application object text file(s) that are the result of the application merge. It will also contain zero or more .CONFLICT files that describe conflicting code.

2. Export all application objects except system tables from the original version of the old application database, such as the original Microsoft Dynamics NAV 2018 database.

Do not export system tables, which have the IDs in the 2000000000 range. Name the file **OldBaseVersion.txt**, and then save the file in the **ORIGINAL** folder that you created earlier.

For example, start the Microsoft Dynamics NAV Development Shell version that matches the database version, and run the **Export-NAVApplicationObject** function as follows:

```
Export-NAVApplicationObject -DatabaseServer MyServer -DatabaseName "Demo Database NAV (11-0)" -Path  
C:\Upgrade\ORIGINAL\OldBaseVersion.txt -Filter 'Id=1..1999999999'
```

3. Export all application objects, except system tables, from the old modified application database, such as the customer's customized Microsoft Dynamics NAV 2018 database.

Name the file **OldCustomVersion.txt**, and then save the file in the **MODIFIED** folder that you created earlier.

For example (using the Microsoft Dynamics NAV Development Shell version that matches the database version), if the customer's database is called *MyCustomerNAV2016Database*, you can run the following command:

```
Export-NAVApplicationObject -DatabaseServer MyServer -DatabaseName "MyCustomerNAV2018Database" -Path  
C:\Upgrade\MODIFIED\OldCUSTOMVersion.txt -Filter 'Id=1..1999999999'
```

TIP

In some cases, existing customizations might be irrelevant after the upgrade because they correspond to new functionality in Business Central.

4. Export all application objects, except system tables, from the new base version, such as the original Business Central database.

Name the file **NewBaseVersion.txt**, and then save the file in the **TARGET** folder that you created earlier.

For example, using the Dynamics NAV Development Shell for Business Central, run the following command:

```
Export-NAVApplicationObject -DatabaseServer MyServer -DatabaseName "Demo Database BC (14-0)" -Path  
C:\Upgrade\Target\NewBaseVersion.txt -Filter 'Id=1..1999999999'
```

Optionally, you can use the [Split-NAVApplicationObjectFile](#) cmdlet to split each text file into separate text files for each application object. This can make it easier to keep track of the process. The end result at this stage is three folders with one or more text files that contain the three sets of application objects that you want to merge.

Task 3: Merge Versions

You now merge the three sets of application objects to create the application for the new database. This section illustrates how to do this by using the [Merge-NAVApplicationObject](#) cmdlet.

The product installation media contains sample scripts that provide examples of how you can use the [Merge-NAVApplicationObject](#) cmdlet to merge application objects. For more information, see [Merge Application Changes](#).

NOTE

In certain scenarios, you can choose to use the [Compare-NAVApplicationObject](#) cmdlet to identify the changes between the existing customized application and the new application. You can then choose to use the [Update-NAVApplicationObject](#) cmdlet to apply all or some of the changes to the new version. For more information, see [Compare and Update Application Object Source Files](#). However, we recommend that you use the Merge-NAVApplicationObject cmdlet in most cases.

Merge the application object versions into text files

1. Run the new Dynamics NAV Development Shell as an administrator.
2. At the command prompt, change to the directory that contains the four folders that contain the application text files, and then run the following command:

```
Merge-NAVApplicationObject -OriginalPath ORIGINAL -TargetPath TARGET -ModifiedPath MODIFIED -ResultPath RESULT
```

For example:

```
Merge-NAVApplicationObject -OriginalPath C:\Upgrade\ORIGINAL -TargetPath C:\Upgrade\TARGET -ModifiedPath C:\Upgrade\MODIFIED -ResultPath C:\Upgrade\RESULT
```

Depending on the number of objects that you are merging and the number of differences found, this can take a few seconds, a few minutes, or longer. When the cmdlet completes, the result of the merge is shown, including a description of any application objects with conflicting code. The **RESULT** folder will contain a text file (.TXT) for each merged application object and possibly one or more .CONFLICT files that describe the code conflicts that occurred during the merge.

At this point, you can either go to Task 4 to analyze and eventually resolve the conflicts, or you can go directly to Task 5 to import the merged objects as-is from the **RESULT** folder to the new Business Central database.

Task 4: Handling Conflicts

Depending on the application that you are upgrading, you can choose to analyze and fix the conflicting code before you import the merged objects into the Dynamics NAV Development Environment for Business Central. The conflicts are shown in the merged text files but are also identified in .CONFLICT files in the subfolders of the **RESULT** folder. The subfolders **ConflictOriginal**, **ConflictModified**, and **ConflictTarget** folders then contain copies of the source files from the versions that have conflicting code.

You can analyze the conflicts in any tool, make the relevant changes, and then run the merge operation again. For more information, see [Handling Merge Conflicts](#). Alternatively, you can go directly to task 5 to import the merged files into the Dynamics NAV Development Environment, and resolve the conflicts there.

Task 5: Import and Compile Merged Objects in an Empty Database

After you have completed the merge, you import the new merged application objects as text files into a new (empty) Business Central database, and then compile all objects. You must resolve any compilation errors before you can continue. The text files include successfully merged code, and code that is partially merged. You can import the partially merged objects into the Business Central development environment and resolve the conflicts there.

1. Create a new Business Central database for the new upgraded application. The database should be empty, except for the system tables.

For example, give the database the name *My Upgraded App*. For more information, see [Creating and Altering Databases](#).

IMPORTANT

You must set the collation of the new database to match the collation of the old application database. To see the collation of the old database, open it in the old Dynamics NAV Development Environment version, then choose **File > Database > Alter > Collation**.

2. Make sure the database includes a valid Business Central license.

For more information, see [Uploading a License File for a Specific Database](#)

3. Import the new merged application object text files (.TXT) from the **Result** folder into the new database.

There are three ways to import the files:

- Use the [Dynamics NAV Development Environment for Business Central.

For more information see [To import objects by using the development environment UI](#).

- Use the finsql.exe to run the [ImportObjects](#) command.

For more information, see [To import objects by running finsql.exe with the ImportObjects command](#).

- Use the Dynamics NAV Development Shell (or Microsoft.Dynamics.NAV.Model.Tools.psd1 module).

The shell includes the **Join-NAVApplicationObjectFile** cmdlet and **Import-NAVApplicationObject** function. The **Join-NAVApplicationObjectFile** cmdlet combines multiple application object text files into one text file. The **Import-NAVApplicationObject** function runs the [ImportObjects](#) command to import an object file.

This means that you can run a command similar to following to create a single text file from the merge application text files in the **Result** folder:

```
Join-NAVApplicationObjectFile -Source C:\Upgrade\RESULT\*.txt -Destination C:\Upgrade\all-merged.txt
```

Then, you can run this command to import the text file:

```
Import-NAVApplicationObject -DatabaseServer MyServer -DatabaseName "My Upgraded App" -Path C:\Upgrade\all-merged.txt
```

4. Connect the new Business Central Server instance to the database.

You can do this with the Business Central Server Administration tool or the [Set-NAVServerConfiguration cmdlet](#) in the Dynamics NAV Administration Shell. In addition, you must add the service account that is used by the Business Central Server instance as a member of the **db_owner** role in the Business Central database on SQL Server.

For more information about how to do this using the Business Central Server Administration tool, see [How to: Connect a Microsoft Dynamics NAV Server Instance to a Database](#) and [Giving the account necessary database privileges in SQL Server](#).

5. Compile all the newly imported objects. Choose to synchronize **later**.

You can use the Dynamics NAV Development Environment or finsql.exe. For more information, see [Compiling Objects](#).

If you use the Dynamics NAV Development Environment, you will first have to set it to use the Business

Central Server instance that connects to the database. For more information, see [Change the Server Instance Used in C/SIDE](#).

When you compile the objects, an error is thrown for each code conflict, and you can use the tools that are available in the development environment to resolve the conflicts.

Task 6: (Multitenant mode only) Check/change the application family and version

The application and tenant databases are tagged with `Family` and `Version`. To perform the data upgrade, the `Family` on the application must match that tenant's `Family`. The `Version` of the application must be greater than or equal to the tenant's `Version`. The easiest way to ensure that `Family` and `Version` of the upgraded application are compatible for data upgrade is to set `Family` to the same value as the old application, and set the `Version` to a higher value than the old application.

To get the `Family` and `Version`, use the [Get-NAVApplication](#) cmdlet, for example:

```
Get-NAVApplication -ServerInstance BC
```

To set the `Family` and `Version`, use the [Set-NAVApplication](#) cmdlet. For example, to set the family, run the following command:

```
Set-NAVApplication -ServerInstance <ServerInstanceName> -ApplicationFamily <Family>
```

To increase the version by 1, run the following command:

```
Set-NAVApplication -ServerInstance <ServerInstanceName> -IncrementApplicationVersion
```

Or, to specify change to another version, run the following command:

```
Set-NAVApplication -ServerInstance <ServerInstanceName> -ApplicationVersion <N.N.N.N> -Force
```

Task 7: (Dynamics NAV upgrade only) Configure pages and reports to be searchable

The MenuSuite is no longer used to control whether a page or report can be found in the search feature of the Web client. This is now determined by specific properties on the page and report objects. This task is not required at this point, and can be done after the data upgrade as well.

For more information, see [Making Pages and Reports Searchable After an Upgrade](#).

Task 8: Build object search index

Build the object search index to make objects able to be searched from **Tell Me** in the client. If you completed step 6, you can skip this step.

In the **Tools** menu of the Dynamics NAV Development Environment, select **Build Object Search Index**.

For more information, see [Making Pages and Reports Searchable After an Upgrade](#).

Task 9. (Dynamics NAV upgrade only) Transition the custom code from

old codeunit 1 to use the new implementation

Because codeunit 1 has been deprecated in Business Central, you must move any custom logic that was included in the old codeunit 1 into the management codeunits and methods described in the article [Transitioning from Codeunit 1](#).

You now have a new database with a fully upgraded application. For a multitenant deployment, you can start the data upgrade. For this, you will use the new server instance that connects to the upgraded application database. See [Upgrading the Data](#).

Task 10: (Single-tenant mode only) Export all objects

With a single-tenant deployment, export all objects of the new database to a .fob type file, such as **objects.fob** file. You will use this .fob file as part of the data upgrade process. The export must include customized objects, upgraded reget-helpports, and all other Business Central objects.

As with exporting objects in Task 1, you can use either the development environment, finsql.exe, or Dynamics NAV Development Shell.

With the Dynamics NAV Development Shell, you can run a command that is similar to the following:

```
Export-NAVApplicationObject c:\Upgrade\objects.fob -DatabaseName "My Upgraded App" -DatabaseServer  
[server_name]\[database_instance]
```

This completes the upgrade of the application code for single-tenant deployment. Next, you must upgrade the data in the database. See [Upgrading the Data](#).

Task 11: (Multitenant mode only) Import the upgrade toolkit objects

The upgrade toolkit includes upgrade codeunits for handling the data upgrade.

For W1 versions, you can find the default upgrade toolkit objects in the **UpgradeToolKit\Data Conversion Tools** folder on the Business Central installation media (DVD). Choose the FOB that matches the Dynamics NAV version from which you are upgrading:

FROM	TO BUSINESS CENTRAL APRIL 2019	TO BUSINESS CENTRAL OCTOBER 2018
Microsoft Dynamics NAV 2015	Upgrade80014x.FOB	Upgrade800130.FOB
Microsoft Dynamics NAV 2016	Upgrade90014x.FOB	Upgrade900130.FOB
Microsoft Dynamics NAV 2017	Upgrade100014x.FOB	Upgrade1000130.FOB
Microsoft Dynamics NAV 2018	Upgrade110014x.FOB	Upgrade1100130.FOB
Business CentralFall 2018]	Upgrade13x14x.FOB	Not applicable

For local versions, you will find the upgrade toolkit objects in the **UpgradeToolKit\Local Objects** folder. The files follow the same naming convention except they include the 2-letter local version, such as **Upgrade110014x.DK.fob** for Denmark or **Upgrade110014x.DE.fob** for Germany.

For information about importing objects, see [Importing Objects](#).

Task 12: (Multitenant mode only) Publish extensions

1. Import upgrade toolkit.
2. Publish the system and test symbols.

Symbols are a prerequisite for extensions. If you installed the **AL Development Environment**, you can find the symbol files where you installed the environment, which by default is C:\Program Files (x86)\Microsoft Dynamics 365 Business Central\140. Otherwise, you can find the files in the **ModernDev** folder on the installation media.

To publish the symbols, open the Business Central Administration Shell as an administrator, and run the following command for each of the symbol files:

```
Publish-NAVApp -ServerInstance <ServerInstanceName> -Path <SymbolFilePath> -PackageType SymbolsOnly
```

3. Generate the application symbol references by using the finsql.exe file as follows:
 - a. Make sure that **Enable loading application symbol references at server startup** (EnableSymbolLoadingAtServerStartup) is set on the Business Central Server instance.

For more information, see [Configuring Dynamics NAV Server](#).

- b. Open a command prompt as an administrator, change to the directory where the `finsql.exe` file has been installed as part of Dynamics NAV Development Environment, and then run the following command:

```
finsql.exe Command=generatesymbolreference, Database="<MyDatabaseName>", ServerName=  
<DatabaseServerName>\<DatabaseInstance>
```

Replace values for the `Database` and `ServerName` settings to suit.

If the application database contains test objects (ID 130000-139999), then make sure to exclude these objects when generating symbols. You can do this by using the `-Filter` parameter and running the command twice:

```
finsql.exe command=generatesymbolreference, ServerName=<DatabaseServerName>\<DatabaseInstance>,  
Database="<MyDatabaseName>, filter="Object ID=1..129999"
```

```
finsql.exe command=generatesymbolreference, ServerName=<DatabaseServerName>\<DatabaseInstance>,  
Database="<MyDatabaseName>, filter="Object ID=140000..1999999999"
```

NOTE

This command does not generate a file. It populates the **Object Metadata** table in the database.

- c. When you run the command, the console returns to an empty command prompt, and does not display or provide any indication about the status of the run. However, the finsql.exe may still be running in the background. It can take several minutes for the run to complete, and the symbols will not be generated until such time. You can see whether the finsql.exe is still running by using Task Manager and looking on the **Details** tab for **finsql.exe**.

When the process ends, a file named **navcommandresult.txt** is saved to the Dynamics NAV Client connected to Business Central installation folder. If the command succeeded, the file will contain text like `[0] [06/12/17 14:36:17] The command completed successfully in '177' seconds.` If the command

failed, another file named **naverrorlog.txt** will be generated. This file contains details about the error(s) that occurred.

For more information about generation symbols, see [Running C/SIDE and AL Side-by-Side](#).

4. Publish new versions of the Microsoft extensions.

The Business Central installation media (DVD) includes several new versions of Microsoft extensions (that is, extensions that have **Microsoft** as the publisher). If your old deployment uses these extensions, you have to upgrade the old versions to the new versions.

IMPORTANT

For Denmark (DK) and German (DE) versions. Some of the local functionality has been moved from the base application to extensions.

If you are upgrading from a Denmark (DK) version of Dynamics NAV 2017 or earlier, you must publish and install the following extensions to get the local functionality:

NAME	EXTENSION PACKAGE
OIOUBL	OIOUBL.app
Payroll Data Import Definitions (DK)	ImportDKPayroll.app
Payment and Reconciliation Formats (DK)	FIK.app
Tax File Formats (DK)	VATReportsDK.app

If you are upgrading from a German (DE) version of Dynamics NAV or Business Central October 2018 (Cumulative Update 2 or earlier), you must publish and install the following extensions to get the local functionality:

NAME	EXTENSION PACKAGE
ELSTER VAT Localization for Germany	Elster.app

The new versions are found in the `\Extensions` folder of the installation media.

To publish the new extension version, run the [Publish-NAVApp](#) cmdlet:

```
Publish-NAVApp -ServerInstance <ServerInstanceName> -Path <ExtensionFileName>
```

See Also

[Upgrading the Data](#)

[Upgrading to Business Central](#)

Upgrading the Data to Business Central: Single-Tenant Deployment

5/28/2019 • 19 minutes to read

[See print-friendly quick reference](#)

This article describes the tasks required for upgrading the data of a Dynamics NAV or Business Central database to a Business Central major version or cumulative update.

This article pertains to a single-tenant deployment. For upgrade instructions for a multitenant deployment, see [Upgrading the Data: Multitenant Deployment](#).

About Data Upgrade

You use data conversion tools provided with Business Central to convert the old data with the old version's table and field structure, so that it functions together with the new version's table and field structure. Mainly, only table objects and table data are modified during the data upgrade process. Other objects, such as pages, reports, codeunits, and XMLports are upgraded as part of the application code upgrade process.

The data upgrade process described in this article leads you through the database conversion (technical upgrade) and then the upgrade of the actual data, which is achieved by using the upgrade toolkit/upgrade codeunits.

Prerequisites

Before you start the upgrade tasks, make sure you meet the following prerequisites:

1. Your computer uses the same codepage as the data that will be upgraded.

If you use conflicting codepages, some characters will not display in captions, and you might not be able to access the upgraded database. This is because Dynamics NAV must remove incorrect metadata characters to complete the data upgrade. In this case, after upgrade, you must open the database in the development environment on a computer with the relevant codepage and compile all objects. This adds the missing characters again.

Optionally, you can export the captions before the upgrade. For more information, see [How to: Add Translated Strings for Conflicting Text Encoding Formats](#).

2. (Upgrading from Dynamics NAV only) Custom V1 extensions used in the old deployment have been converted to V2 extensions.

For more information, see [Converting Extensions V1 to Extensions V2](#).

3. You have upgraded the application code, and have the FOB files that contain the upgraded application code.

For more information about upgrading the application code, see [Upgrading the Application Code](#).

4. You have the upgrade toolkit FOB files for the version that you are upgrading from.

The upgrade toolkit includes upgrade codeunits for handling the data upgrade. The upgrade toolkit can be in the same FOB file as the application code or in a separate FOB file.

For W1 versions, you can find the default upgrade toolkit objects in the **UpgradeToolKit\Data Conversion Tools** folder on the Business Central installation media (DVD). Choose the FOB that matches the Dynamics NAV version from which you are upgrading:

FROM	TO BUSINESS CENTRAL APRIL 2019	TO BUSINESS CENTRAL OCTOBER 2018
Microsoft Dynamics NAV 2015	Upgrade80014x.FOB	Upgrade800130.FOB
Microsoft Dynamics NAV 2016	Upgrade90014x.FOB	Upgrade900130.FOB
Microsoft Dynamics NAV 2017	Upgrade100014x.FOB	Upgrade1000130.FOB
Microsoft Dynamics NAV 2018	Upgrade110014x.FOB	Upgrade1100130.FOB
Business CentralFall 2018]	Upgrade13x14x.FOB	Not applicable

For local versions, you will find the upgrade toolkit objects in the **UpgradeToolKit\Local Objects** folder. The files follow the same naming convention except they include the 2-letter local version, such as **Upgrade110014x.DK.fob** for Denmark or **Upgrade110014x.DE.fob** for Germany.

5. You have exported the customized permission sets (except SUPER) and permissions from the old database that you want to reuse in the upgraded database.

- When upgrading from Dynamics NAV

To export permission sets and permissions, you run running XMLPort 9171 and 9172.

It is important that you exclude the SUPER permission set when running XMLPort 9171. You can do this by adding the filter `Role ID is <>SUPER`.

For more information, see [Exporting and Importing Permission Sets and Permissions](#).

- When upgrading from an earlier version of Business Central

In the client, search for and open the Permission Sets page, select the user-defined permission sets that you want to keep, and then choose **Export Permission Sets**.

6. If the old deployment uses data encryption, you have exported the encryption key file that it used for the data encryption.

For more information, see [How to: Export and Import Encryption Keys](#).

7. (Optional) Make a copy of the configuration file (web.config or navsettings.json) for all Business Central Web Server instances in the old deployment.

8. Business Central has been installed.

As a minimum, you must install the following components:

- Server
- SQL Server Components
- Business Central Web Server components
- Dynamics NAV Development Environment
- AL Development Environment
- (optionally) Business Central Server Administration tool

For more information, see [Installing Business Central Using Setup](#).

NOTE

If the old Dynamics NAV application uses Payment Services for Microsoft Dynamics ERP, be aware that this was discontinued in Microsoft Dynamics NAV 2017. This means that most of the objects that are associated with this feature will be deleted during the upgrade. Some objects you will have to manually delete.

Task 1: Create full SQL backup of old database

Create a full backup of the old database in the SQL Server. Alternatively, you can make a copy of the old database and perform the upgrade tasks on the copy.

For more information, see [Create a Full Database Backup \(SQL Server\)](#).

Task 2 Uninstall all extensions in old database

Open the Dynamics NAV Administration Shell or Business Central Administration Shell that matches to old database, and run these commands:

1. To get a list of the extensions that are installed, run this command:

```
Get-NAVAppInfo -ServerInstance <ServerInstanceName> -Tenant default
```

Replace `<ServerInstanceName>` with the name of the Dynamics NAV Server instance that the database connects to.

2. For each extension, run this command to uninstall it:

```
Uninstall-NAVApp -ServerInstance <ServerInstanceName> -Name <Name> -Version <N.N.N.N>
```

Replace `<ServerInstanceName>` with the name of the Dynamics NAV Server instance that the database connects to.

Replace `<Name>` and `<N.N.N.N>` with the name and version of the Extension V1 as it appeared in the previous step.

Alternately, to remove them all at once, you can run this command:

```
Get-NAVAppInfo -ServerInstance <ServerInstanceName> -Tenant default | % { Uninstall-NAVApp -  
ServerInstance <ServerInstanceName> -Name $_.Name -Version $_.Version }
```

Task 3: Upload Business Central partner license to old database

By using the Dynamics NAV Development Environment that matches the old database, upload the Business Central license to the database.

For more information, see [\[Uploading a License File for a Specific Database\]](#).

Task 4: Delete all objects except tables in old database

In the development environment version that matches the database, open the old database, open Object Designer, select all objects except tables, and then choose **Delete**.

You can also use the [DeleteObjects](#) command of the finsql.exe.

Task 5: Clear server instance and debugger breakpoint records in old database

Clear all records from the **dbo.Server Instance** and **dbo.Debugger Breakpoint** tables in the database in SQL Server.

1. Make sure that you stop the old server instance, and close any tools that connect to the database, such as the Dynamics NAV Administration Tool and development environment.
2. Using SQL Server Management Studio, open and clear the **dbo.Server Instance** and **dbo.Debugger Breakpoint** tables of the old database. For example, you can run the following SQL query:

```
DELETE FROM [<My NAV Database Name>].[dbo].[Server Instance]
DELETE from [<My NAV Database Name>].[dbo].[Debugger Breakpoint]
```

Task 6: Convert old database to Business Central

If the database is on Azure SQL Database, you must first add your user account to the **dbmanager** database role on master database. This membership is only required for converting the database, and can be removed afterwards.

To convert the old database to the Business Central format, open the old database in the new Dynamics NAV Development Environment for Business Central, and follow the conversion instructions.

For more information about how to open a database, see [Open a Database](#).

IMPORTANT

Do not run schema synchronization at this time. Choose to run it **later**.

Task 7: Import upgraded application objects to converted database

Using Dynamics NAV Development Environment for Business Central, import the application objects that you want in the database. This includes the application objects FOB file (from the application code upgrade) and the upgrade toolkit objects FOB file.

1. Import the application objects FOB file first, and then import the upgrade toolkit FOB file.

For more information, see [Importing Objects](#).

2. **IMPORTANT** When prompted about table synchronization, set the **Synchronize Schema** option to **Later**.
3. When you import the FOB file, if you experience metadata conflicts, the **Import Worksheet** windows appears.

Review the Worksheet page. For more information, see [Import Worksheet](#).

Choose **Replace All**, and then **OK** to continue.

Task 8: Connect a Business Central Server instance to converted database

You use the Business Central Server Administration tool or [Set-NAVServerConfiguration cmdlet](#) in the Business Central Administration Shell to connect a Business Central Server instance to the converted database.

The service account that is used by the Business Central Server instance must be a member of the **db_owner** role

in the Business Central database on SQL Server or Azure SQL Database.

For more information, see [Connecting a Server Instance to a Database](#) and [Giving the account necessary database privileges in SQL Server](#).

IMPORTANT

When upgrading a large database, you should increase the **SQL Command Timeout** setting for the Business Central Server instance, to avoid timeouts during schema synchronization. The default setting is 30 minutes.

Task 9: Compile all objects in converted database

1. In the Dynamics NAV Development Environment, set it to use the server instance that connects to the database.

For more information, see [Changing the Server Instance](#).

2. Use the Dynamics NAV Development Environment or finsql.exe to compile all objects. This includes the imported application objects, data tables, and system tables.

IMPORTANT

Choose to run schema synchronization later. For example, in Object Designer, choose **Tools**, choose **Compile**, set the **Synchronize Schema** option to **Later**, and then choose **OK**. For more information, see [Compiling Objects](#).

3. (Upgrade from Microsoft Dynamics NAV 2016 and earlier only) If you get errors on the following table objects, use the Object Designer to delete the objects because they are no longer used.

- Table 470 Job Queue (replaced by the [Task Scheduler](#))
- Table 824 DO Payment Connection Details
- Table 825 DO Payment Connection Setup
- Table 827 DO Payment Credit Card
- Table 828 DO Payment Credit Card Number
- Table 829 DO Payment Trans. Log Entry
- Table 1510 Notification Template

When you delete a table object, in the **Delete** confirmation dialog box that appears, set the **Synchronize Schema** option to **Force**.

IMPORTANT

In this step, it is very important that you do not use the **Sync. Schema For All Tables** option from the **Tools** menu.

4. (Upgrade from Microsoft Dynamics NAV 2016 and earlier only) If the old database includes test runner codeunits, you will get errors on these codeunits that the OnBeforeTestRun and OnAfterTestRun trigger signatures are not valid. To fix these issues, you change the signature of the OnBeforeTestRun and OnAfterTestRun triggers to include the *TestPermission* parameter.

For more information, see [Resolving OnBeforeTestRun and OnAfterTestRun Trigger Errors When Converting a Database](#).

The triggers for codeunit **130400 CAL Test Runner** and **130402 CAL Command Line Test Runner** will be updated for you during the data upgrade.

Task 10: (Upgrade from Dynamics NAV 2018 or Business Central Fall 2018 only) Increase the application version of converted database

You must increase the application version that is assigned to the database.

Use the [Set-NAVApplication](#) cmdlet of the Business Central Administration Shell to increase the application version number of the database from its current version.

To see the current version, use the following command:

```
Get-NAVApplication -ServerInstance <ServerInstanceName>
```

To increase the version by 1, run the following command:

```
Set-NAVApplication -ServerInstance <ServerInstanceName> -IncrementApplicationVersion
```

Or, to specify change to another version, run the following command:

```
Set-NAVApplication -ServerInstance <ServerInstanceName> -ApplicationVersion <N.N.N.N> -Force
```

For example, if the old version was `11.0.24279.0`, then you could change the version to `14.0.24279.0`.

Task 11: Run the schema synchronization on converted database

Synchronize the database schema with validation.

For example, run the [Sync-NAVTenant](#) cmdlet from the Business Central Administration Shell.

```
Sync-NAVTenant -ServerInstance <ServerInstanceName>
```

When completed, the tenant (database) should have the status **OperationalDataUpgradePending**. To verify this, run the following cmdlet:

```
Get-NAVTenant -ServerInstance <ServerInstanceName> -tenant default
```

For more information, see [Synchronizing the Tenant Database and Application Database](#).

Task 12: Run data upgrade on converted database

A data upgrade runs the upgrade toolkit objects, such as upgrade codeunits and upgrade tables, to migrate business data from the old table structure to the new table structure. You can start the data upgrade from the Dynamics NAV Development Environment or Business Central Administration Shell.

Open the Business Central Administration Shell as an administrator, and then run [Start-NavDataUpgrade](#) cmdlet as follows:

```
Start-NavDataUpgrade -ServerInstance <ServerInstanceName>
```

Replace `<ServerInstanceName>` with the name of the Business Central Server instance that is connected to the database.

NOTE

In the last phase of data upgrade, all companies will be initialized by running codeunit 2 Company Initialization. This is done automatically. If you want to skip company initialization, then use the `Start-NavDataUpgrade` with the `-SkipCompanyIntitilization` parameter.

To view the progress of the data upgrade, you can run Get-NavDataUpgrade cmdlet with the `-Progress` switch.

The data upgrade process runs `CheckPreconditions` and `Upgrade` functions in the upgrade codeunits. If any of the preconditions are not met or an upgrade function fails, you must correct the error and resume the data upgrade process. If CheckPreconditions and Upgrade functions are executed successfully, codeunit 2 is automatically run to initialize all companies in the database unless you set the `-SkipCompanyIntitilization` parameter.

Task 13: Upgrade Javascript-based control add-ins to new versions

The Business Central Server installation includes new versions of Microsoft-provided Javascript-based control add-ins, such as the Business Chart control add-in. If your application is using any of these add-ins, you must upgrade them to the new versions as follows:

1. Open the Business Central client.

If the application uses the Business Chart control add-in, you will get an error about **High Charts**. Upgrading the Business Chart control add-in will clear this error.

2. Search for and open the **Control Add-ins** page.

The page lists all the registered control add-ins.

3. Choose **Actions > Control Add-in Resource > Import**.

4. Locate and select the .zip file for the control add-in and choose **Open**.

The .zip files are located in the **Add-ins** folder of the Business Central Server installation. There is a sub-folder for each add-in. For example, the path to the Business Chart control add-in is

```
C:\Program Files\Microsoft Dynamics 365 Business Central\140\Service\Add-ins\BusinessChart\Microsoft.Dynamics.Nav.Client.BusinessChart.zip
```

5. After you have imported all the new control add-in versions, restart Web Server instance.

Task 14: Publish and install/upgrade extensions

Complete this task if you are upgrading from a Microsoft Dynamics NAV 2018 deployment that uses V2 extensions or a Denmark (DK) version of Microsoft Dynamics NAV 2017 or earlier.

The Business Central installation media (DVD) includes several new versions of Microsoft extensions (that is, extensions that have **Microsoft** as the publisher). If your old deployment uses these extensions, you have to upgrade the current versions to the new versions.

In addition, other extensions used in the old deployment that you still want to use must be repaired to work on the new platform.

IMPORTANT

For Denmark (DK) and German (DE) versions. Some of the local functionality has been moved from the base application to extensions.

If you are upgrading from a Denmark (DK) version of Dynamics NAV 2017 or earlier, you must publish and install the following extensions to get the local functionality:

NAME	EXTENSION PACKAGE
OIOUBL	OIOUBL.app
Payroll Data Import Definitions (DK)	ImportDKPayroll.app
Payment and Reconciliation Formats (DK)	FIK.app
Tax File Formats (DK)	VATReportsDK.app

If you are upgrading from a German (DE) version of Dynamics NAV or Business Central October 2018 (Cumulative Update 2 or earlier), you must publish and install the following extensions to get the local functionality:

NAME	EXTENSION PACKAGE
ELSTER VAT Localization for Germany	Elster.app

1. To get list of the extensions currently published on the application, run the following command from the Business Central Administration Shell:

```
Get-NAVAppInfo -ServerInstance <ServerInstanceName>
```

2. Publish the system.app and test.app symbol files.

If you installed the **AL Development Environment**, you can find the symbol files where you installed the environment, which by default is C:\Program Files (x86)\Microsoft Dynamics 365 Business Central\140. Otherwise, you can find the files in the **ModernDev** folder on the installation media.

To publish the symbols, open the Business Central Administration Shell as an administrator, and run the following command for each of the symbol files:

```
Publish-NAVApp -ServerInstance <ServerInstanceName> -Path <SymbolFilePath> -PackageType SymbolsOnly
```

3. Generate the application symbol references by using the finsql.exe file as follows:

- a. Make sure that **Enable loading application symbol references at server startup** (EnableSymbolLoadingAtServerStartup) is set on the Business Central Server instance.

For more information, see [Configuring Business Central Server](#).

- b. Open a command prompt as an administrator, change to the directory where the `finsql.exe` file has been installed as part of Dynamics NAV Development Environment, and then run the following command:

```
finsql.exe Command=generatesymbolreference, Database="<MyDatabaseName>", ServerName=  
<DatabaseServerName>\<DatabaseInstance>
```

Replace values for the `Database` and `ServerName` settings to suit.

If the application database contains test objects (ID 130000-139999), then make sure to exclude these objects when generating symbols. You can do this by using the `-Filter` parameter and running the command twice:

```
finsql.exe command=generatesymbolreference, ServerName=<DatabaseServerName>\<DatabaseInstance>, Database="<MyDatabaseName>, filter="Object ID=1..129999"
```

```
finsql.exe command=generatesymbolreference, ServerName=<DatabaseServerName>\<DatabaseInstance>, Database="<MyDatabaseName>, filter="Object ID=140000..1999999999"
```

NOTE

This command does not generate a file. It populates the **Object Metadata** table in the database.

- c. When you run the command, the console returns to an empty command prompt, and does not display or provide any indication about the status of the run. However, the `finsql.exe` may still be running in the background. It can take several minutes for the run to complete, and the symbols will not be generated until such time. You can see whether the `finsql.exe` is still running by using Task Manager and looking on the **Details** tab for **finsql.exe**.

When the process ends, a file named **navcommandresult.txt** is saved to the Dynamics NAV Client connected to Business Central installation folder. If the command succeeded, the file will contain text like `[0] [06/12/17 14:36:17] The command completed successfully in '177' seconds.` If the command failed, another file named **naverrorlog.txt** will be generated. This file contains details about the error(s) that occurred.

For more information about generation symbols, see [Running C/SIDE and AL Side-by-Side](#).

- d. Restart the Business Central Server instance.
4. Upgrade the Microsoft extensions that were published in the old deployment to new versions. For Denmark (DK) and German (DE), you must also complete this step to install the local functionality extensions mentioned at the start of this task.

The new extension versions are found in the `\Extensions` folder of the installation media (DVD). Follow these steps for each extension by using the Business Central Administration Shell:

- a. Publish the new extension version by running the [Publish-NAVApp](#) cmdlet:

```
Publish-NAVApp -ServerInstance <ServerInstanceName> -Path <ExtensionFileName>
```

- b. Synchronize the schema with the database by running the [Sync-NAVApp](#) cmdlet:

```
Sync-NAVApp -ServerInstance <ServerInstanceName> -Name <Name> -Version <N.N.N.N>
```

- c. Upgrade the data of the extensions. This step is not required for the newly published local functionality extensions.

To run the data upgrade, run the [Start-NAVAppDataUpgrade](#) cmdlet:

```
Start-NAVAppDataUpgrade -ServerInstance <ServerInstanceName> -Name <Name> -Version <N.N.N.N>
```

Apart from upgrading the data, this command will install the new extension version.

- d. Install the newly published local functionality extensions by running the [Install-NAVApp](#) cmdlet:

```
Install-NAVApp -ServerInstance <ServerInstanceName> -Name <Name> -Version <N.N.N.N>
```

For more information about publishing extensions, see [Publish and Install an Extension](#).

5. Repair, synchronize, and install other currently published extension versions that you still want to use, but have not been upgraded to new versions.

For each extension, complete the following steps from the Business Central Administration Shell:

- a. Compile the extension to make it work with the new platform by running the [Repair-NAVApp](#) cmdlet.

```
Repair-NAVApp -ServerInstance <ServerInstanceName> -Name <Extension Name> -Version <N.N.N.N>
```

- b. Synchronize the schema with the database by running the [Sync-NAVApp](#) cmdlet:

```
Sync-NAVApp -ServerInstance <ServerInstanceName> -Name <Name> -Version <N.N.N.N>
```

- c. Install the extension by running the [Install-NAVApp](#) cmdlet:

```
Install-NAVApp -ServerInstance <ServerInstanceName> -Name <Name> -Version <N.N.N.N>
```

6. (Optional) Unpublish unused extension versions by running the [Unpublish-NAVApp](#):

```
Unpublish-NAVApp -ServerInstance <ServerInstanceName> -Name <Name> -Version <N.N.N.N>
```

Task 15: Import permission sets and permissions

Import the permission sets and permissions XML files that you exported from the old database as follows:

- Upgrade from Dynamics NAV:

1. Open table **2000000004 Permission Sets** in the client, and delete all permission sets except SUPER.

NOTE

You are only required to delete those permission sets that are also included in the permission sets XML file that you will import. Because if you try to import a permission set with the same name as one already in the database, you will get an error.

2. Run XMLport **9171** and XMLport **9172** to import the permission sets and permission XML files.

For more information, see [How to: Export and Import Permission Sets and Permissions](#).

- Upgrade from an earlier Business Central version:

1. In the client, search for and open the **Permission Sets** page.
2. Delete all user-defined permissions.
3. Choose **Import Permission Sets**, then select the permissions set file that you exported previously.

Task 16: (Optional) Import data encryption key

If you want to use data encryption as before, you must import the data encryption key file that was exported previously.

For more information, see [Exporting and Importing Encryption Keys](#).

Task 17: Set the language of customer database

In the Dynamics NAV Development Environment, choose **Tools**, choose **Language**, and then select the language of the original customer database.

Task 18: (Optional) Update Web Server instance configuration file

If you have installed the Business Central Web Server, populate the navsettings.json file for the Business Central Web Server instance with the settings of the old web.config file or navsettings.json.

- If the old deployment used a web.config file, then you have to manually change the settings in the navsetting.json file that is used on the new Business Central Web Server instance.
- If you upgraded from Business Central October 2018, you can replace the navsettings.json file on the new Business Central Web Server instance with the old file. However, as of Business Central April 2019, the following settings are now configured under a root element called `ApplicationIdSettings` instead of the root element `NAVWebSettings`.

- `AndroidPrivacy`
- `AndroidSoftwareLicenseTerms`
- `AndroidThirdPartyNotice`
- `BaseHelpUrl`
- `BaseSettingsSectionName`
- `CommunityLink`
- `FeedbackLink`
- `IosPrivacy`
- `IosSoftwareLicenseTerms`
- `IosThirdPartyNotice`
- `KeyboardShortcutsLink`
- `PrivacyLink`
- `LegalLink`
- `SignInHelpLink`

If the old navsettings.json file uses any of these settings, then you will have to move them from the `NAVWebSettings` element to the `ApplicationIdSettings` element.

For more information about the navsettings.json file, see [Configuring Business Central Web Server Instances](#).

(Optional) Task 19: Delete upgrade objects

At this point, you have upgraded the database to Business Central. Now, you can delete the upgrade codeunits and upgrade table objects that you imported in task 9. This task is recommended but not required.

When you delete tables, on the **Delete** dialog box, set the **Synchronize Schema** option to **Force**.

See Also

Upgrading the Application Code
Upgrading to Business Central

Business Central Single-Tenant Full Upgrade Quick Reference

3/31/2019 • 2 minutes to read

This article provides an overview of the full upgrade process for Business Central in a single-tenant deployment. For more detailed steps, see [Upgrading the Data: Single-Tenant Mode](#).

Prerequisite tasks on old database

STEP	MORE INFO	DONE
Upgrade application code.	See...	
Convert custom V1 extensions to V2 extensions.	See...	
Export permissions and permission sets. Important: Make sure your computer uses the same codepage as the data.	See...	
Export encryption keys from the old deployment.	See...	
Prepare for transitioning from codeunit 1. Note: Dynamics NAV upgrade only	See...	
Install Business Central components.	See...	

Prepare the old database for data upgrade

STEP	MORE INFO	DONE
Backup the database.	See...	
Uninstall all extensions.	See...	
Upload a Business Central partner license.	See...	
Delete all objects except tables. Important: Do not synchronize schema at this point.	See...	
Clear server instance and debugger breakpoint tables.	See...	

Run the data upgrade

STEP	MORE INFO	DONE
Open Dynamics NAV Development Environment for Business Central as an administrator		
Connect to and convert the database.	See...	
Import upgraded application and upgrade toolkit objects (.fob files). Important: Select to synchronize later .	See...	
Connect a Business Central Server instance to the converted database.	See...	
Compile all objects. Important: Choose to synchronize schema later .	See...	
Increase the application version of the database, Note: Dynamics NAV 2018 upgrade only	See...	
Synchronize the database.	See...	
Run the data upgrade.	See...	
Update Javascript control add-ins the data upgrade.	See...	

Publish, upgrade, and install extensions

STEP	MORE INFO	DONE
Publish system and test symbols, generate application symbols.	See...	
Publish, synchronize, and upgrade to new versions of Microsoft extensions from installation media.	"	
Repair, synchronize, and install old extension versions that were not upgraded in previous step.	"	
Run the data upgrade on the new extension versions.	"	
Repair other custom extensions to work on new platform.	"	

Post-upgrade tasks

STEP	MORE INFO	DONE
Import permissions and permission sets.	See...	
Import encryption keys	See...	
Upload the customer license.	See...	

Upgrading the Data to Business Central: Multitenant Deployment

3/31/2019 • 10 minutes to read

[See print-friendly quick reference](#)

This article describes the tasks required for upgrading data to the latest Business Central in a multitenant deployment.

About Data Upgrade

In this scenario, you already have an upgraded application that is mounted on a Business Central Server. You will then mount the old tenants on the server instance and perform the data upgrade.

You use data conversion tools provided with Business Central to convert the old data with the old version's table and field structure, so that it functions together with the new version's table and field structure. Mainly, only table objects and table data are modified during the data upgrade process. Other objects, such as pages, reports, codeunits, and XMLports are upgraded as part of the application code upgrade process.

The data upgrade process described in this article leads you through the database conversion (technical upgrade) and then the upgrade of the actual data, which is achieved by using the upgrade toolkit/upgrade codeunits.

IMPORTANT

Before you begin, read the article [Important Information and Considerations for Before Upgrading](#). This article contains information about limitations and things that might require you to perform extra tasks before you upgrade, such as the use of extensions V1 and the deprecation of codeunit 1.

Prerequisites

Before you start the upgrade tasks, make sure you have the following prerequisites:

1. Your computer uses the same codepage as the data that will be upgraded.

If you use conflicting codepages, some characters will not display in captions, and you might not be able to access the upgraded database. This is because Business Central must remove incorrect metadata characters to complete the data upgrade. In this case, after upgrade, you must open the database in the development environment on a computer with the relevant codepage and compile all objects. This adds the missing characters again.

2. (Upgrading from Dynamics NAV only) Custom V1 extensions used in Dynamics NAV have been converted to V2 extensions.

For more information, see [Converting Extensions V1 to Extensions V2](#).

3. Business Central has been installed with the upgraded application and upgrade toolkit.

As a minimum, you must install the following components:

- Business Central Server instance connected to the application database.
- Dynamics NAV Development Environment for Business Central

- AL Development Environment

This installs the required system and test symbols for V2 extensions.

- Business Central Server Administration tool (optional)
- Business Central Web Server components (not required for upgrade).

For more information about upgrading the application code, see [Upgrading the Application Code](#).

4. Permission sets (except SUPER) and permissions have been exported from the old tenant database.

- When upgrading from Dynamics NAV

To export permission sets and permissions, you run running XMLPort 9171 and 9172.

It is important that you exclude the SUPER permission set when running XMLPort 9171. You can do this by adding the filter `Role ID is <>SUPER`.

For more information, see [Exporting and Importing Permission Sets and Permissions](#).

- When upgrading from an earlier version of Business Central

In the client, search for and open the **Permission Sets** page, select the user-defined permission sets that you want to keep, and then choose **Export Permission Sets**.

5. If the old application uses data encryption, you have the encryption key file that it used for the data encryption.

For more information, see [Export and Import Encryption Keys](#).

NOTE

If the old Dynamics NAV application uses Payment Services for Microsoft Dynamics ERP, be aware that this was discontinued in Microsoft Dynamics NAV 2017. This means that most of the objects that are associated with this feature will be deleted during the upgrade. Some objects you will have to manually delete.

Prepare the tenant database for data upgrade

You perform these tasks on each tenant that you want to upgrade.

1. Backup the tenant database.

Create a full backup of the old database in the SQL Server. Alternatively, you can make a copy of the old database and perform the upgrade tasks on the copy.

For more information, see [Create a Full Database Backup \(SQL Server\)](#).

2. (Dynamics NAV upgrade only) Uninstall all V1 extensions.

Make sure that all V1 extensions are uninstalled. Open the Dynamics NAV Administration Shell that matches to old database, and run these commands:

- a. To get a list of the V1 extensions that are installed, run this command:

```
Get-NAVAppInfo -ServerInstance <OldServerInstanceName> -Tenant <TenantID>
```

V1 extensions are indicated by `CSIDE` in the `Extension Type` column. 2. For each extension, run the [Uninstall-NAVApp](#) cmdlet to uninstall it:

```
Uninstall-NAVApp -ServerInstance <OldServerInstanceName> -Name <Name> -Version <N.N.N.N>
```

3. Dismount the tenant.

Before you upgrade the tenant, you must dismount it from the old server instance. To dismount the tenant, run the [Dismount-NAVTenant](#) cmdlet:

```
Dismount-NAVTenant -ServerInstance <OldServerInstanceName> -Tenant <TenantID>
```

Run the data upgrade on the tenant

You perform these tasks on each tenant that you want to upgrade.

1. Mount the tenant.

Mount the tenant on the new Business Central Server instance that connects to the newly application database. To mount the tenant, use the [Mount-NAVTenant](#) cmdlet:

```
Mount-NAVTenant -ServerInstance <ServerInstanceName> -DatabaseName <Database name> -DatabaseServer <server\instance> -Tenant <TenantID> -AllowAppDatabaseWrite
```

IMPORTANT

You must use the same tenant ID for the tenant that was used in the old deployment; otherwise you will get an error when mounting or syncing the tenant. If you want to use a different ID for the tenant, you can either use the `-AlternateId` parameter now or after upgrading, dismount the tenant, then mount it again using the new ID and the `OverwriteTenantIdInDatabase` parameter.

NOTE

For upgrade, we recommend that you use the `-AllowAppDatabaseWrite` parameter. After upgrade, you can dismount and mount the tenant again without the parameter if needed.

2. Synchronize the tenant.

Synchronize the tenant database schema with validation by running the [Sync-NAVTenant](#) cmdlet from the Business Central Administration Shell.

```
Sync-NAVTenant -ServerInstance <ServerInstanceName> -Tenant <TenantID>
```

When completed, the tenant should have the status **OperationalDataUpgradePending** or, if there are published extensions with newer versions than on the tenant, **OperationalSyncPending**. To verify this, run the following cmdlet:

```
Get-NAVTenant -ServerInstance <ServerInstanceName> -tenant default -ForceRefresh
```

3. If there are published extensions with newer versions than on the tenant, synchronize all published extensions with the tenant database.

Synchronize the schema with the database by running the [Sync-NAVApp](#) cmdlet for each extension version:

```
Sync-NAVApp -ServerInstance <ServerInstanceName> -Name <Name> -Version <N.N.N.N> -Tenant <TenantID>
```

Or, to synchronize all published extensions using one command:

```
Get-NAVAppInfo -ServerInstance <ServerInstanceName> -Tenant <TenantID> | % { Sync-NAVApp -ServerInstance  
<ServerInstanceName> -Name $_.Name -Version $_.Version }
```

When completed, the tenant should have the status **OperationalDataUpgradePending**.

4. Run the data upgrade.

A data upgrade runs the upgrade toolkit objects, such as upgrade codeunits and upgrade tables, to migrate business data from the old table structure to the new table structure. It will also upgrade the published extensions.

You can start the data upgrade by running the run [Start-NavDataUpgrade](#) cmdlet the Business Central Administration Shell:

```
Start-NavDataUpgrade -ServerInstance <ServerInstanceName>
```

Important: If you have extensions, then you must run the data upgrade so that it executes functions in the serial mode as follows.

```
Start-NavDataUpgrade -ServerInstance <ServerInstanceName> -Tenant <tenantID> -FunctionExecutionMode  
Serial
```

To view the progress of the data upgrade, you can run [Get-NavDataUpgrade](#) cmdlet with the `-Progress` switch.

The data upgrade process runs `CheckPreconditions` and `Upgrade` functions in the upgrade codeunits. If any of the preconditions are not met or an upgrade function fails, you must correct the error and resume the data upgrade process. If `CheckPreconditions` and `Upgrade` functions are executed successfully, codeunit 2 is automatically run to initialize all companies in the database unless you set the `-SkipCompanyIntitialization` parameter.

5. Install extensions on the tenant.

Install the desired extensions on the tenant by running the [Install-NAVApp](#) cmdlet:

```
Install-NAVApp -ServerInstance <ServerInstanceName> -Name <Name> -Version <N.N.N.N>
```

Post-upgrade tasks

1. Import permission sets and permissions

Import the permission sets and permissions XML files that you exported from the old database as follows:

- For upgrade from Dynamics NAV:
 - a. Open table **2000000004 Permission Sets** in the client, and delete all permission sets except SUPER.

NOTE

You are only required to delete those permission sets that also included in the permission sets XML file that you will import. Because if you try to import a permission set with the same name as on already in the database, you will get an error.

- b. Run XMLport **9171** and XMLport **9172** to import the permission sets and permission XML files.

For more information, see [How to: Export and Import Permission Sets and Permissions](#).

- For upgrade from an earlier Business Central version:
 - a. In the client, search for and open the **Permission Sets** page.
 - b. Delete all user-defined permissions.
 - c. Choose **Import Permission Sets**, then select the permissions set file that you exported previously.

2. Import encryption keys.

For more information, see [Exporting and Importing Encryption Keys](#).

3. (Optional) Update Web Server instance configuration file

If you have installed the Business Central Web Server, populate the navsettings.json file for the Business Central Web Server instance with the settings of the old web.config file or navsettings.json.

- If the old deployment used a web.config file, then you have to manually change the settings in the navsetting.json file that is used on the new Business Central Web Server instance.
- If you upgraded from Business Central October 2018, you can replace the navsettings.json file on the new Business Central Web Server instance with the old file. However, as of Business Central April 2019, the following settings are now configured under a root element called `ApplicationIdSettings` instead of the root element `NAVWebSettings`.

- `AndroidPrivacy`
- `AndroidSoftwareLicenseTerms`
- `AndroidThirdPartyNotice`
- `BaseHelpUrl`
- `BaseSettingsSectionName`
- `CommunityLink`
- `FeedbackLink`
- `IosPrivacy`
- `IosSoftwareLicenseTerms`
- `IosThirdPartyNotice`
- `KeyboardShortcutsLink`
- `PrivacyLink`
- `Legallink`
- `SignInHelpLink`

If the old navsettings.json file uses any of these settings, then you will have to move them from the `NAVWebSettings` element to the `ApplicationIdSettings` element.

For more information about the navsettings.json file, see [Configuring Business Central Web Server Instances](#).

4. Upload the customer license.

For more information, see [Uploading the License File](#)

See Also

[Upgrading the Application Code](#)

[Upgrading to Business Central](#)

Business Central Multitenant Full Upgrade Quick Reference

3/31/2019 • 2 minutes to read

This article provides an overview of the full upgrade process for Business Central in a multitenant deployment. For more detailed steps, see [Upgrading the Data: Multitenant Mode](#).

Prerequisite tasks

STEP	MORE INFO	DONE
In the old deployment, convert custom V1 extensions to V2 extensions.	See...	
Export permissions and permission sets from the old deployment. Important: Make sure computer uses the same codepage as the data.	See...	
Export encryption keys from the old deployment.	See...	
Prepare for transitioning from codeunit 1.	See...	
Install Business Central components.	See...	

Upgrade the application and prepare it for data upgrade

STEP	MORE INFO	DONE
Upgrade the application code.	See...	
Mount the upgraded application on the Business Central Server instance.	See...	
Import upgrade toolkit (.fob)	See...	
Publish system and test symbols from the installation media, and generate application symbols.	See...	
Publish the new Microsoft extension versions from the installation media.	See...	
Upload a Business Central partner license.	See...	

Prepare the tenant database for data upgrade

STEP	MORE INFO	DONE
Backup the tenant database.	See...	
Uninstall all V1 extensions.	See...	
Dismount the tenant from the old server instance.	See...	

Run the data upgrade on the tenant

STEP	MORE INFO	DONE
Mount the tenant on the Business Central Server instance. Important: Use the <code>-AllowAppDatabaseWrite</code> parameter.	See...	
Synchronize the tenant.	See...	
Synchronize all extensions.	See..	
Run the data upgrade. Important: If there are V2 extensions, you must use the <code>-FunctionExecutionMode Serial</code> parameter.	See...	
Install the new V2 extensions that were not installed in the old tenant.	See...	

Post-upgrade tasks

STEP	MORE INFO	DONE
Import permissions and permission sets.	See...	
Import encryption keys	See...	
Upload the customer license.	See...	

Important Information and Considerations for Before Upgrading to Dynamics 365 Business Central

4/4/2019 • 5 minutes to read

Depending on which version you are upgrading from, and the degree to which your solution differs from the standard version of Business Central, you may want to prepare your solution for the upgrade. This topic provides important information and tips for things to consider when you prepare to upgrade to Business Central.

Upgrading from Dynamics NAV to Business Central online

You can upgrade to Business Central online from supported versions of Dynamics NAV on-premises, provided that your application customization is handled by extensions. Any data from tables with code customizations cannot be carried forward from Dynamics NAV.

IMPORTANT

Upgrading from Dynamics NAV to Business Central online is only partially supported. In the current version of Business Central, when you connect your on-premises solution to the intelligent cloud, the on-premises deployment remains the primary application, and the cloud tenant is, with very few exceptions, read-only. For more information, see [Connect to the intelligent cloud](#).

The process consists of two parts:

- Upgrade from Dynamics NAV to Business Central using the tools described in [Upgrading to Business Central on-premises](#).
- Convert non-standard functionality and customizations to apps and per-tenant extensions. For more information, see [Deploying a Tenant Customization](#).
- Enable replication to a cloud tenant as described in [Connect to the intelligent cloud](#), and then switch to use the Business Central online tenant going forward.

Upgrading from Dynamics NAV

Codeunit 1 has been deprecated and replaced

Dynamics NAV included codeunit **1 ApplicationManagement**. In Business Central, this codeunit has been retired, and new 'system' codeunits have been introduced in the 2 billion range.

For information, see [Transitioning from Codeunit 1 to System Codeunits](#).

V1 Extensions have been discontinued

With Business Central, extensions V1 are no longer supported for on-premise installations. As a result, any custom extensions V1 must be converted to extensions V2 in the old environment before upgrading to Business Central.

For information about how to convert to extensions V2, see [Converting Extensions V1 to Extensions V2](#).

MenuSuite not used for page and report search

With Business Central, the MenuSuite is no longer used to control whether a page or report can be found in the search feature of the Web client. This is now determined by specific properties on the page and report objects. As part of the application code upgrade process, you change these properties on existing pages and reports used by the MenuSuite to ensure that they are still searchable from the Web client. For more information, see [Making](#)

Dynamics 365 for Sales integration

Because of changes in Dynamics 365 for Sales and the integration since previous releases, if your application is integrating with Dynamics 365 for Sales, then you must perform a full upgrade instead of just a technical upgrade.

New and changed application features

There are several new and changed application features available in Business Central April 2019 for users, administrators, and developers. For an overview of these features, see [Overview of Dynamics 365 Business Central April '19 release](#).

To take advantage of these all these features, you will have to perform an application code upgrade, not just a technical (platform) upgrade.

Changes to profiles in the CRONUS International Ltd. demonstration database and promoted actions

With the Business Central April 2019 release, profiles that are part of the CRONUS International Ltd. demonstration database, such as the **Sales Order Processor** profile, customize fewer pages compared to earlier releases. For customers that rely on these profiles, their users might experience slight differences in the layout of actions in the action bar on pages. Additionally, the layout of promoted actions on over 380 core application pages has been fine-tuned.

To ensure that users are not disrupted by these changes, we recommend that administrators and partners who are upgrading a customer to Business Central April 2019, review the layout of promoted actions when combined with their own code and profile customization.

Names of variables

Business Central introduces new methods and statements. If your solution includes variables where the name is now used by a standard AL method or statement such as `REGISTERTABLECONNECTION` or `FOREACH`, you must change the variables before you upgrade to Business Central. Alternatively, you can enclose the variable names in quotation marks. If you do not, and you import an object that has this code in text format, you cannot compile the object.

Deprecated or redesigned functionality

If you are upgrading a solution that depends on functionality that is deprecated or changed in the default version of Business Central, you must verify that the upgrade codeunits migrate data correctly. See the [See Also](#) section for links to descriptions of deprecated functionality.

Deprecated fields and fields marked as obsolete

Sometimes Microsoft will refactor code so that fields are no longer used, or the functionality is moved from the base application to an extension, for example. Typically, the upgrade toolkit will manage the upgrade impact, but for transparency, you can find a list of fields that are deprecated in the current release or marked to be obsolete in a later release. For more information, see [Deprecated Fields, and Fields Marked as Obsolete](#).

Upgrade codeunits

When you introduce changes to the database schema, Business Central will check if these changes are destructive or not. If the database check indicates that the change may lead to data deletion, such as if you are dropping a table column so that the contents of that column will be deleted, this is considered a destructive change. You will be prompted to handle the situation using upgrade codeunits.

Company names

If a company name includes a special character, an error may display during the upgrade. In this context, special characters include the following:

[~ @ # \$ % & * () . ! % - + / = ?]

If you are going to upgrade a database where one or more company name includes a special character, we recommend that you rename the company before you start the upgrade process. After the upgrade is successfully finished, you can rename the company again.

System tables with non-English names

In older versions of Dynamics NAV, you could translate the columns in system tables to a language other than English. Starting with version 3.0, we advised heavily against this, and versions later than Microsoft Dynamics NAV 2013 R2 require that all columns in all system tables are in English. As a result, if you try to open a database with non-English system tables in Microsoft Dynamics NAV 2013 R2 or later, an error displays, saying that one or more columns do not exist.

Make sure that all objects were compiled in a development environment with the right .ETX and .STX files. You can verify that you are running in the correct environment with English (US) as the base language by opening the **ndo\$dbproperty** table in SQL Server Management Studio. In the **Identifiers** column, the word `object` must be written exactly as shown here.

See Also

[Upgrading the Application Code](#)

[Upgrading the Data](#)

[Deprecated Fields, and Fields Marked as Obsolete](#)

Deprecated Fields, and Fields Marked as Obsolete

5/28/2019 • 18 minutes to read

In the latest version of Business Central, a number of fields have been deprecated in the current release or marked to be obsolete in a later release.

Definitions

Deprecated fields fall into one of the following groups:

1. Fields moved to an extension by Microsoft

Partner impact: Remember to install the extension when you upgrade an existing solution from an earlier version of Business Central.

Specifically for the extensions that are required for connecting on-premises solutions with Business Central online for intelligent insights, you must install the **Intelligent Cloud Base Extension** extension first, and then the product-specific extension or extensions.

2. Fields marked as Obsolete:Pending

Partner impact: None in the current release, this is just a heads-up that a change is coming.

3. Fields no longer in use in Microsoft code

Partner impact: Refactor your code as soon as possible.

Fields marked as Obsolete:Pending in Business Central

A number of fields are marked as Obsolete:Pending. There is no impact on code in this release.

Austria

The following fields are marked as Obsolete:Pending in the AT version.

TABLE ID	TABLE NAME	FIELD ID	FIELD NAME	COMMENTS
98	General Ledger Setup	11011	Sales VAT Advance Notif. Nos.	Will be removed in a later release.
242	Source Code Setup	5005350	Phys. Invt. Order	Will be removed in a later release.
257	VAT Statement Name	11000	Sales VAT Adv. Notification	Will be removed in a later release.
313	Inventory Setup	5005350	Phys. Inv. Order Nos.	Will be removed in a later release.
313	Inventory Setup	5005352	Posted Phys. Inv. Order Nos.	Will be removed in a later release.
11011	Sales VAT Advance Notification	All	All	Will be removed in a later release.

TABLE ID	TABLE NAME	FIELD ID	FIELD NAME	COMMENTS
11012	Transmission Log Entry	All	All	Will be removed in a later release.
11013	Electronic VAT Decl. Setup	All	All	Will be removed in a later release.
11014	Certificate	All	All	Will be removed in a later release.
5005350	Phys. Inventory Order Header	All	All	Will be removed in a later release.
5005351	Phys. Inventory Order Line	All	All	Will be removed in a later release.
5005352	Phys. Invt. Recording Header	All	All	Will be removed in a later release.
5005353	Phys. Invt. Recording Line	All	All	Will be removed in a later release.
5005354	Post. Phys. Invt. Order Header	All	All	Will be removed in a later release.
5005355	Posted Phys. Invt. Order Line	All	All	Will be removed in a later release.
5005356	Posted Phys. Invt. Rec. Header	All	All	Will be removed in a later release.
5005357	Posted Phys. Invt. Rec. Line	All	All	Will be removed in a later release.
5005358	Phys. Inventory Comment Line	All	All	Will be removed in a later release.
5005359	Posted Phys. Invt. Track. Line	All	All	Will be removed in a later release.
5005360	Phys. Invt. Tracking Buffer	All	All	Will be removed in a later release.
5005361	Expect. Phys. Inv. Track. Line	All	All	Will be removed in a later release.
5005362	Post. Exp. Ph. In. Track. Line	All	All	Will be removed in a later release.

Belgium

The following fields are marked as ObsoleteState:Pending in the BE version.

TABLE ID	TABLE NAME	FIELD ID	FIELD NAME	COMMENTS
4	Currency	2000000	ISO Currency Code	Will be removed in a later release.
9	Country/Region	2000000	ISO Country/Region Code	Will be removed in a later release.
11306	Electronic Banking Setup	22	Notification E-mail address	Will be removed in a later release.
11306	Electronic Banking Setup	23	Language	Will be removed in a later release.
11306	Electronic Banking Setup	31	IBS Service Version	Will be removed in a later release.
11306	Electronic Banking Setup	21	IBS Version	Will be removed in a later release.
11306	Electronic Banking Setup	24	Upload Integration Mode	Will be removed in a later release.
11306	Electronic Banking Setup	29	IBS Log Download Nos.	Will be removed in a later release.
11306	Electronic Banking Setup	40	Test Environment	Will be removed in a later release.
11306	Electronic Banking Setup	30	IBS Request ID	Will be removed in a later release.
11306	Electronic Banking Setup	28	IBS Log Upload Nos.	Will be removed in a later release.
11306	Electronic Banking Setup	25	Upload Path	Will be removed in a later release.
11306	Electronic Banking Setup	26	Download Integration Mode	Will be removed in a later release.
11306	Electronic Banking Setup	27	Download Path	Will be removed in a later release.
2000010	IBS Log	All	All	Will be removed in a later release.
2000011	IBS Contract	All	All	Will be removed in a later release.
2000012	IBS Account	All	All	Will be removed in a later release.
2000013	IBS Account Conflict	All	All	Will be removed in a later release.

Canada

The following fields are marked as ObsoleteState:Pending in the CA version.

TABLE ID	TABLE NAME	FIELD ID	FIELD NAME	COMMENTS
98	General Ledger Setup	10004	SAT Certificate Thumbprint	Will be removed in a later release.
10000	PAC Web Service	22	Certificate Thumbprint	Will be removed in a later release.

Germany

The following fields are marked as ObsoleteState:Pending in the DE version.

TABLE ID	TABLE NAME	FIELD ID	FIELD NAME	COMMENTS
98	General Ledger Setup	11011	Sales VAT Advance Notif. Nos.	Will be removed in a later release.
242	Source Code Setup	5005350	Phys. Invt. Order	Will be removed in a later release.
257	VAT Statement Name	11000	Sales VAT Adv. Notification	Will be removed in a later release.
313	Inventory Setup	5005352	Posted Phys. Inv. Order Nos.	Will be removed in a later release.
313	Inventory Setup	5005350	Phys. Inv. Order Nos.	Will be removed in a later release.
11011	Sales VAT Advance Notification	All	All	Will be removed in a later release.
11012	Transmission Log Entry	All	All	Will be removed in a later release.
11013	Electronic VAT Decl. Setup	All	All	Will be removed in a later release.
11014	Certificate	All	All	Will be removed in a later release.
5005350	Phys. Inventory Order Header	All	All	Will be removed in a later release.
5005351	Phys. Inventory Order Line	All	All	Will be removed in a later release.
5005352	Phys. Invt. Recording Header	All	All	Will be removed in a later release.
5005353	Phys. Invt. Recording Line	All	All	Will be removed in a later release.

TABLE ID	TABLE NAME	FIELD ID	FIELD NAME	COMMENTS
5005354	Post. Phys. Invt. Order Header	All	All	Will be removed in a later release.
5005355	Posted Phys. Invt. Order Line	All	All	Will be removed in a later release.
5005356	Posted Phys. Invt. Rec. Header	All	All	Will be removed in a later release.
5005357	Posted Phys. Invt. Rec. Line	All	All	Will be removed in a later release.
5005358	Phys. Inventory Comment Line	All	All	Will be removed in a later release.
5005359	Posted Phys. Invt. Track. Line	All	All	Will be removed in a later release.
5005360	Phys. Invt. Tracking Buffer	All	All	Will be removed in a later release.
5005361	Expect. Phys. Inv. Track. Line	All	All	Will be removed in a later release.
5005362	Post. Exp. Ph. In. Track. Line	All	All	Will be removed in a later release.

Mexico

The following fields are marked as ObsoleteState:Pending in the MX version.

TABLE ID	TABLE NAME	FIELD ID	FIELD NAME	COMMENTS
98	General Ledger Setup	10004	SAT Certificate Thumbprint	Will be removed in a later release.
10000	PAC Web Service	22	Certificate Thumbprint	Will be removed in a later release.

Netherlands

The following fields are marked as ObsoleteState:Pending in the NL version.

TABLE ID	TABLE NAME	FIELD ID	FIELD NAME	COMMENTS
312	Purchases & Payables Setup	11312	Show Totals on Purch. Inv./CM.	Will be removed in a later release.

Spain

The following fields are marked as ObsoleteState:Pending in the ES version.

TABLE ID	TABLE NAME	FIELD ID	FIELD NAME	COMMENTS
----------	------------	----------	------------	----------

TABLE ID	TABLE NAME	FIELD ID	FIELD NAME	COMMENTS
10751	SII Setup	9	IntracommunityEndp ointUrl	Will be removed in a later release.

Switzerland

The following fields are marked as ObsoleteState:Pending in the CH version.

TABLE ID	TABLE NAME	FIELD ID	FIELD NAME	COMMENTS
4	Currency	3010541	ISO Currency Code	Will be removed in a later release.
98	General Ledger Setup	11011	Sales VAT Advance Notif. Nos.	Will be removed in a later release.
98	General Ledger Setup	11011	Sales VAT Advance Notif. Nos.	Will be removed in a later release.
98	General Ledger Setup	11004	Post Pmt.Disc Tol. to Pmt.Disc	Will be removed in a later release.
242	Source Code Setup	5005350	Phys. Invt. Order	Will be removed in a later release.
257	VAT Statement Name	11000	Sales VAT Adv. Notification	Will be removed in a later release.
311	Sales & Receivables Setup	11501	Line Amt. Round LCY	Will be removed in a later release.
313	Inventory Setup	5005350	Phys. Inv. Order Nos.	Will be removed in a later release.
313	Inventory Setup	5005352	Posted Phys. Inv. Order Nos.	Will be removed in a later release.
11011	Sales VAT Advance Notification	All	All	Will be removed in a later release.
11012	Transmission Log Entry	All	All	Will be removed in a later release.
11013	Electronic VAT Decl. Setup	All	All	Will be removed in a later release.
11014	Certificate	All	All	Will be removed in a later release.
5005350	Phys. Inventory Order Header	All	All	Will be removed in a later release.
5005351	Phys. Inventory Order Line	All	All	Will be removed in a later release.

TABLE ID	TABLE NAME	FIELD ID	FIELD NAME	COMMENTS
5005352	Phys. Invt. Recording Header	All	All	Will be removed in a later release.
5005353	Phys. Invt. Recording Line	All	All	Will be removed in a later release.
5005354	Post. Phys. Invt. Order Header	All	All	Will be removed in a later release.
5005355	Posted Phys. Invt. Order Line	All	All	Will be removed in a later release.
5005356	Posted Phys. Invt. Rec. Header	All	All	Will be removed in a later release.
5005357	Posted Phys. Invt. Rec. Line	All	All	Will be removed in a later release.
5005358	Phys. Inventory Comment Line	All	All	Will be removed in a later release.
5005359	Posted Phys. Invt. Track. Line	All	All	Will be removed in a later release.
5005360	Phys. Invt. Tracking Buffer	All	All	Will be removed in a later release.
5005361	Expect. Phys. Inv. Track. Line	All	All	Will be removed in a later release.
5005362	Post. Exp. Ph. In. Track. Line	All	All	Will be removed in a later release.

United Kingdom

The following fields are marked as ObsoleteState:Pending in the GB version.

TABLE ID	TABLE NAME	FIELD ID	FIELD NAME	COMMENTS
10533	MTD-Liability	All	All	Will be removed in a later release.
10534	MTD-Payment	All	All	Will be removed in a later release.
10535	MTD-Return Details	All	All	Will be removed in a later release.

United States

The following fields are marked as ObsoleteState:Pending in the US version.

TABLE ID	TABLE NAME	FIELD ID	FIELD NAME	COMMENTS
98	General Ledger Setup	10004	SAT Certificate Thumbprint	Will be removed in a later release.
10000	PAC Web Service	22	Certificate Thumbprint	Will be removed in a later release.

Fields no longer in use in Microsoft code in Business Central

A number of fields that are related to pictures are no longer in use, because the pictures are now stored in Image fields that are of Media type, in Business Central. The fields are marked as ObsoleteState:Pending.

TABLE ID	FIELD ID	COMMENTS		
18	Customer	89	Picture	Will be removed in a later release.
23	Vendor	89	Picture	Will be removed in a later release.
130	Incoming Document	20	URL2	Will be removed in a later release.
130	Incoming Document	21	URL3	Will be removed in a later release.
130	Incoming Document	22	URL4	Will be removed in a later release.
156	Resource	52	Picture	Will be removed in a later release.
167	Job	57	Picture	Will be removed in a later release.
270	Bank Account	89	Picture	Will be removed in a later release.
5050	Contact	89	Picture	Will be removed in a later release.
5200	Employee	19	Picture	Will be removed in a later release.
5600	Fixed Asset	22	Picture	Will be removed in a later release.

For more information about Media Data Type, see [Media Data Type](#) documentation.

A field that is related to VAT Registration Number validation is no longer in use, because the feature was replaced EU VAT Registration No. Validation Service Setup Business Central. The field is marked as ObsoleteState:Pending.

TABLE ID	FIELD ID	COMMENTS		
98	General Ledger Setup	161	VAT Reg. No. Validation URL	Will be removed in a later release.

For more information about validation of VAT Registration Numbers, see [Setting Up Calculations and Posting Methods for Value-Added Tax](#) documentation.

A flow field that was used to calculate Balance in My Account page is no longer in use in Business Central and has been replaced with Account Balance field to improve performance. The field is marked as ObsoleteState:Pending.

TABLE ID	FIELD ID	COMMENTS		
9153	My Account	4	Balance	Will be removed in a later release.

Fields moved to an extension by Microsoft in Business Central

A number of fields have been moved from the base application to an extension.

Denmark

The functionality for payments and reconciliation in the Danish version (FIK) has been moved to the Payments and Reconciliations (DK) extension. For more information, see [The Payments and Reconciliations \(DK\) Extension](#) in the Dynamics 365 Business Central documentation.

TABLE ID	TABLE NAME	OLD FIELD ID	NEW FIELD ID	OLD FIELD NAME	NEW FIELD NAME
23	Vendor	13650	13651	Giro Acc No.	GiroAccNo
25	Vendor Ledger Entry	13650	13651	Giro Acc No.	GiroAccNo
38	Purchase Header	13650	13651	Giro Acc No.	GiroAccNo
79	Company Information	13600	13651	Bank Creditor No.	BankCreditorNo
81	General Journal Line	13650	13651	Giro Acc No.	GiroAccNo
122	Purchase Invoice Header	13650	13651	Giro Acc No.	GiroAccNo
273	Bank Acc. Reconciliation	13600	13601	FIK Payment Reconciliation	FIKPaymentReconciliation
274	Bank Acc. Reconciliation Line	13600	13601	Payment Reference	PaymentReference
289	Payment Method	13601	13652	Payment Type Validation	PaymentTypeValidation

TABLE ID	TABLE NAME	OLD FIELD ID	NEW FIELD ID	OLD FIELD NAME	NEW FIELD NAME
372	Payment Buffer	13650	13651	Giro Acc No.	GiroAccNo
1226	Payment Export Data	13650	13651	Recipient Giro Acc No.	RecipientGiroAccNo
1250	Bank Statement Matching Buffer	13601	13652	Match Status	MatchStatus
1250	Bank Statement Matching Buffer	13600	13653	Description	DescriptionBankStatement

Fields marked as ObsoleteState:Pending in Business Central

A number of fields are marked as ObsoleteState:Pending. There is no impact on code in this release.

Iceland

The following fields are marked as ObsoleteState:Pending in the IS version.

TABLE ID	FIELD ID	COMMENTS
21	10900	Will be removed in a later release.
311	10900	Will be removed in a later release.

United Kingdom

The following fields are marked as ObsoleteState:Pending in the UK version.

TABLE ID	FIELD ID	COMMENTS
18	10500	Will be removed in a later release.
23	10500	Will be removed in a later release.
36	10501	Will be removed in a later release.
38	10501	Will be removed in a later release.
112	10501	Will be removed in a later release.
114	10501	Will be removed in a later release.
122	10501	Will be removed in a later release.
124	10501	Will be removed in a later release.
5107	10501	Will be removed in a later release.
5109	10501	Will be removed in a later release.

A number of fields are also deleted in the UK version. For more information, see [Deprecated Features in the UK Version](#).

Denmark

The following fields are marked as ObsoleteState:Pending in the Danish version.

TABLE ID	FIELD ID	COMMENTS
3	13600	Will be removed in a later release.
4	13600	Will be removed in a later release.
9	13600	Will be removed in a later release.
18	13605	Will be removed in a later release.
36	13600	Will be removed in a later release.
36	13601	Will be removed in a later release.
36	13602	Will be removed in a later release.
36	13604	Will be removed in a later release.
36	13605	Will be removed in a later release.
36	13606	Will be removed in a later release.
36	13607	Will be removed in a later release.
36	13620	Will be removed in a later release.
37	13600	Will be removed in a later release.
295	13600	Will be removed in a later release.
295	13602	Will be removed in a later release.
295	13605	Will be removed in a later release.
295	13606	Will be removed in a later release.
295	13607	Will be removed in a later release.
295	13608	Will be removed in a later release.
295	13620	Will be removed in a later release.
296	13600	Will be removed in a later release.
297	13600	Will be removed in a later release.
297	13601	Will be removed in a later release.
297	13602	Will be removed in a later release.

TABLE ID	FIELD ID	COMMENTS
297	13605	Will be removed in a later release.
297	13606	Will be removed in a later release.
297	13607	Will be removed in a later release.
297	13608	Will be removed in a later release.
297	13620	Will be removed in a later release.
298	13600	Will be removed in a later release.
302	13600	Will be removed in a later release.
302	13601	Will be removed in a later release.
302	13602	Will be removed in a later release.
302	13605	Will be removed in a later release.
302	13606	Will be removed in a later release.
302	13607	Will be removed in a later release.
302	13608	Will be removed in a later release.
302	13620	Will be removed in a later release.
303	13600	Will be removed in a later release.
304	13600	Will be removed in a later release.
304	13601	Will be removed in a later release.
304	13602	Will be removed in a later release.
304	13605	Will be removed in a later release.
304	13606	Will be removed in a later release.
304	13607	Will be removed in a later release.
304	13608	Will be removed in a later release.
304	13620	Will be removed in a later release.
305	13600	Will be removed in a later release.
311	13600	Will be removed in a later release.

TABLE ID	FIELD ID	COMMENTS
311	13601	Will be removed in a later release.
311	13602	Will be removed in a later release.
311	13603	Will be removed in a later release.
311	13604	Will be removed in a later release.
5107	13600	Will be removed in a later release.
5107	13601	Will be removed in a later release.
5107	13602	Will be removed in a later release.
5107	13605	Will be removed in a later release.
5107	13606	Will be removed in a later release.
5107	13607	Will be removed in a later release.
5107	13608	Will be removed in a later release.
5107	13620	Will be removed in a later release.
5108	13602	Will be removed in a later release.
5900	13600	Will be removed in a later release.
5900	13601	Will be removed in a later release.
5900	13604	Will be removed in a later release.
5900	13608	Will be removed in a later release.
5900	13620	Will be removed in a later release.
5902	13600	Will be removed in a later release.
5911	13600	Will be removed in a later release.
5911	13601	Will be removed in a later release.
5992	13600	Will be removed in a later release.
5992	13601	Will be removed in a later release.
5992	13602	Will be removed in a later release.
5992	13604	Will be removed in a later release.

TABLE ID	FIELD ID	COMMENTS
5992	13608	Will be removed in a later release.
5992	13620	Will be removed in a later release.
5993	13600	Will be removed in a later release.
5994	13600	Will be removed in a later release.
5994	13601	Will be removed in a later release.
5994	13602	Will be removed in a later release.
5994	13604	Will be removed in a later release.
5994	13608	Will be removed in a later release.
5994	13620	Will be removed in a later release.
5995	13600	Will be removed in a later release.

Fields no longer in use in Microsoft code in Business Central

A number of fields that are related to product groups are no longer in use, because the feature was replaced by item categories in Microsoft Dynamics NAV 2017. The fields are marked as ObsoleteState:Pending.

TABLE ID	FIELD ID	COMMENTS
5723	All	Deprecated. Do not use.
27	5704	Will be removed in a later release.
32	5707	Will be removed in a later release.
37	5712	Will be removed in a later release.
83	5707	Will be removed in a later release.
111	5712	Will be removed in a later release.
113	5712	Will be removed in a later release.
115	5712	Will be removed in a later release.
123	5712	Will be removed in a later release.
125	5712	Will be removed in a later release.
246	5705	Will be removed in a later release.
753	5707	Will be removed in a later release.

TABLE ID	FIELD ID	COMMENTS
5108	5712	Will be removed in a later release.
5110	5712	Will be removed in a later release.
5741	5712	Will be removed in a later release.
5745	5707	Will be removed in a later release.
5747	5707	Will be removed in a later release.
5902	5712	Will be removed in a later release.
5991	5712	Will be removed in a later release.
5993	5712	Will be removed in a later release.
6651	5712	Will be removed in a later release.
6661	5712	Will be removed in a later release.

For more information about the impact, see [The new Item Categories feature replaced the Product Group feature in Dynamics NAV 2017](#) on the Dynamics NAV team blog. For more information about item categories, see [How to: Categorize Items](#) in the Dynamics 365 Business Central documentation.

See Also

[Upgrading to Business Central](#)

[Upgrading the Application Code](#)

[Important Information and Considerations for Before Upgrading to Dynamics 365 Business Central](#)

Deprecated Features in the Austrian Version of Dynamics 365 Business Central

6/25/2019 • 2 minutes to read

This topic lists and describes the local functionality for Austria that has been removed from Business Central, made available from a new page or report, or replaced by a new feature.

Copy Existing Items to Create New Items

When you add a new item, to save time, you can use the **Copy Item** function to copy an existing item to use as a template for a new item.

MOVED, REMOVED, OR REPLACED?	WHY?
Moved	The Copy Item feature is no longer specific to Austria, so we have made it generally available in the standard product.

Physical Inventory Order

You can take inventory of your items by using the **Physical Inventory Order** and **Physical Inventory Recording** pages. The physical inventory order contains data for planning, realizing, and analyzing physical inventory. The physical inventory recording contains the items and quantities to be counted and forms the basis of the print-out to be used in the warehouse.

MOVED, REMOVED, OR REPLACED?	WHY?
Moved	The Physical Inventory Order feature is no longer specific to Austria, so we have made it generally available in the standard product.

Blanket Order Archiving and Document Line Tracking

You can archive and delete blanket sales and purchase orders. You can view documents that are related to sales order lines and purchase order lines, including from archived order lines. Related documents that you can track include quotes, shipments, receipts, and blanket orders. This helps you to identify documents used to process orders.

MOVED, REMOVED, OR REPLACED?	WHY?
Moved	Blanket Order Archiving and Document Line Tracking features are no longer specific to Austria, so we have made them generally available in the standard product.

See Also

[Upgrading to Business Central](#)
[Upgrading the Application Code](#)
[Deprecated Fields, and Fields Marked as Obsolete](#)
[Austria Local Functionality](#)

Deprecated Features in the Belgian Version of Microsoft Dynamics 365 Business Central

5/28/2019 • 2 minutes to read

This topic lists and describes the local functionality for Belgium that has been removed from Dynamics 365 Business Central, made available from a new page or report, or replaced by a new feature.

Isabel integration

You can integrate with Isabel to make it easy to upload and download bank files.

MOVED, REMOVED, OR REPLACED?	WHY?
Removed	This feature uses an automation that is not supported by the Dynamics 365 Business Central compiler and blocks the C/AL to AL conversion. In an earlier release we made this functionality unavailable, and now we have removed it. Microsoft partners can provide their solutions for customers who need to integrate with Isabel.

See Also

- [Upgrading to Microsoft Dynamics 365 Business Central](#)
- [Upgrading the Application Code](#)
- [Deprecated Fields, and Fields Marked as Obsolete](#)
- [Belgium Local Functionality](#)

Deprecated Features in the Canadian Version of Dynamics 365 Business Central

6/4/2019 • 2 minutes to read

This topic lists and describes the local functionality for Canada that has been removed from Business Central, made available from a new page or report, or replaced by a new feature.

Customer Statement Report

Shows a list of financial transactions for a selected customer statements for a given period of time. For example, use the report as part of your payment collection process.

MOVED, REMOVED, OR REPLACED?	WHY?	WHEN?
Replaced	The Standard Statement report (report 1316) provides the same capabilities as the Customer Statement report (report 10072), plus the ability to customize layouts in Word, and it is available for all countries.	2019 release wave 2

See Also

[Upgrading Business Central](#)

[Upgrading the Application Code](#)

[Deprecated Fields, and Fields Marked as Obsolete](#)

[Canadian Local Functionality in Business Central](#)

Deprecated Features in the Dutch Version of Dynamics 365 Business Central

5/28/2019 • 2 minutes to read

This topic lists and describes the local functionality for the Netherlands that has been removed from Business Central, made available from a new page or report, made available to one or more additional countries, or replaced by a new feature.

Checking Totals for Purchase Invoices and Purchase Credit Memos

If the total amount on a purchase document does not match the total amount from the purchase lines, you can find out why by letting Business Central calculate the total amount, total base amount, total VAT amount, and total amount including VAT for the purchase lines. The totals display in fields at the bottom of the **Purchase Invoice** or **Purchase Credit Memo** pages.

By default, Business Central does not show these totals. To display them, on the **Purchases & Payables Setup** page, choose the **Show Totals on Purch. Inv./CM.** check box.

NOTE

To use this feature, your purchase invoices or purchase credit memos must have at least one purchase line, and a quantity. Additionally, when you turn on this feature Business Central recalculates totals on all purchase invoices and credit memos. Depending on the number of documents, this can take some time.

MOVED, REMOVED, OR REPLACED?	WHY?
Moved	The feature to check totals for purchase invoices and credit memos is no longer specific to the Netherlands, so we have made it generally available in the standard product.

Standard Sales and Purchase Codes

If you often need to create sales and purchase lines with similar information, you can set up standard codes representing sales and purchase lines that you can then insert on recurring sales and purchase documents, for example, for recurring replenishment orders.

MOVED, REMOVED, OR REPLACED?	WHY?
Removed	The feature has been removed from the Dutch version because it is generally available in the standard product.

See Also

[Upgrading to Dynamics 365 Business Central](#)

[Upgrading the Application Code](#)

[Deprecated Fields, and Fields Marked as Obsolete](#)

[Netherlands Local Functionality](#)

Deprecated Features in the Finnish Version of Dynamics 365 Business Central

5/28/2019 • 2 minutes to read

This topic lists and describes the local functionality for Finland that has been removed from Business Central, made available from a new page or report, or replaced by a new feature.

Multiple Interest Rates

When you create finance charge terms and reminder terms, for delayed payment penalty, you can specify multiple interest rates so that the penalty fee is calculated from different interest rates in different periods.

MOVED, REMOVED, OR REPLACED?	WHY?
Moved	The Multiple Interest Rates feature is no longer specific to Finland, so we have made it generally available in the standard product.

See Also

[Upgrading to Business Central](#)

[Upgrading the Application Code](#)

[Deprecated Fields, and Fields Marked as Obsolete](#)

[Finland Local Functionality](#)

Deprecated Features in the German Version of Dynamics 365 Business Central

6/25/2019 • 2 minutes to read

This topic lists and describes the local functionality for Germany that has been removed from Business Central, made available from a new page or report, or replaced by a new feature.

Copy Existing Items to Create New Items

When you add a new item, to save time, you can use the **Copy Item** function to copy an existing item to use as a template for a new item.

MOVED, REMOVED, OR REPLACED?	WHY?
Moved	The Copy Item feature is no longer specific to Germany, so we have made it generally available in the standard product.

Physical Inventory Order

You can take inventory of your items by using the **Physical Inventory Order** and **Physical Inventory Recording** pages. The physical inventory order contains data for planning, realizing, and analyzing physical inventory. The physical inventory recording contains the items and quantities to be counted and forms the basis of the print-out to be used in the warehouse.

MOVED, REMOVED, OR REPLACED?	WHY?
Moved	The Physical Inventory Order feature is no longer specific to Germany, so we have made it generally available in the standard product.

Blanket Order Archiving and Document Line Tracking

You can archive and delete blanket sales and purchase orders. You can view documents that are related to sales order lines and purchase order lines, including from archived order lines. Related documents that you can track include quotes, shipments, receipts, and blanket orders. This helps you to identify documents used to process orders.

MOVED, REMOVED, OR REPLACED?	WHY?
Moved	Blanket Order Archiving and Document Line Tracking features are no longer specific to Germany, so we have made them generally available in the standard product.

See Also

[Upgrading to Business Central](#)
[Upgrading the Application Code](#)
[Deprecated Fields, and Fields Marked as Obsolete](#)
[Germany Local Functionality](#)

Deprecated Features in the Icelandic Version of Dynamics 365 Business Central

5/28/2019 • 2 minutes to read

This topic lists and describes the local functionality for Iceland that has been removed from Business Central, made available from a new page or report, or replaced by a new feature.

Icelandic Tax Regulations of Conditional Discounts

The local tax regulation of conditional discounts feature enables you to issue a credit memo if a conditional discount is given to a customer. The payment for a conditional discount must be made within a specified period.

MOVED, REMOVED, OR REPLACED?	WHY?
Moved	This feature is no longer specific to Iceland, so we have made it generally available in the standard product. There is a field for number series for credit invoices on the Sales and Receivables Setup form, and a field on the Customer Ledger Entry table to link the appropriate entries to a credit invoice.

See Also

[Upgrading to Business Central](#)

[Upgrading the Application Code](#)

[Deprecated Fields, and Fields Marked as Obsolete](#)

[Iceland Local Functionality](#)

Deprecated Features in the Italian Version of Dynamics 365 Business Central

6/25/2019 • 2 minutes to read

This topic lists and describes the local functionality for Italy that has been removed from Business Central, made available from a new page or report, or replaced by a new feature.

Report Trade with Customers and Vendors in Blacklist Countries/Regions

You must submit a periodic report of transactions with customers and vendors in certain countries/regions that the Italian government has identified in a blacklist.

MOVED, REMOVED, OR REPLACED?	WHY?
Removed	The functionality for blacklisted countries/regions has been removed from the Italian version.

Multiple Interest Rates

When you create finance charge terms and reminder terms, for delayed payment penalty, you can specify multiple interest rates so that the penalty fee is calculated from different interest rates in different periods.

MOVED, REMOVED, OR REPLACED?	WHY?
Moved	The Multiple Interest Rates feature is no longer specific to Italy, so we have made it generally available in the standard product.

See Also

[Upgrading to Business Central](#)

[Upgrading the Application Code](#)

[Deprecated Fields, and Fields Marked as Obsolete](#)

[Italy Local Functionality](#)

Deprecated Features in the Mexican Version of Microsoft Dynamics 365 Business Central

6/4/2019 • 2 minutes to read

This topic lists and describes the local functionality for the Mexico that has been removed from Business Central, made available from a new page or report, or replaced by a new feature.

Customer Statement Report

Shows a list of financial transactions for a selected customer statements for a given period of time. For example, use the report as part of your payment collection process.

MOVED, REMOVED, OR REPLACED?	WHY?	WHEN?
Replaced	The Standard Statement report (report 1316) provides the same capabilities as the Customer Statement report (report 10072), plus the ability to customize layouts in Word, and it is available for all countries.	2019 release wave 2

See Also

[Upgrading Business Central](#)

[Upgrading the Application Code](#)

[Deprecated Fields, and Fields Marked as Obsolete](#)

[Mexican Local Functionality in Business Central](#)

Deprecated Features in the Norwegian Version of Dynamics 365 Business Central

5/28/2019 • 2 minutes to read

This topic lists and describes the local functionality for Norway that has been removed from Business Central, made available from a new page or report, or replaced by a new feature.

Multiple Interest Rates

When you create finance charge terms and reminder terms, for delayed payment penalty, you can specify multiple interest rates so that the penalty fee is calculated from different interest rates in different periods.

MOVED, REMOVED, OR REPLACED?	WHY?
Moved	The Multiple Interest Rates feature is no longer specific to Norway, so we have made it generally available in the standard product.

Paper Sources and Tray Numbers

When printing Norwegian sales documents, you can set up different tray numbers and paper sources on the first, last, and other pages.

MOVED, REMOVED, OR REPLACED?	WHY?
Removed	The feature is not used.

See Also

[Upgrading to Business Central](#)

[Upgrading the Application Code](#)

[Deprecated Fields, and Fields Marked as Obsolete](#)

[Norway Local Functionality](#)

Deprecated Features in the Swedish Version of Dynamics 365 Business Central

5/28/2019 • 2 minutes to read

This topic lists and describes the local functionality for Sweden that has been removed from Business Central, made available from a new page or report, or replaced by a new feature.

Multiple Interest Rates

When you create finance charge terms and reminder terms, for delayed payment penalty, you can specify multiple interest rates so that the penalty fee is calculated from different interest rates in different periods.

MOVED, REMOVED, OR REPLACED?	WHY?
Moved	The Multiple Interest Rates feature is no longer specific to Sweden, so we have made it generally available in the standard product.

See Also

[Upgrading to Business Central](#)

[Upgrading the Application Code](#)

[Deprecated Fields, and Fields Marked as Obsolete](#)

[Sweden Local Functionality](#)

Deprecated Features in the Swiss Version of Dynamics 365 Business Central

6/25/2019 • 2 minutes to read

This topic lists and describes the local functionality for Switzerland that has been removed from Business Central, made available from a new page or report, or replaced by a new feature.

Copy Existing Items to Create New Items

When you add a new item, to save time, you can use the **Copy Item** function to copy an existing item to use as a template for a new item.

MOVED, REMOVED, OR REPLACED?	WHY?
Moved	The Copy Item feature is no longer specific to Switzerland, so we have made it generally available in the standard product.

Physical Inventory Order

You can take inventory of your items by using the **Physical Inventory Order** and **Physical Inventory Recording** pages. The physical inventory order contains data for planning, realizing, and analyzing physical inventory. The physical inventory recording contains the items and quantities to be counted and forms the basis of the print-out to be used in the warehouse.

MOVED, REMOVED, OR REPLACED?	WHY?
Moved	The Physical Inventory Order feature is no longer specific to Switzerland, so we have made it generally available in the standard product.

Blanket Order Archiving and Document Line Tracking

You can archive and delete blanket sales and purchase orders. You can view documents that are related to sales order lines and purchase order lines, including from archived order lines. Related documents that you can track include quotes, shipments, receipts, and blanket orders. This helps you to identify documents used to process orders.

MOVED, REMOVED, OR REPLACED?	WHY?
Moved	Blanket Order Archiving and Document Line Tracking features are no longer specific to Switzerland, so we have made them generally available in the standard product.

See Also

[Upgrading to Business Central](#)
[Upgrading the Application Code](#)
[Deprecated Fields, and Fields Marked as Obsolete](#)
[Switzerland Local Functionality](#)

Deprecated Features in the UK Version of Dynamics 365 Business Central

5/28/2019 • 4 minutes to read

This topic lists and describes the local functionality for the United Kingdom that has been removed from Business Central, made available from a new page or report, or replaced by a new feature.

Accounting periods and system calendar

If your fiscal year is different than the calendar, you can measure your fiscal period in other units of time, such as months or quarters. To do this, you set up system calendars and accounting periods.

MOVED, REMOVED, OR REPLACED?	WHY?
Removed	Lack of use. Additionally, there are standard features for accounting periods that provide most of the same functionality as the UK accounting periods.

Create and export a Bankers' Automated Clearing Service file

You can use Bankers' Automated Clearing Service (BACS) to process financial transactions electronically. To do so, you must export vendor payments to a BACS file using the Export BACS option.

MOVED, REMOVED, OR REPLACED?	WHY?
Removed	This banking format standard is no longer used. This functionality is now covered by extensions such as the Envestnet Yodlee Bank Feeds, AMC Banking, and various other formats.

Non-invoiced stock reports

For month-end reconciliation and auditing, you can use the **Stock Received Not Invoiced** report to view stock that is received but not yet invoiced, and the **Stock Shipped Not Invoiced** report to see stock that has been shipped but not yet invoiced.

MOVED, REMOVED, OR REPLACED?	WHY?
Moved	The functionality for the Shipped, Non-Invoiced Sales Orders and the Received, Not Invoiced Purchase Order reports are no longer specific to the UK, so we have made them generally available as views for sales orders and purchase orders. The views are available in the Navigation Pane as Shipped Not Invoiced and Shipped Not Received options under Sales Orders and Purchase Orders, respectively.

Print Unposted Sales and Unposted Purchase reports

The Unposted Sales and Unposted Purchase reports let you print a list of sales and purchase documents that are not yet posted.

MOVED, REMOVED, OR REPLACED?	WHY?
Moved	The Unposted Sales and Unposted Purchase reports are now available from the Navigation Pane as views under Sales Orders and Purchase Orders.

Other VAT reports

You can use the following reports for VAT reporting:

- **Day Book VAT Entry** - Displays the daily total for VAT entries for a specific period.
- **Day Book Cust. Ledger Entry** - Displays the daily total for customer ledger entries for a specific period.
- **Day Book Vendor Ledger Entry** - Displays the daily total for vendor ledger entries for a specific period.

MOVED, REMOVED, OR REPLACED?	WHY?
Moved	These VAT-related reports are no longer specific to the UK, so we have made them generally available in the standard product.

Specify the supply type on documents

You can specify supply types such as sales, loan, exchange, hire, lease, rental, sales on commission, on tax invoices. To do this, you must update the codes and names of the supply types in the **Types of Supply** window.

MOVED, REMOVED, OR REPLACED?	WHY?
Removed	Lack of use. The business need that this functionality was introduced to cover is no longer relevant.

Multiple Interest Rates

When you create finance charge terms and reminder terms, for delayed payment penalty, you can specify multiple interest rates so that the penalty fee is calculated from different interest rates in different periods.

MOVED, REMOVED, OR REPLACED?	WHY?
Moved	The Multiple Interest Rates feature is no longer specific to the UK, so we have made it generally available in the standard product.

Objects and Fields that are deleted in Dynamics 365 Business Central

Table 10505 has been deleted. The following list shows additional fields that are deleted as a result of the features that have been removed.

TABLE ID	FIELD ID	COMMENTS
23	10550	Deleted.
81	10550	Deleted.
81	10551	Deleted.

TABLE ID	FIELD ID	COMMENTS
81	10552	Deleted.
81	10553	Deleted.
271	10550	Deleted.
312	10550	Deleted.
312	10551	Deleted.
334	10505	Deleted.
363	10550	Deleted.
7118	10505	Deleted.
7152	10550	Deleted.

See Also

[Upgrading to Business Central](#)

[Upgrading the Application Code](#)

[Deprecated Fields, and Fields Marked as Obsolete](#)

[United Kingdom Local Functionality](#)

Deprecated Features in the United States Version of Dynamics 365 Business Central

6/4/2019 • 2 minutes to read

This topic lists and describes the local functionality for the United States that has been removed from Business Central, made available from a new page or report, or replaced by a new feature.

Customer Statement Report

Shows a list of financial transactions for a selected customer statements for a given period of time. For example, use the report as part of your payment collection process.

MOVED, REMOVED, OR REPLACED?	WHY?	WHEN?
Replaced	The Standard Statement report (report 1316) provides the same capabilities as the Customer Statement report (report 10072), plus the ability to customize layouts in Word, and it is available for all countries.	2019 release wave 2

See Also

[Upgrading Business Central](#)

[Upgrading the Application Code](#)

[Deprecated Fields, and Fields Marked as Obsolete](#)

[Norway Local Functionality in Business Central](#)

Migrate Legacy Help to the Dynamics 365 Business Central Format

3/31/2019 • 4 minutes to read

The standard version of Business Central follows a [user assistance model](#) with tooltips to explain all fields and actions and conceptual descriptions of functionality that is published to a public website. If you are building an app, you are expected to comply with this model. However, there are many ways in which you can migrate and reuse any existing Help that you might have.

Reusing existing web content

When you move to Business Central, you can reuse your existing product Help solution in most situations, especially if the content is already published to an internal or external website. In that case, all you have to do is to add that website to the configuration of Business Central online or on-premises. For more information, see [Configuring the Help Experience](#).

More specifically, if you have content that you created for Dynamics NAV then you can choose to reuse that for your Business Central solution.

For example, you have a Dynamics NAV Help Server website with HTML files that describe your solution according to the Microsoft Dynamics NAV 2013 R2 documentation model and format. In that scenario, you can reuse the Help Server website and rebrand that and the content accordingly. You then connect your Business Central solution with that Help Server website. For more information, see [Configuring the Help Experience](#).

NOTE

For apps for Business Central online, you must apply tooltips to controls and actions in both page objects and page extensions, and you must supply context-sensitive links. For more information, see [Configuring the Help Experience](#).

Converting existing content

If your existing content is in a different format, such as PDF files, Word documents, or printed manuals, you must decide if you want to keep the content as-is, or if you want to convert it to a format that can be accessed from inside Business Central. There are third party tools available that can help you migrate to other formats, depending on the the current format and the target format.

If you are migrating your solution from Microsoft Dynamics NAV 2013 R2 or later versions of Dynamics NAV then you most likely have been using the Dynamics NAV Help Server, and your Help content is in HTML format. That means that you can reuse your existing content as-is, or you can use publicly available third-party solutions to convert some or all of your HTML files to Markdown, if you want to follow similar processes to the ones the Microsoft team follows. For more information, see the [Moving to Markdown](#) section.

If you are migrating from an earlier version of Dynamics NAV then you can choose to first migrate to the Microsoft Dynamics NAV 2013 R2 format, and then migrate again to Markdown or similar formats. For more information, see [Upgrading Your Existing Help Content](#) in the legacy docs for Microsoft Dynamics NAV 2013 R2.

If you are migrating your solution from Dynamics GP, you might have content in PDF files. In that case, you can choose to convert the content to Markdown as described in the [Moving to Markdown](#) section, and then publish to a new online library on your current website, for example.

Converting legacy Dynamics NAV field Help to tooltips

For the default version of Business Central, Microsoft extracted the first paragraph from the HTML files of the Dynamics NAVHelp for table fields, and then imported the text into the page objects of the base application as tooltips. You can build a similar tool if you want to reuse your existing content in the same way.

The tooltips play an important role as part of the Business Central [user assistance model](#), and we encourage you to apply tooltips to your controls and actions as well.

Moving to Markdown

The Microsoft team converted a subset of the legacy Help for Dynamics NAV to build the new Help library at <https://docs.microsoft.com/dynamics365/business-central/>. If you want to customize or extend the Microsoft Help, you can fork our public repo for either the source repo in English (US) at <https://github.com/MicrosoftDocs/dynamics365smb-docs>, or one of the related repos with translations into the supported languages. The readme.md file in the source repo provides tips and tricks for working with the Microsoft GitHub repos and Markdown.

Converting your existing content to Markdown can be done using third-party tools, including but not limited to [PanDoc](#) or the [Writage](#) plugin for Word.

When you have converted your content to Markdown, you can use a Git repo in Azure DevOps as your source repository, create a private or public repo in GitHub, or set up a project in [MkDocs](#), for example. Then you can use open source tools such as [DocFx](#) to generate content for your website. In general, working in Markdown means that you have access to a world of open source tools and do not have a hard dependency on Microsoft providing you with tools.

If you do not yet have a website that you publish content to, then there are several ways in which you can create such a site. The [MkDocs](#) project generates a website for you, but you can also work with a web designer to build a site that resembles the [Docs.microsoft.com](https://docs.microsoft.com) site, if that is what your customers will prefer.

See Also

[Configuring the Help Experience](#)

[User Assistance Model](#)

[Development of a Localization Solution](#)

[System Requirements](#)