

Microsoft Dynamics 365 Business Central Extension-Entwicklung mit Visual Studio Code



V18.00.01

HINWEIS

Diese Präsentation dient nur zu Informationszwecken. Bei Verwendung der Präsentation zu anderen Zwecken lehnt get&use Academy GmbH jede Gewährleistung wegen Sach- und Rechtsmängeln ab. get&use Academy GmbH lehnt jede Haftung für direkte und indirekte Schäden - sei es aus Vertrag oder aus Gesetz - ab, die in Verbindung mit der Anwendung und sonstiger Nutzung der Präsentation entstehen können. Diese Präsentation kann bei Bedarf und ohne vorherige Ankündigung von get&use Academy GmbH geändert werden. Der Inhalt dieser Präsentation ist urheberrechtlich geschützt. Ohne schriftliche Erlaubnis von get&use Academy GmbH darf die Präsentation weder ganz noch teilweise in irgendeiner Form vervielfältigt oder bereitgestellt werden. Die beschriebenen Programme dürfen nur gemäß den Lizenzbedingungen angewendet oder kopiert werden.

COPYRIGHTVERMERK

Copyright © 2024 get&use Academy GmbH,
Deutschherrnstr. 15-19, 90429 Nürnberg, Germany.
Alle Rechte vorbehalten.

WARENZEICHEN

Die Warenzeichen, auf die in dieser Präsentation Bezug genommen wird, sind Warenzeichen oder eingetragene Warenzeichen der jeweiligen Eigentümer bzw. Hersteller. Alle nicht ausdrücklich gewährten Rechte sind vorbehalten.

Introduction

Agenda

- Overview of Extensions
- Development Components
- Skills needed

Customizations vs Extensions

- Customization:
Modification of base tables, code and other objects.
- Extensions:
Addition of well-defined modules that are invoked alongside the base application.

Extensions Framework – Roadmap

NAV 2016

NAV 2017

NAV 2018 and on

Objects

- Pages (new and modified)
- Tables (new and modified)
- MenuSuites
- Codeunits

Data

- Permission sets (per tenant)

Developing

- Upgrading capabilities

Objects

- Reports
- XML ports
- Queries

NET Add-ins

- Server-side .NET interop add-ins
- Client-side JavaScript extensibility control add-ins
- Client-side WinForms extensibility control add-ins

Data

- Web services (per tenant)
- Table data (new tables added for extension)
- Custom report layouts

Translations

- Language files

Developing

- Improved upgrade APIs
- Debugging and code coverage

Platform Improvements

- Improvements to page and table extensions

Developing

- Improved developer tooling
- Language enhancements

Apps

- Improved support for large extensions applications

Development Components

Customizations

- C/SIDE directly connected to SQL Server Database
- Optionally PowerShell for synchronizing and upgrading scenarios

Extensions v1

- Development in C/SIDE, similar to customizations
- Based on delta files with limited features
- PowerShell for creating, publishing and installing navx package

Extensions v2

- Development in Visual Studio Code
- File based, more features compared to Extensions v1
- Deployment can be done with VS Code
- Optionally PowerShell for managing and installing app package

Skillset

- Knowledge and using of event based architecture.
- Apply design patterns.
- Know how to make use of Wizards, Assisted Setup, Notifications.
- Knowledge of AL code.
- Knowledge of VS Code.
- Source code management with Git.

Docker

Session objectives

- Get a basic understanding of Docker.
- See how Docker helps with running Business Central.
- Understand the flexibility of Business Central on Docker.
- Learn about Microsoft plans on Business Central on Docker.

Agenda

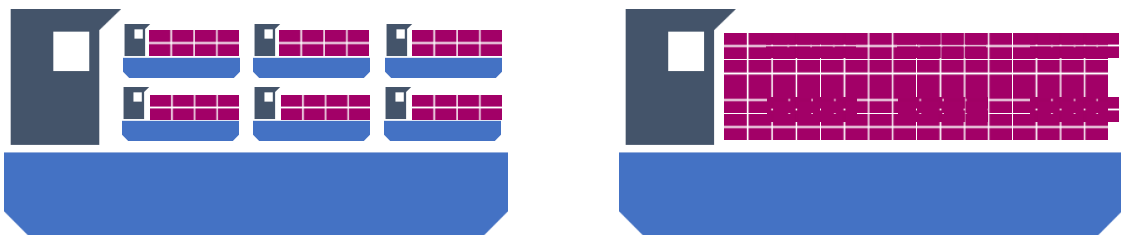
- Introduction to Docker / Windows Containers.
- Get up and running.
- Architecture of Business Central on Docker.
- Extend the standard Business Central image.
- Build and reuse your own images.
- Resource governance.
- Running a multi-CU environment.
- What is Microsoft shipping.

Introduction to Docker / Windows Containers

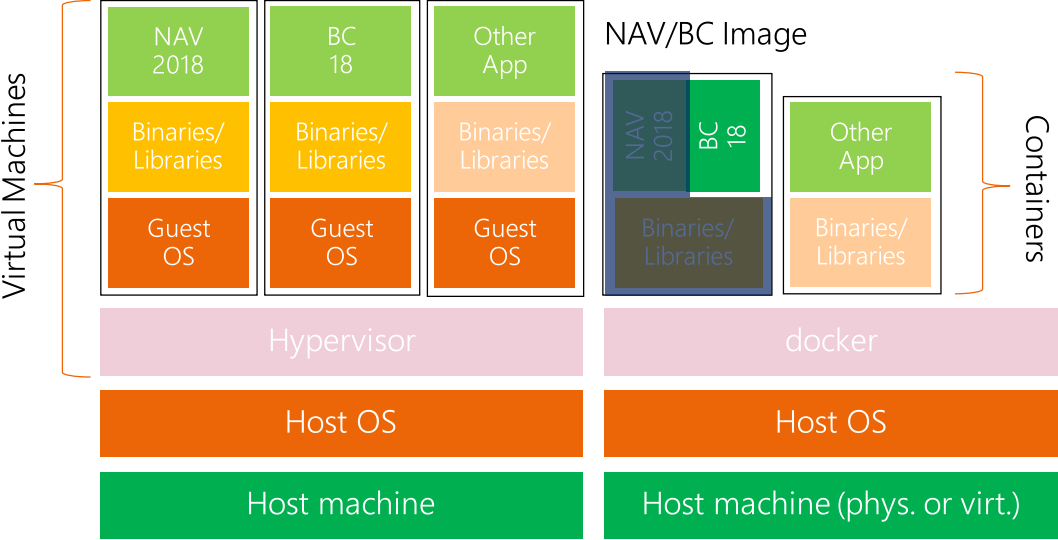
What is Docker?

- Leading cross platform software container environment.
- What is a Docker container and a Docker image?
 - An image is a template with the minimum amount of OS, libraries and application binaries needed (no GUI!) .
 - A container is an instance of an image with an immutable base and its changes on top.
 - Business Central / NAV in a container is a ready-to-run, configurable Service Tier with optional built-in SQL and IIS.
- Key features of Docker.
 - Greatly reduce "work on my machine"/ "works here" problems.
 - Run and manage containers side-by-side but isolated on shared resources (CPU, RAM, OS, kernel).
 - Easily create "locked down" configurations, but allow extending / adjusting it.
 - Provide resource governance for applications that natively don't have that.

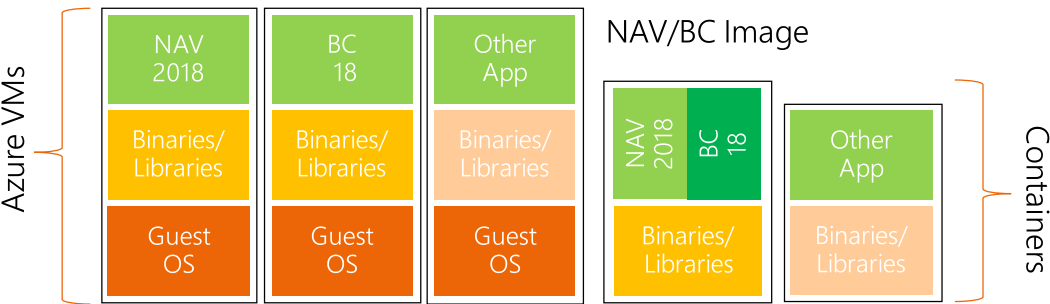
Virtual Machines vs. Containers



Virtual Machines vs. Containers

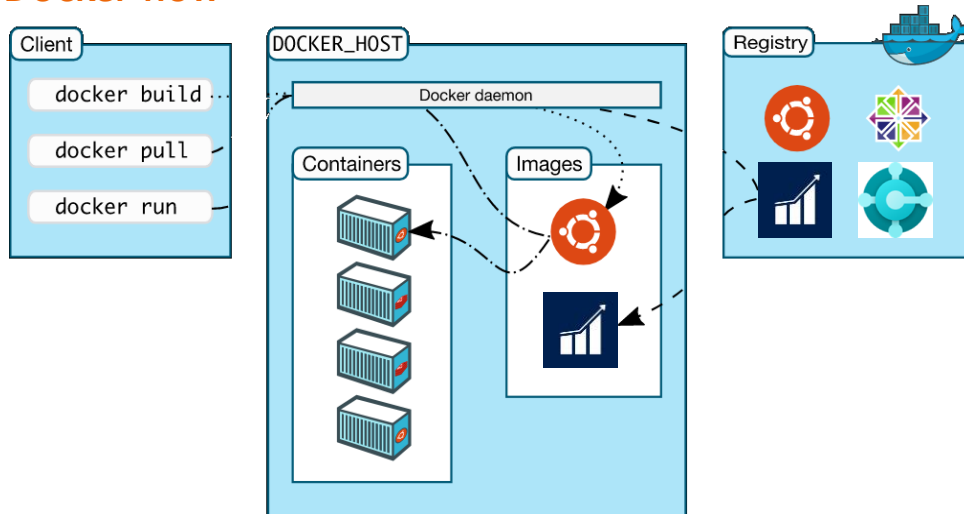


Azure VMs vs. Azure Containers*



*Azure Container Service for Windows is in private preview

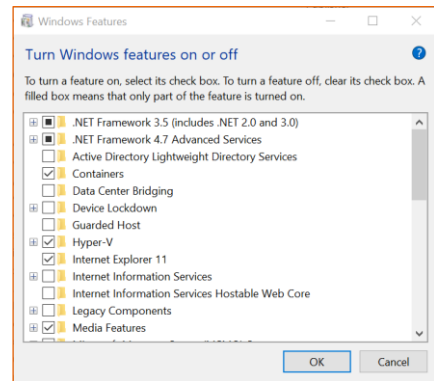
Docker flow



Get up and running

OS and prerequisites

- Windows Server 2016 and 2019
 - Activate feature: Containers
 - Business Central Images for Server 2019 are smaller than for Server 2016
- Windows 10
 - Professional / Enterprise edition
 - Active features: Hyper-V, Containers
- Windows Server 1709
 - Activate features: Hyper-V, Containers



Install Docker

- Install with:
Install-Module DockerProvider -Force
Install-Package Docker -ProviderName DockerProvider -Force
- More info: <https://docs.docker.com/install/windows/docker-ee/>

Install PowerShell module BcContainerHelper

- Written by Freddy Kristiansen.
- PowerShell module for easier management of NAV and Business Central containers.
- Install with:
install-module BcContainerHelper -force
- If you already have BcContainerHelper installed, update to latest version with:
update-module BcContainerHelper -force
- More info: <https://freddysblog.com/2020/08/11/bccontainerhelper>

OS impact on Docker editions

- Windows Server
 - Docker EE (licensed within Windows Server license)
 - Process Isolation (by default)
 - Hyper-V Isolation (can be used when needed)
- Windows 10
 - Docker CE
 - Hyper-V Isolation only until 1804
 - Hyper-V isolation optionally available from 1804
 - Docker EE not by default but see <https://www.kauffmann.nl/2019/03/04/how-to-install-docker-on-windows-10-without-hyper-v/>
- Source: <https://learn.microsoft.com/en-us/windows-server/get-started/editions-comparison-windows-server-2016?tabs=full-comparison>

Basic docker image commands

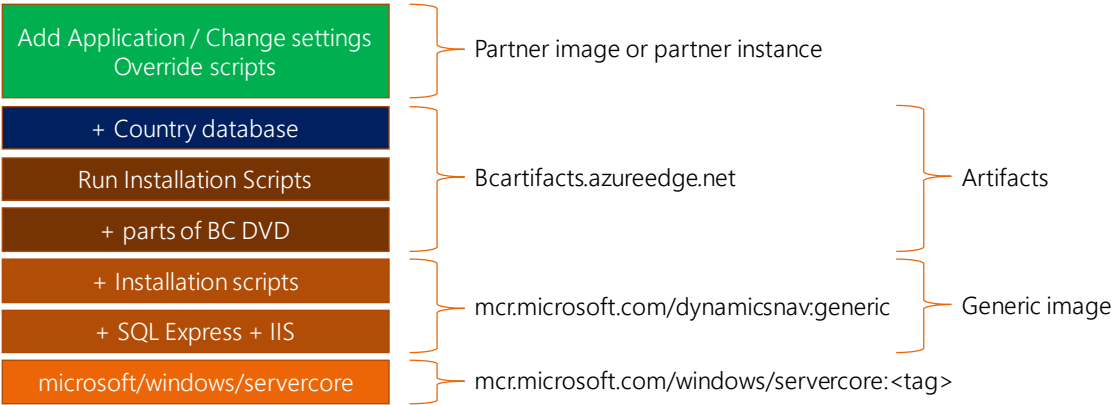
- List existing images: `docker images`
- Download image: `docker pull <repository>/<product>[:tag]`
- Remove image: `docker rmi <image-id>`

Basic docker container commands

- List running containers: `docker ps`
- List all containers: `docker ps -a`
- Run new container: `docker run <image>`
- Start existing container: `docker start <container>`
- Stop running container: `docker stop <container>`
- Remove container: `docker rm <container>`

Architecture of Business Central on Docker

Business Central Container Image Architecture (from 07/20)



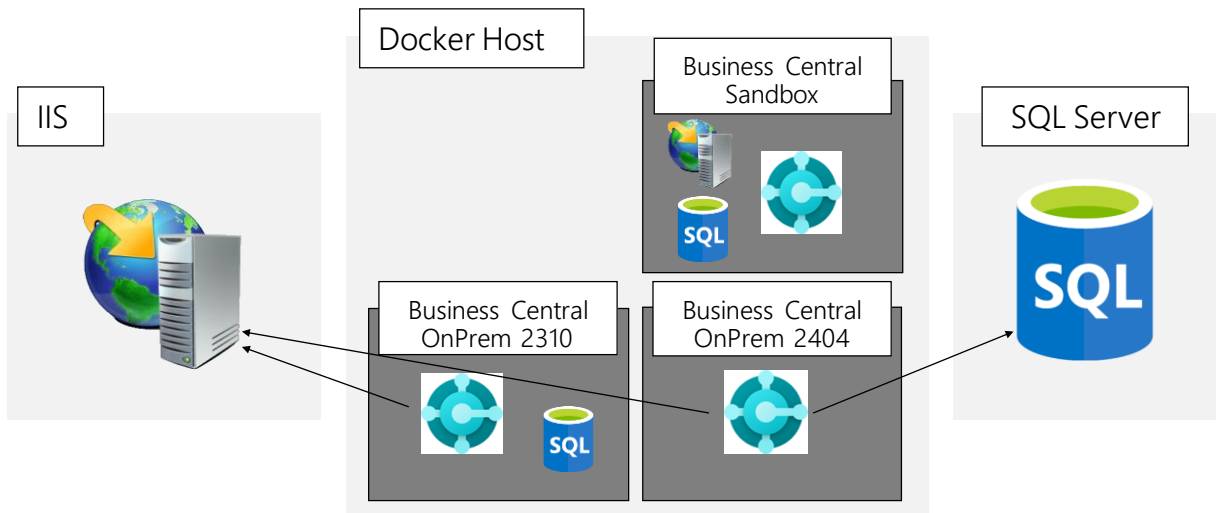
Docker containers and data

- Layering in Docker containers:
 - Image layers (read only) + one container layer (read/write).
 - All write operations are being made with the top layer (the container layer).
 - When you modify an existing file, Docker checks if it exists in the container layer:
 - Yes → modify this file.
 - No → drill down through all layers from the top one until the first version of the file is reached. Copy this file into the container layer and use it.
 - Volumes:
 - Map a host folder to a folder inside the container which leads to the same performance as directly on the host and results in persistency even if the container is removed (and the container layer with it).

Database handling – basic scenarios

- Database in container without volume.
Is the database tightly coupled with exclusively one Business Central server instance inside the container and can easily be recreated e.g. in a demo / sales environment and doesn't rely too much on performance.
- Database in container with a volume.
Same as 1) but you need better performance and the database needs to survive a recreation of the container, e.g. for a CU update.
- Database in central SQL server.
Multiple Business Central server instances connected to the database or you need to fully tweak performance.

Running the Business Central Container Image



What happens when running a specific image

- Start SQL Server (if necessary).
- Start IIS (if necessary).
- Create Certificate (if necessary).
- Reconfigure Business Central Service Tier.
- Start Business Central Service Tier.
- Setup Web Client (if necessary).
- Setup File Share (if necessary).
- Setup Users (if necessary).

Extend the standard Business Central docker images

Scenarios

- Use your own license file.
- Use your own database.
- Use your own domain name and SSL Certificate.
- Publish ports on the host for public access.
- Add your control add-in.s
- Use ClickOnce for the Windows Client.
- Setup additional users.
- Modify customsettings.config.
- Modify web.config.
- And many many more...

Build and reuse your own images

Build and reuse your own images – Requirement

- After extending the standard Business Central Docker images, you need to persist that and reuse your images.
- Partner perspective: Changes need to be reliably delivered to customers.
- Customer/hoster perspective: Your additional changes need to be persisted and reused.
- All: Internal / dev / qa / test needs to be as close to production as possible.

Build and reuse your own images – Solution

Docker commit "saves changes" by creating a new image based on a changed Container. Docker tags allow you to identify different "versions".

Docker registries allow you to store and distribute custom images.

- Step 1: Make the changes.
- Step 2: Commit your image with your own tag.
- Step 3: Pull and run the image.

Resource Governance

Resource Governance – Requirement

- If a Business Central Server instance goes crazy, there is no way to stop it from bringing the whole machine down.
- We don't have tooling to limit resource consumption.
- Partner/development perspective: errors in development might block your whole dev / QA team.
 - If you work with a centralized dev environment.
- Customer/hoster perspective: a problem in one Business Central instance can block all other instances.
 - You can work with one instance per machine but this is a waste of resources.

Resource Governance – Solution

- Use resource limits for Containers.
- There are limits for RAM and CPU usage.
 - You have to add them when starting the Container and won't be able to change them while the Container is running.
 - On Linux you can also configure if and how Swap is used, should come to Windows soon.
- If the main process in a Container hits an OutOfMemoryExceptions, it stops and restarts the Container (if configured that way).
 - But users on that instance will lose their session.
 - If you run in "Swarm mode" Docker will automatically make sure multiple instances are always running.

Running a Multi-CU Environment

Running a Multi-CU Environment – Requirement (1)

You can't run multiple Business Central Cumulative Updates (CUs) for the same release well on one machine.

- Microsoft delivers monthly CUs, which makes this a permanent issue.
- Partner perspective: Customers running on different CUs need support on different Cus.
 - Problems need to be reproduced on the same CU / custom dll version as the customer has.
 - Fixes need to be delivered using the same CU as the customer has.
 - Fixes by Microsoft need to be tested (are bugs fixed and no new bugs or different behaviour introduced).
 - Sometimes compatibility between CUs breaks.

Running a Multi-CU Environment – Requirement (2)

- Customer/hoster perspective: Fixes / enhancements include or depend on new CUs.
 - Update test / staging and later production environments with new CUs.
 - Sometimes you might want to go back.
 - Parallel business tests might collide with the technical CU tests.
- Different CUs use the same files, links etc., just with different versions, so there is no good way to install them in parallel on the same machine.

Running a Multi-CU Environment – Solution

- Multiple Containers with different CUs can safely run in parallel on the same machine.
- A Container instance has its own separate file system.
 - We don't get conflicts for Business Central Server and Web Clients.
 - With ClickOnce we can even deploy separate Windows Clients and Dev Environments.
 - The Container can optionally include its own database or you can connect it to the right one.
- Update is only an easy docker pull of the new image.
 - Going back means just throwing away the new Container and creating or re-starting the old one.

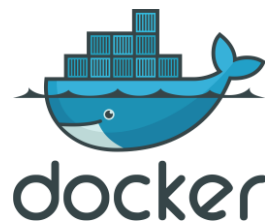
What will Microsoft be shipping

Cumulative Updates

From Autumn 2017:

- From NAV 2016 to current Business Central on-prem
- Supported for test and development
- All on mcr.microsoft.com/businesscentral/onprem

ALSO ON



Azure Images/Templates

From Autumn 2017:

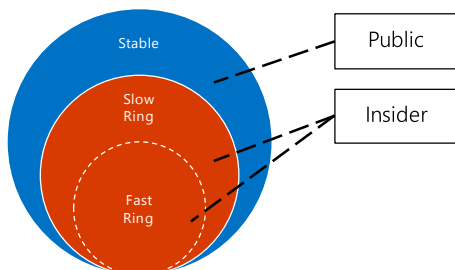
- Azure Demo Environments
- Business Central Sandbox Environments
- All combined into one Azure template using docker



Modern Development Tools

From Autumn 2017:

- Stable: Month minus one
- Slow ring: Monthly (Insider)
- Fast ring: Bi-weekly builds (Insider)
- Preview of individual new components



Business Central and NAV Artifacts

- Use BcContainerHelper command **Get-BCArtifactUrl** to get the download url of artifacts.
- Example:
Download the artifact for the latest W1 sandbox and create a docker image based on the best fitting generic image for my machine.

```
$artifactUrl = Get-BCArtifactUrl -type Sandbox -select Latest  
Download-Artifacts -artifactUrl $artifactUrl -includePlatform  
$useBestContainerOS = $true  
New-BCImage -artifactUrl $artifactUrl -imageName "my-sandbox"
```

Exercise: Create a Docker Container

- Create a docker container for Business Central development.

Description:

- Execute the PS script "gua Docker Container ertsellen.ps1" located in c:\guADockerScripts.
- Enter BC as name of your container.
- Select the latest gua.bc:onprem image.
- If script asking for password, use EiWoha7T@.
- If script asking for authentication, use Windows.
- Test the container by opening the web client with <http://bc/bc>.

Introduction Visual Studio Code

Agenda

- General overview
- Directory and file concept
- Command palette
- Configuration Settings
- Source control integration
- Code snippets

General Overview

- Light-weight, multi-platform source code editor:
 - Windows, macOS, Linux
 - Built-in support: JavaScript, TypeScript, Node.js
- Key strength: extensibility (through built-in API)
- Rich ecosystem of extensions:
 - Languages: C++, C#, Python, PHP, ...
 - Runtimes: .NET, Unity, ...
 - Utilities: debuggers, code analysis, themes, snippets, ...
 - AL Language extension for developing for Dynamics NAV and Dynamics 365 Business Central.

VS Code Anatomy

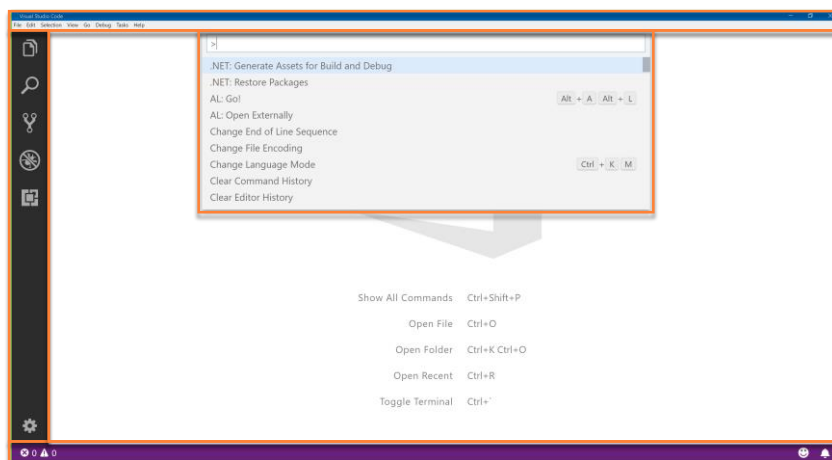
Menu

Command
Palette

Text Editor

Activity bar

Status bar



Directory and File Concept

VS Code operates on individual files or workspaces:

- File
 - Single file opened for editing.
 - File is always in a specific language mode, detected from extension.
 - Language mode can be changed manually.
- Directory (or Folder)
 - Principal way of working with VS Code.
 - Directories are referred to as workspaces.
 - Directory that is opened is referred to as workspace root.
 - A workspace can be mapped to a source control repository.

Command Palette

- Menus contain only the basic functionality.
- There are no toolbars, icons, ribbons.
- Command Palette:
 - Primary means of interaction with VS Code.
 - **Ctrl+Shift+P** or **F1**
 - Quick keyboard-based access to all functionality (basic and extension)
 - **Ctrl+P**
 - File access and CLI access to supported functionality (including extensions)

Configuration Settings

- All configuration (built-in and extensions) is done through configuration files.
- Configuration language is JSON (JavaScript Object Notation).
- Editing configuration:
 - **Ctrl+Shift+P > Preferences**
 - **File > Preferences**

Most commonly used configuration files

Configuration file	Use
User settings (settings.json)	<ul style="list-style-type: none">• Configuration settings that apply to the entire application.• Override any default application settings and behavior.
Workspace (settings.json)	<ul style="list-style-type: none">• Workspace-level (directory) configuration settings that apply to a specific workspace.• Override user settings.
Language snippets ({language}.json)	<ul style="list-style-type: none">• Define user-configurable snippets.• Available per language.• Editor must be in the specified language mode.
Keyboard shortcuts (keybindings.json)	<ul style="list-style-type: none">• Define application-wide keyboard shortcuts.• Override any application defaults.• Override any installed keymaps.

Source Control Integration

- Integrated support for Git.
 - Most common commands.
- Supports all Git repositories and providers:
 - Github
 - Visual Studio Team Services
 - Privately hosted
 - ...

Git feature integration

- Git context and actions are shown in Activity Bar and Status Bar.
- Committing changes.
- Branching and tagging.
- Gutter indicators.
- Comparing changes and merging conflicts.
- Initializing Git repositories.

Code Snippets

- Each language provides default language snippets.
- User can add own snippets, per language.
 - Insert a snippet:
 - **Ctrl+Shift+P** > **Insert Snippet**, Select from list
 - Start typing the snippet prefix
 - Create snippets:
 - **Ctrl+Shift+P** > **Preferences: Configure User Snippets**, Select from list
- Where are snippets stored?
 - The user-defined snippets that you create, are stored here:
%USERPROFILE%\AppData\Roaming\Code\User\snippets
 - The snippets that come from an extension, are stored here:
%USERPROFILE%\vscode\extensions\<extensionname>\snippets
- Documentation to create snippets in VS Code can be found at
<https://code.visualstudio.com/docs/editor/userdefinedsnippets>

Code Snippets – Tools

- snippet-generator.app
 - When you are creating your VSCode snippets, simply paste the text that you want to convert to a snippet to this tool, and you immediately get it converted to a JSON representation for a VSCode snippet.
 - <https://snippet-generator.app/>
- More info: <https://www.waldo.be/2020/01/20/vscode-snippets/>

Developing an Extension with VS Code

Agenda

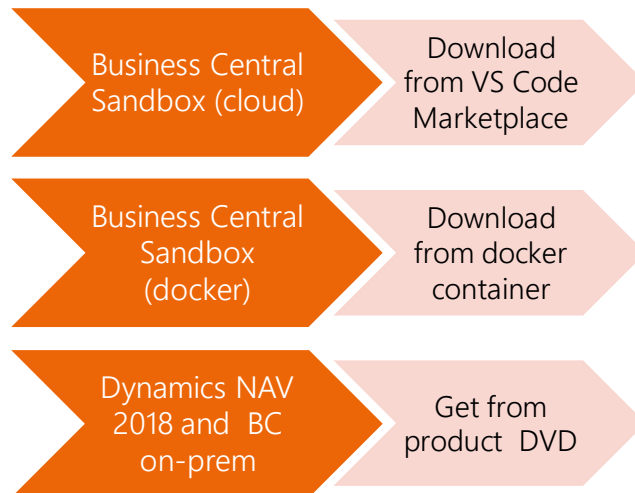
- Getting started with VS Code and AL
- AL Language
- Developing in AL
- Running C/SIDE and AL Side-by-Side
- Special Object Types

Getting started with VS Code and AL

Getting Started

- AL Language extension
- Creating a demo workspace
- Configuring the workspace
- AL Language extension commands

How to get AL Language extension



What's in the AL Language extension

- Creating Business Central objects / Writing AL code
- Syntax highlighting
- Intellisense
- Code formatting
- Compilation errors and warnings
- Deploying to Business Central
- Automating creation of Business Central-ready workspaces
- Code Analyzers
- Debugger
- ...

AL Language extension community

- Community can participate in testing and providing feedback on

<http://github.com/microsoft/al>

Creating a demo workspace

Extension autogenerated the Business Central extension workspace structure.

- To create a demo workspace:
 - Press ***Ctrl+Shift+P***
 - Enter ***AL: Go!***
- The default workspace contains:
 - Configuration files
 - An example AL file

Configuring the workspace

- All VS Code workspaces are configured through .json files. JSON stands for JavaScript Object Notation.
- Configuring any aspect of the VS Code itself is also done through .json files.
- Configuration files support:
 - Tooltips: Explain the meaning of a configuration option.
 - Intellisense: Simplify looking up and setting correct options.
 - General overview of configuration options.

AL workspace configuration files

File	Purpose
.vscode\launch.json	Core VS Code configuration file. VS Code uses this file to control the launch process. AL Language extension.
.vscode\settings.json	Core VS Code configuration file. This is central workspace configuration file. This file overrides user defaults. Less important than launch.json.
app.json	Serves the role of the Business Central extension manifest file.

launch.json

- When placed in .vscode directory, this file is understood and processed by the VS Code.
- Individual extensions can publish their own launch providers (types) that define how the launch process works for the extension.
- AL Language extension uses it to configure connection parameters for Microsoft Dynamics Business Central server.

AL configuration settings in launch.json

Setting	Value
server	The HTTP URL of your server, for example: http://localhost .
serverInstance	The instance name of your server, for example: BC.
port	The port on which the development endpoint is running on the server, default value: 7049.
tenant	The tenant ID in case the server is configured for multitenancy.
authentication	Specifies which authentication method should be used for publishing the extension. Supported methods are UserPassword, Windows and AAD.
startupObjectType	The type of the startup object to launch when you press F5. Currently only objects of type Page are supported. Table or Page.
startupObjectId	The ID of the startup object to launch when you press F5. Currently only objects of type Page are supported.
schemaUpdateMode	Remove data on deployment, or try to keep it. Three values: ForceSync , Recreate , Synchronize (Default).

Important settings in app.json

Setting	Value
name	The name of the app.
publisher	The author of the app.
version	The version of the app, important for upgrade scenarios.
application	The version of the base Business Central application, e.g. 11.0.0.0. Only valid until version 14! Beginning with version 15 the BaseApp is a dependency.
idRanges	List of id ranges. All objects must be in these ranges. The developer preview comes with permissions for 50100 – 50149. Dynamics 365 Sandbox has 50000 – 99999 and 70000000 – 75000000.
dependencies	List of dependencies for the extension package
features	Section to switch on "features". Use "TranslationFile" to generate .xlf file.
showMyCode	To enable viewing the source code when debugging into an extension. Also necessary when using the in-client visual designer.
target	By default this is "Extension". For Business Central, you can set this to "Internal" to get access to otherwise restricted APIs. The Dynamics Business Central Server setting must then also be set to "Internal".
...	

AL Language Extension Commands

Accessed through the Command Palette (Ctrl+Shift+P), prefixed by "AL: "

Command	Description
Clear Credentials Cache	Clear credentials.
Download source code	Download the source code from the in-client designer.
Download Symbols	Downloads object definitions for compiler and Intellisense.
Generate Manifest	Generates the app.json file and populates it with default values.
Go!	Generates an entire workspace structure and files to kick-start extension development.
Package	Builds the .app package.
Publish	Publishes the .app package to Business Central, installs it, and then runs the configured start page.
Publish and open in designer	Open the in-client designer after publishing.
Publish without debugging	Publishes the .app package to Business Central, installs it, the configured start page will NOT run.
Open Externally	Opens the design of a report in the external editor (RDLC, Word).

AL Language

AL Language

- Understanding AL
- AL Syntax
- .al files
- Symbols
- Object types
- Additional features

Understanding AL

- AL is not the same as C/AL.
- AL is very similar to NAV text file format.
- There are important distinctions:
 - AL omits most IDs except object ID and field ID in tables.
 - AL is not sensitive to regional settings.
 - Boolean constants are true/false (as opposed to Yes/No).
 - Naming conflicts are checked at compile time.
 - Certain keywords are changed.
 - AL introduces more object and variable types.
 - No .NET interop access.
 - AL does not use UPPERCASE, but case is never autocorrected.
 - AL is Unicode.
 - There is no Documentation trigger in objects, comments can be freely added in code.
- <https://learn.microsoft.com/de-de/dynamics365/business-central/dev-itpro/developer/devenv-differences>

Understanding AL (continued)

- AL only supports extension development.
- It change and introduce more features, e.g.:
 - Translation files
 - Code analysis
 - Debugger
 - ...

AL Syntax

- AL code syntax is the same as in C/AL.
- All C/AL code will compile under VS Code except unsupported variable types.
- All C/AL keywords are still present in AL.
- The only differences are:
 - C/AL always auto-capitalizes keywords and autocorrects capitalization on function calls and variables.
 - AL never auto-capitalizes or autocorrects anything.
 - AL intellisense suggests PascalCase on all AL built-in functions and keywords.

.al Files

- All AL code must be placed inside .al files.
- .al files may reside anywhere in the workspace folder hierarchy.
- One .al file may contain more object definitions – but it is a good practice to have one file per object!
- The workspace must contain at least one .al file - otherwise, a compile-time error occurs.
- .al files may be empty but must be present.
- AL syntax highlighting works only inside .al files.

Symbols

The AL Language extension “understands” the language itself. It does not “understand” the underlying application model (available objects, properties, methods...).

- AL Language uses symbols to “understand” the application model.
- Symbols
 - provide list of available objects and their members.
 - allow AL Language to remain strong-typed.

To reference an object from AL code, it must be present either in the symbols or in another .al file.

Symbol packages

- Symbols are provided in symbol package files.
- A symbol package file has .app extension.
- Symbol packages are nothing but other extensions.

Note:

Any .app file that can be found on the packages path is loaded as a symbol package, and its objects are available to the AL compiler.

Making symbols available

- To make symbols available to VS Code, you can:
 - Manually copy the symbols into the workspace.
 - Download the symbols for the workspace.
 - Have a central repository of symbols for all workspaces.

Keep in mind – symbols can be shared between workspaces!

Configuring symbols

- Symbols are configured with the "al.packageCachePath" setting in workspace settings.
- Default value is "./.alpackages".
- Regardless of whether this setting is configured in user settings or workspace settings, it is always evaluated against the current workspace path.

Object Types

AL Language supports the following object types:

Object Type	AL Keyword
Table	table
Table Extension (new)	tableextension
Page	page
Page Extension (new)	pageextension
Page Customization (new)	pagecustomization
Profile (new)	profile
Codeunit	codeunit

Object Types (continued)

Object Type	AL Keyword
Report	report
Report Extension	reportextension
Request Page (new)	requestpage
Xmlport	xmlport
Query	query
Control Add-in (new)	controladdin
Dotnet Interop	dotnet
Enum	enum
Enum Extension	enumextension

Object Types (continued)

Object Type	AL Keyword
Permission Set	permissionset
Permission Set Extension	permissionsetextension
Interface	Interface

Extension object types

- In AL, it is impossible to "customize" existing objects.
- You can provide extension objects for:
 - Tables
 - Pages
 - Permissionsets (from April 2020 in BC16)
 - Reports (from April 2021 in BC 18)
- Extension objects are more functional than deltas and provide much more functionality.
- Extension provide much smaller conflict surface than deltas.

Additional features

- Snippets
- REST API
- XML
- JSON
- ControlAddin + UserControl
- Translations
- ...

Note: We'll get to know most of these in much more detail.

Guidelines, Rules and Tools

Using a Prefix or Suffix

In your extension, the name of each application object, must contain a prefix or suffix.

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 additionally be defined at the control/field/action/group level.

Prefix/Suffix 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 [Code Analysis 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.

File naming notation

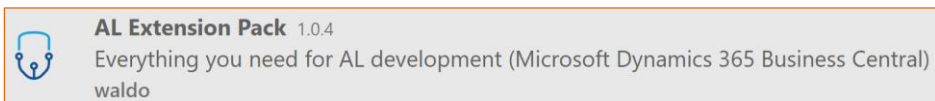
Each file name has object name with only characters [A-Za-z0-9], object type and dot al, for file type.

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.

- Full objects
 - <ObjectNameSuffix>.<FullTypeName>.al
 - <PrefixObjectName>.<FullTypeName>.al
- Extensions
 - <ObjectNameSuffix>.<FullTypeName>Ext.al
 - <PrefixObjectName>.<FullTypeName>Ext.al

Extension Recommendation for VS Code

- Install this extension from marketplace:



- This extension installs a number of other extensions.
- We will use some of these in the course of the course to accelerate the development.

Configure CRS AL Language Extension

- The rename function in this extension supports us in complying with the rules for the use of prefix/suffix and the file name conventions.
- The following settings must be made in settings.json for this:

```
"CRS.FileNamePattern": "<ObjectNameShort>.<ObjectTypeShortPascalCase>.al",  
"CRS.FileNamePatternExtensions": "<ObjectNameShort>.<ObjectTypeShortPascalCase>.al",  
"CRS.FileNamePatternPageCustomizations": "<ObjectNameShort>.<ObjectTypeShortPascalCase>.al",  
"CRS.ObjectNamePrefix": "BSB ",  
"CRS.OnSaveAlFileAction": "Rename",
```

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 three analyzers:

- **CodeCop** is an analyzer that enforces the official AL Coding Guidelines ([CodeCop Analyzer - Business Central | Microsoft Learn](#)).
- **PerTenantExtensionCop** is an analyzer that enforces rules that must be respected by extensions meant to be installed for individual tenants ([PerTenantExtensionCop Analyzer - Business Central | Microsoft Learn](#)).
- **AppSourceCop** is an analyzer that enforces rules that must be respected by extensions meant to be published to Microsoft AppSource ([AppSourceCop Analyzer - Business Central | Microsoft Learn](#)).
- **UICop** is an analyzer that enforces rules that must be respected by extensions meant to be installed for individual tenants ([UICop Analyzer - Business Central | Microsoft Learn](#)).

Enable Analyzers / Define Analyzers to be used

Enable all Analyzers in settings.json:

```
"al.enableCodeAnalysis": true
```

Define Anylyzers to be used in settings.json:

```
"al.codeAnalyzers": [  
    "${CodeCop}",  
    "${PerTenantExtensionCop}",  
    "${UICop}"  
]
```

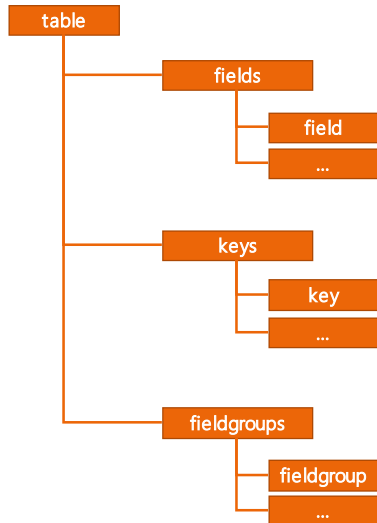
Developing in AL

Developing in AL

- Defining data model.
- Presenting the data.
- Debugging
- Writing code.

Defining data model

Structure of a table



AL file structure of a table

```
table 50100 "Table Sample"
{
    //Table Properties
    fields{}
    keys{}
    fieldgroups{} } Object Content
    //Table Triggers
    //Global Labels
    //Global Variables
    //Other Functions
}
```

Basics of table extensions

- Change existing tables that are not in the current app but in a base app.
- Possible changes are
 - Adding new fields (**fields**)
 - Adding new keys, but limited to fields of this extension
 - Adding New Methods (Functions)
 - Connection to triggers **OnInsert, OnModify, OnDelete, OnRename** - either before or after
 - Changing selected properties of existing fields (modify) and the page itself
- Table extensions can contain variables and program code.

Full list of properties and comparison:

<https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/properties/devenv-table-property-overview>

AL file structure of a tableextension

```
tableextension 50100 "Sample TableExt" extends BaseTable
{
    fields
    {
        modify(ExistingField) { ... }
        field(NewField) { ... }
    }
    keys
    {
        key { ... }
    }
    fieldgroups { ... }
    //"var" und Funktionen
}
```

Companion Tables (up to BC22)

- TableExtension fields and keys are created in several Companion Tables in addition to the extended table.
- Companion tables also contain the primary key of the extended table.
- During the schema synchronization, data records with initial values are automatically generated for all data records in the extended table.

Customer		Customer\$2e454762-3069-4358-84f4-aac98a890888	
No_ [Base]	Name [Base]	No_ [Base]	Shoe Size [Ext1]
		Customer\$9d8ed184-07ba-4219-860d-be711aa1c888	
		No_ [Base]	Hair Color [Ext2]

Companion Tables (from BC23)

- Added fields from all extensions to a table are now stored in the same companion table.
- Server will never need to do more than a single join of the base table to its companion table.
- This drastically reduce the performance impact of table extensions to base tables.

Customer\$437dbf0e-84ff-417a-965d-ed2bb9650972	
No_	Name
Customer\$437dbf0e-84ff-417a-965d-ed2bb9650972\$ext	
No_	Shoe_Size\$2e454762-... Hair_Color\$9d8ed184-...

Basics of Permission Set and Permission Set Extension (BC 18)

When a table is defined, the code analyzers require it to be contained in a permission set of the extension!

- Permission Set Object
 - Describes permissions on objects.
 - Can be set with a property to assignable or not assignable.
 - Can include other permission sets.
 - See also <https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-permissionset-object>
- Permission Set Extension Object
 - Adds permissions to an existing permission set defined in AL.
 - See also <https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-permissionset-ext-object>

Exercise: Creating Table 50100 BSB Book

Field No.	Name und Caption	Data Type	Length	Comment
1	No.	Code	20	Must not be empty
2	Description	Text	100	
3	Search Description	Code	100	Standard implementation
4	Blocked	Boolean		
5	Type	Option		,Hardcover,Paperback
7	Created	Date		Not editable
8	Last Date Modified	Date		Not editable
10	Author	Text	50	
11	Author Provision %	Decimal		Min 0, max 2 decimal places
15	ISBN	Code	20	
16	No. of Pages	Integer		
17	Edition No.	Integer		
18	Date of Publishing	Date		

Exercise: Tableextension 50100 BSB Customer

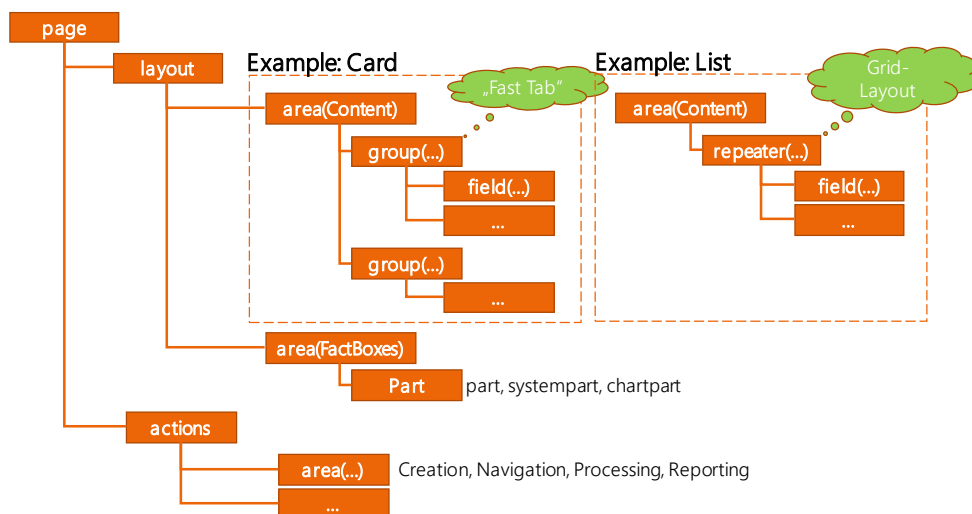
Field No.	Name und Caption	Data Type	Length	Comment
50100	<i>BSB Favorite Book No.</i>	Code	20	TableRelation BSB Book
50101	<i>BSB Favorite Book Description</i>	Text	100	FlowField

Presenting the data

Object types

- Page
- Page Extension
- Profile & Page Customization
- Report
- XmlPort
- Query

Struktur of a page



Extending existing pages

- Object: pageextension
- Snippet: tpageext
- The following changes are possible:
 - Adding new groups, fields, parts, and actions.
 - Functions: addfirst, addlast, addafter, addbefore
 - Moving groups, fields, parts, and actions.
 - Functions: movefirst, movelast, moveafter, movebefore
 - Modifying properties on existing groups, fields, parts, and actions.
 - Function: modify
- Full list of properties and comparison:
- <https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/properties/devenv-page-property-overview>

```
pageextension 50100 "BSB Customer Card" extends "Customer Card"
{
    layout
    {
        addafter(General)
        {
            group("BSB Bookstore")
            {
                Caption = 'Bookstore';
                field("BSB Favorite Book No."; "BSB Favorite Book No.")
                {
                    ApplicationArea = All;
                    Importance = Promoted;
                }
            }
        }
    }
}
```

Exercise: Creating Book Card page

50100 BSB Book Card

Book Card | Arbeitsdatum: 27.01.2022

B1 - Buch 1

General

No. B1

Description Buch 1

Search Description BUCH 1

Gesperrt ☐

Autor Autor 1

Type Hardcover

ISBN 1

Created 06.04.2021

Last Date Modified

Invoicing

Autor Provision % 11

Publishing

Edition No. 1

Date of Publishing 07.04.2021

No. of Pages 10

Links +

(In dieser Ansicht kann nichts angezeigt werden)

Notizen +

(In dieser Ansicht kann nichts angezeigt werden)

Exercise: Creating Book List page

50101 BSB Book List

Books: Alle Suchen + Neu Löschen In Excel öffnen

No. 1	Description	ISBN	Autor
B1	Buch 1	1	Autor 1
B2	Buch 2	2	Autor 2
B3	Buch 3	3	Autor 3

Links +

(In dieser Ansicht kann nichts angezeigt werden)

Notizen +

(In dieser Ansicht kann nichts angezeigt werden)

Exercise: Extending Customer Card page

Pageextension 50100 BSB Customer Card

Debitorenkarte | Arbeitsdatum: 27.01.2022

10000 · Möbel-Meller KG

Neuer Beleg Genehmigen Genehmigungsanforderung Preise und Rabatte Navigieren Debitor Weitere Optionen

Allgemein

Mehr anzeigen

Nr.10000

NameMöbel-Meller KG

Saldo (MW)259.426,65

Fälliger Saldo (MW)0,00

Kreditlimit (MW)0,00

Gesperrt

Gesamtumsatz0,00

Kosten (MW)0,00

Bookstore

Favorite Book No.

Favorite Book Description

Find Objects in Page or Report Search

- To find a object in the search, use the following object properties:
 - UsageCategory
 - ApplicationArea

```
page 50101 "Book List"
{
    Caption = 'Books';
    PageType = List;
    SourceTable = Book;
    Editable = false;
    UsageCategory = Lists;
    ApplicationArea = All;
}
```

Customizing existing pages

- Object: pagecustomization
- Snippet: tpagecust
- Must be published with a profile.
- More restrictions than page extension object.
 - No variables, procedures or triggers
- Add changes to page layout and actions.
- Doesn't require an object id.

```
pagecustomization "BSB Customer Card" customizes "Customer Card"
{
    layout
    {
        modify("Location Code") { Visible = false; }
    }

    actions
    {
        movebefore(NewSalesInvoice; NewSalesOrder)
    }
}
```

```
profile "BSB Bookstore"
{
    Caption = 'get&use Academy Bookstore';
    RoleCenter = "Order Processor Role Center";
    Customizations = "BSB Customer Card", "BSB Customer List";
}
```

Exercise: Adding Book Factbox and Profile

- Create page 50102 **BSB Book Factbox**.
 - Fields: As shown in screenshot.
 - No.-OnDrillDown trigger opens the Book Card.
- Add the FactBox after **Customer Picture** on **Customer Card**.

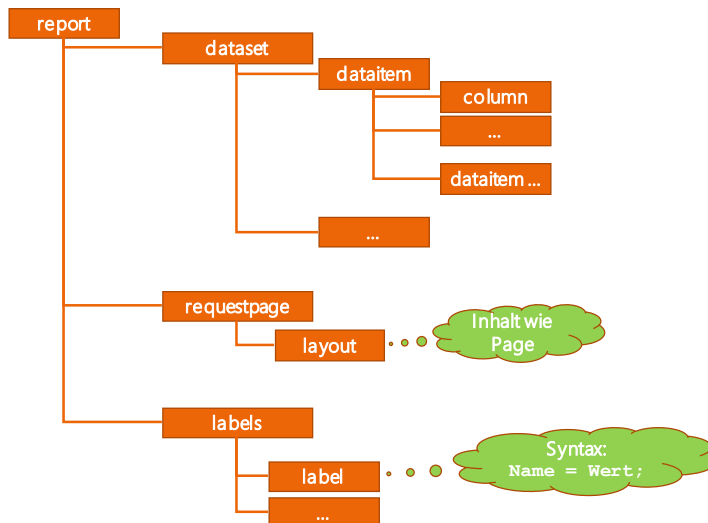
Book Details	
Book No.	B1
Book Description	Buch 1
Date of Publishing	07.04.2021
No. of Pages	10
Autor	Autor 1

- Create a new profile **BSB Bookstore** and a pagecustomization for page **Customer List** and **Customer Card**.
 - Hide the *Location Code*.

Creating reports

- Snippet: treport
- Two tasks:
 - Declare data model.
 - Define visual layout.
- Report layout is defined outside of the report object, but a default layout is being generated when report is built for the first time.

Struktur of a report



AL file structur of a report

```
report 50100 „Report Sample”
{
    //Report Properties
    dataset{ }
    requestpage{ }
    labels{ }
    //Report Triggers
    //Global Labels
    //Global Variables
    //Other Functions
}
```

} Object Content

AL file structur of a report: dataset definition

```
dataset
{
    dataitem(Customer;Customer)
    {
        column(Name;Name) { IncludeCaption = true; }

        dataitem(Orders;"Sales Header")
        {
            DataItemLink = "Sell-to Customer No." = field("No.");
            column(Amount;Amount) { IncludeCaption = true; }
        }
    }
}
```

Creating reportextensions (1)

- Snippet: treportext
- Two tasks:
 - Declare data model.
 - Define visual layout.
- Report layout is defined outside of the report object, but a default layout is being generated when reportextension is built for the first time.
- For a report to be extended, the **Extensible** property must be set to **true**.
- Available from Dynamics 365 Business Central 2021 release wave 1 (BC18).

Creating reportextensions (2)

- With report extensions, you can extend an existing report by:
 - Adding columns to existing dataitems in the report dataset.
 - Adding new dataitems but not on the root level.
 - Adding OnPreReport() and OnPostReport() trigger implementations.
 - Adding to request pages.
 - Adding to a new report layout to reflect the new fields that are added with an extension.
- In an upcoming service update, Microsoft will add support for labels, as well as some limited abilities to modify existing data items, such as adding to request fields, calcfields, or triggers.
- The layout
 - in a report extension will not automatically be used when the report extension is deployed.
 - is created as a built-in report layout during installation.
 - can be used if it's selected in **Report Layout Selection** in the field **Report Layout Description**.

Exercise: Creating the Book List report

Create the report 50100 **BBS Book - List**.

- The user can decide to show **Page Count**.
- Call report in action of page **Book List**.

BOOK - LIST

Gespeicherte Einstellungen

Änderungen an den Optionen und Filtern unten werden nur gespeichert unter: "Zuletzt verwendete Optionen und Filter"

Standardwerte verwenden von: Zuletzt verwendete Optionen und Filter

Options

Show No. of Pages: ☒

Filter: Book

* No.

* Author

+ Filter...

03.01.2020 07:17
Page 1 of 1
BC15ONPREM\USER

Senden an... Drucken Vorschau Abbrechen

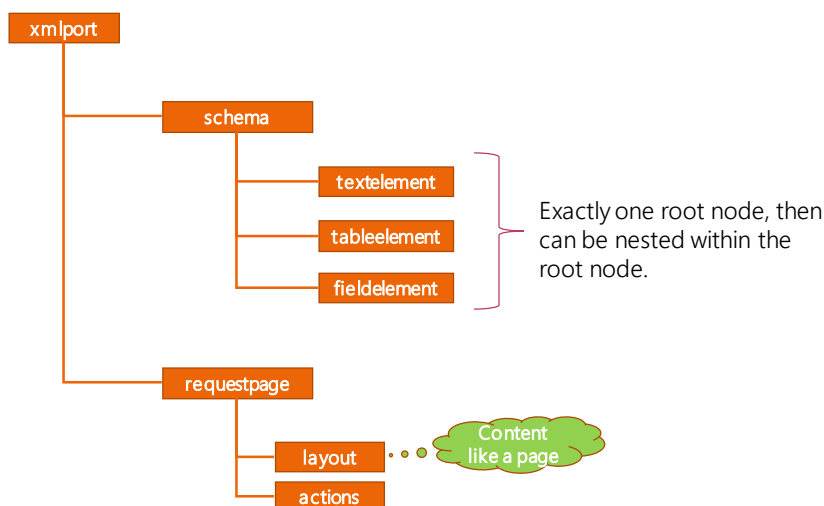
Book - List
CRONUS AG

No.	Description	Author	No. of Pages
-----	-------------	--------	--------------

XMLport – in general

- Imports and exports data in the formats:
 - XML (Extensible Markup Language)
 - Text with delimiters
 - Text with fixed field width
- The call is made:
 - Directly with dialogue
 - Indirectly via another object
- Request pages for XMLports are not supported by the Business Central Web client in versions prior to Dynamics 365 Business Central 2019 release wave 2 CU 2.

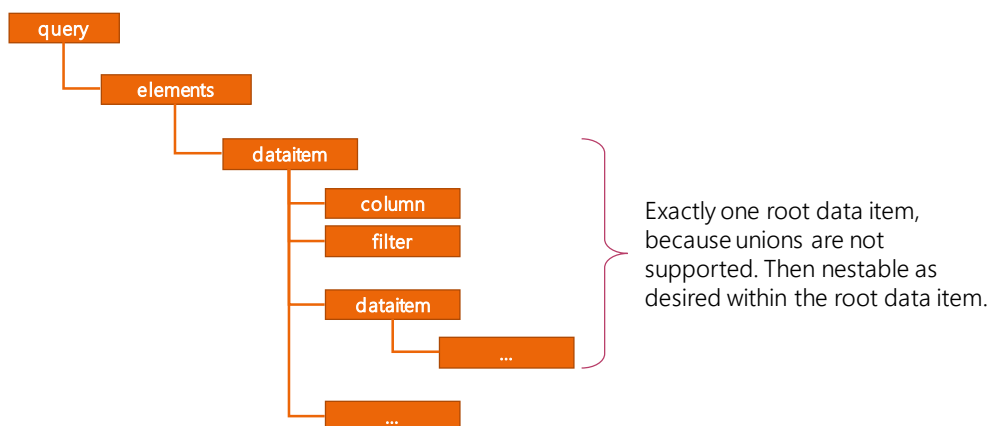
Structur of a XMLport



Query – in general

- Query objects are used when access via records is too slow in specific cases.
- A query offers the following performance benefits:
 - While all columns are selected when working with records, the columns can (and must) be selected individually during the query.
 - Records always represent rows from the same table, while queries support different joins for cross queries across multiple tables.
 - Queries directly support the formation of aggregates using SQL aggregate functions. Records can do this only with Flow Fields or iterative.
- Flow Fields and Flow Filters are supported.
- Queries can be read in AL (forward-facing) and published as a web service for external applications.
- Common use in fact boxes and reports.

Structur of a query



Debugging

Support for debugging

- **F5** starts a debugging session.
 - Always in a new session.
- **Ctrl+F5** starts the client without debugging.
- Use **F9** to set breakpoints.
- To hit the breakpoints, just trigger the code.
- Only breakpoints of Extension code.

Writing code

Extending Application Logic

- Application logic can be extended in:
 - Table Extension objects
 - Page Extension objects
 - Codeunits
- Codeunits are primary means of extending application logic.

Codeunit object overview

- Snippet: tcodeunit
- Just like in C/SIDE.
 - You can set properties (all are the same).
 - You can define OnRun trigger (it is not mandatory).
 - You can create your own procedures (functions).

Procedures

- Snippet: tprocedure
- In addition to simple types, can return new complex types.
 - JSON objects
 - XML objects
 - .Net replacements
- Function types can be set through attributes.

Subscribing to event publishers

- How to do:
 - Declare a procedure.
 - Decorate it with [EventSubscriber] attribute.
Use intellisense to complete the definition of the attribute and to get to the parameters.
- Subscriptions work exactly as from NAV 2016.

.Net wrappers

- HTTP
- JSON
- XML
- Text
- TextBuilder
- List and Dictionary

In general

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.

Text and TextBuilder data type

- TextBuilder based on .Net StringBuilder. Including its functions like:
 - AppendLine
 - Append
 - Replace
 - ...
- Text based on .Net String (it always was). New for Text functions like:
 - Contains
 - StartsWith
 - EndsWith
 - Replace
 - ...

```
local procedure TextAndTextBuilderDemo()
var
    Txt: Text;
    TxtBuilder: TextBuilder;
begin
    TxtBuilder.AppendLine('We can append news lines');
    TxtBuilder.Append('...or characters to lines. ');
    TxtBuilder.Replace('to lines', 'the current line');
    Txt := TxtBuilder.ToText();
    Message(Txt);
    Message(Txt.ToUpper());
    if Txt.EndsWith('line.') then
        Message('Text ends with ''line.'');
end;
```

- Full list: <https://learn.microsoft.com/de-de/dynamics365/business-central/dev-itpro/developer/methods-auto/library>

List and Dictionary data type

- Strongly type lists and dictionaries.
- Based on .Net generic List and Dictionary.
- Only possible with simple types (for now).
- Syntax:
 - List of [Integer]
 - Dictionary of [Integer,Text]

List example

```
local procedure ListDemo()
var
    Names: List of [Text];
begin
    Names.Add('Peter');
    Names.Add('Paul');
    Names.Add('Mary');
    if Names.Contains('Paul') then
        Message('Paul ist in the list at index %1', Names.IndexOf('Paul'));
    Message('Name at index 1: %1', Names.Get(1));
end;
```

Dictionary Example

```
local procedure CountCharacters(Txt: Text; var Dict: Dictionary of [Char, Integer])
var
    I: Integer;
    Cnt: Integer;
begin
    Clear(Dict);

    for I := 1 to StrLen(Txt) do
        if Dict.Get(Txt[I], Cnt) then
            Dict.Set(Txt[I], Cnt + 1)
        else
            Dict.Add(Txt[I], 1);
        end;
    end;
```

foreach

- Iterating over enumerable types in AL.
- Current support for
 - List
 - XmlNodeList
 - XmlAttributeCollection
 - JsonArray

```
local procedure ShowDictionary(var Dict: Dictionary of [Char, Integer])
var
    TxtBuilder: TextBuilder;
    Keys: List of [Char];
    Cnt: Integer;
    C: Char;
begin
    Keys := Dict.Keys();
    foreach C in Keys do begin
        Dict.Get(C, Cnt);
        TxtBuilder.AppendLine(StrSubstNo('%1:%2', C, Cnt));
    end;
    Message(TxtBuilder.ToText());
end;
```


Running C/SIDE and AL Side-by-Side

Until Version 1904 of Business Central on-premises

General

Business Central on-premises until Version 1904 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 new command and argument called **generatesymbolreference** has been added to finsql.exe

Prerequisite – Business Central on-premises server setting

First you must enable the Business Central on-premises server setting "Enable Symbol Loading at Server Startup".

This setting **must be enabled** to allow any symbol generation.

You can do this by PS:

```
Set-NAVServerInstance nav -Stop  
Set-NAVServerConfiguration nav -KeyName  
EnableSymbolLoadingAtServerStartup -KeyValue true  
Set-NAVServerInstance nav -Start
```

If the setting is not enabled, the generatesymbolreference setting with finsql.exe does not have any effect!

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.

```
finsql.exe Command=generatesymbolreference, Database=...
```

This command **should be run at least once** to generate the initial set of symbols to which incremental updates can be applied.

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.

```
finsql.exe generatesymbolreference=yes, Database=...
```

Special Object Type – Extensible Enum

Enum – the alternative for data type Option

Why is an alternative necessary?

- Option is as it is.
- No way to extend options in AL.

Advantages of the new object type Enum:

- It's an object type and therefore the availability is global
- Can be extensible in AL.
- In C/SIDE there are two Option properties .
(EnumTypeId, EnumTypeName) to gain compatibility with Enum.

```
enum 50100 "BSB Book Type"
{
    Extensible = true;

    0 references
    value(0; " ") { Caption = 'None'; }
    0 references
    value(1; Hardcover) { Caption = 'Hardcover'; }
    0 references
    value(2; Paperback) { Caption = 'Paperback'; }
}
```

Extending Enum

An Enum can be extended in order to add more values to the enumeration list in which case the **Extensible** property must be set to "true":

```
enumextension 50200 "BSCL Book Type" extends "BSB Book Type"
{
    0 references
    value(50200; eBook) { Caption = 'E-Book'; }
}
```

Exercise: Change Option to Enum

- In table 50100 **BSB Book** change data type of field 5 *Type* from Option to new enum 50100 **BSB Book Type**.
- When creating the new enum, make sure that you select the values so that the previously entered values of the field *Type* remain unchanged.
- Please note that the values of the data type option and enum are saved as an integer in the database!

In-App Visual Designer

Agenda

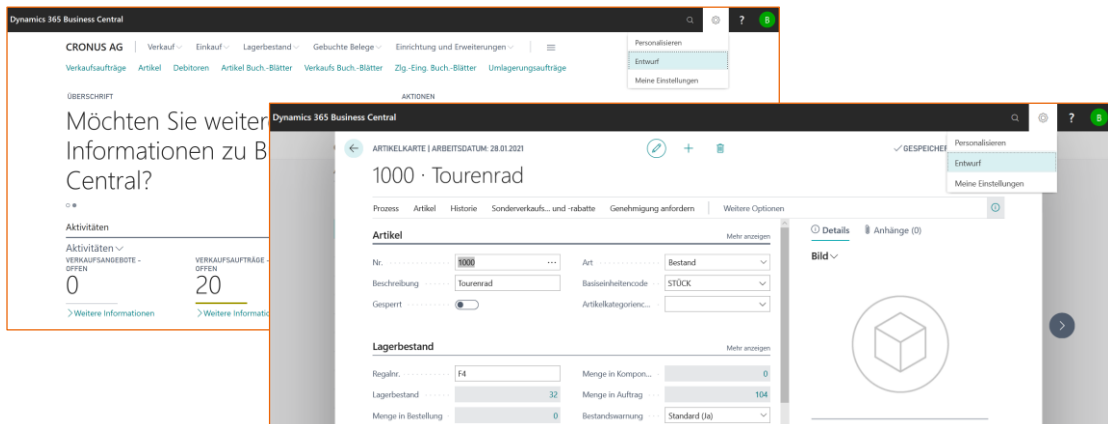
- Accessing in-app designer.
- Features of in-app designer.
- Exporting the VS Code package from in-app designer.
- Working with the VS Code package generated from designer.

What is In-app Designer

- What-You-See-Is-What-You-Get editor for Business Central pages.
- Allows creating, exporting and sharing simple customization of user interface.
- Simulates web, tablet, and phone experiences.
- Powered by "Extensions v2" technology.

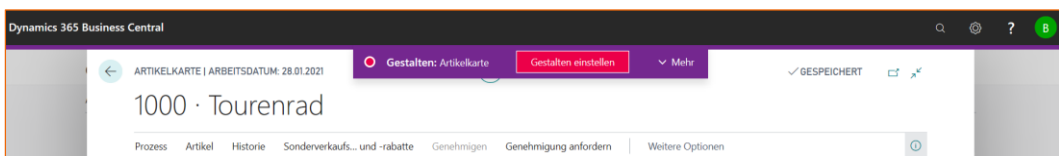
Accessing In-app Designer

- In-app Designer is available in every page:

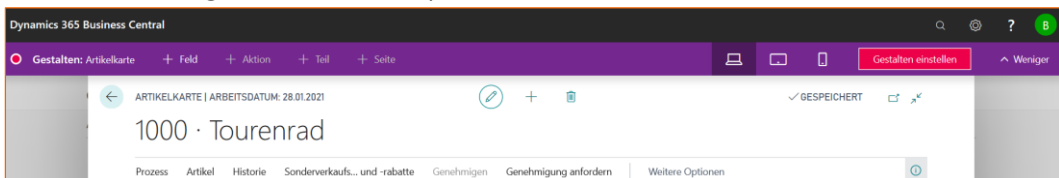


Accessing In-app Designer

- Simple design mode



- Advanced design mode (in simple mode, click More)

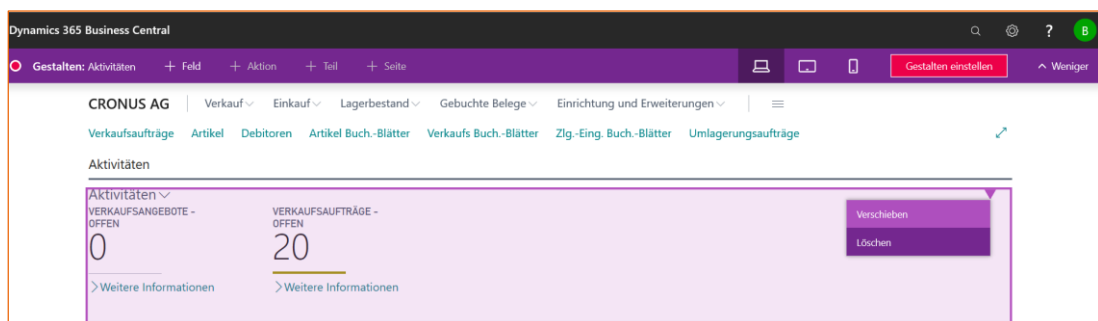


Features of In-app Designer (1)

- Managing fields
 - Hiding fields.
 - Adding fields from the table.
 - Rearranging fields on page (also between FastTabs).
 - Works in repeaters, card-type pages, page parts.
- Managing FastTabs
 - Renaming
- Simulating clients
 - Web
 - Tablet
 - Phone

Features of In-app Designer (2)

- Managing RoleCenters
 - Reposition Cue tile.
 - Hide Cue tile.
 - Hide Cue Group.



Finishing Work

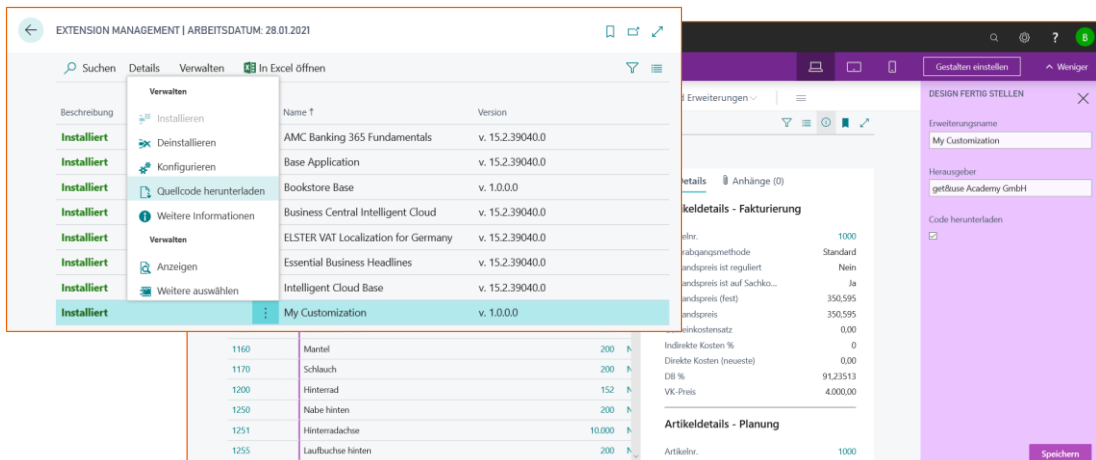
- Customizations can be:
 - Applied to all users of the current tenant.
 - Exported as VS Code package.
 - In Dynamics 365 Business Central exported as package.

Customizations are Extensions

- After designer work is completed, Business Central
 - Builds an extension package.
 - Publishes the extension package.
 - Installs the extension package.

Exporting the package to VS Code

- Customizations can be exported and further edited in VS Code.



To edit in VS Code

- Unzip the downloaded package.
- Right-click the extracted folder.
- Click Open with Code.

Publish and open in in-app designer

- Shortcut VS Code: **F6**
 - Publish current extension.
 - Open it in the in-app designer.
- Shortcut VS Code: **ALT+F6**
 - Pull changes made by the in-app designer.
 - Add them as code back into your source.

Enhancing the Extension

Agenda

- Dependencies between extensions
- Control Add-ins
- Translations

Dependencies

Dependencies

- An extension can depend on another extension.
- Dependencies:
 - Help structure more complex deployment scenarios.
 - Boost code and business logic reusability.
 - Increase maintenance flexibility.

Declaring dependencies

Dependency is declared in the manifest file (app.json)

```
"dependencies": [  
  {  
    "id": "e08b0233-2914-49ac-a5f8-fadc71a472fb",  
    "name": "Bookstore",  
    "publisher": "get&use Academy GmbH",  
    "version": "1.0.0.0"  
  }  
],
```

Exercise: Extension with Dependency

1. Create new a Extension **Bookstore Customer Likes** (ID-Range 50200-50249) with dependency on **Bookstore Base**. Use BSCL as prefix.
2. Extend table **BSB Book** with the new FlowField 50200 ***BSCL No. of Customer Likes***.
3. Add this field to the page **BSB Book Factbox**.
4. Extend enum 50100 **BSB Book Type** with value „eBook“ and caption „E-Book“.
5. Add procedure **ShowFavoriteBook** to the table **Customer**
6. Add action to open **BSB Book Card** to **Customer Card** (should only be enabled, if ***Favorite Book No.*** exist)
7. A book record should only be deletable, if no customer had liked this book.

Control add-ins

Control add-ins

- AL type: **controladdin**
 - Replacement for old XML manifest.
 - Interface for bidirectional JavaScript-to-AL communication.
 - Properties: <https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-control-addin-object>
 - Style Guide: <https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-control-addin-style>
- Control in page: **usercontrol**
 - Refers to controladdin.
 - Implements the triggers.

Translations

Translations

- New translation process decouples translation from development.
- New label syntax.
 - Default translation.
 - Some attributes.
- Generate standard format translation file (XLIFF).
- Translation file can be combined with the extensions to support Multilanguage.
- Caption and CaptionML cannot be used simultaneously.

Label syntax

- Comment, Locked and MaxLength are optional.
- New data type Label.
- Same syntax for captions and labels.

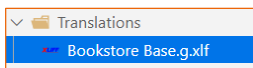
```
TestMsg: Label 'This is only a test message', Comment = 'Message displayed when opening page.', MaxLength = 30, Locked = true;
```


Enable translation feature in app.json

- Configure in app.json.

```
"features": [  
  "TranslationFile"  
]
```

- Invoke package command **Ctrl+Shift+B** to get the xlf file.



- Use external tool to translate.

Translation file usage

- The generated translation file cannot be used as the translated file.
- During every compilation, the generated translation file will be overwritten with a new file.
- Files need to be copied and modified.
- You need one .xlf file per target language.
- A translated file must contain:
 - Target language attribute in the <file> element.
 - <target> element directly under the <source> element.
 - The <target> element must hold the translated value.
 - The <target> element must have an attribute "state=translated".

How to translate manually

- Copy the generated .xlf file.
 - Tip: set the target language in the file name.
- Open the copied file in VS Code.
- Use find and replace with regular expressions to add the <target> tags.
 - Search for:
(<source>([^\<]*)</source>)
 - Replace with:
\$1\n\t\t\t\t\t<target state="translated">\$2</target>
- Fill in the translations.

How to translate with tools

- Copy the generated .xlf file.
 - Tip: set the target language in the file name.
- Use one of these tools to translate the file.
 - Multilingual app toolkit 4.0 editor:
<https://learn.microsoft.com/de-de/windows/apps/design/globalizing/multilingual-app-toolkit-editor-downloads>
 - Microsoft Lifecycle Services:
<https://lcs.dynamics.com/>
→ how-to: <http://www.dynamics.is/?p=3003>

Migrate your IP to Extension

Basic Steps

- Starting Point: Deltas of your app, in previous version of NAV.
- Apply those Deltas to NAV 2018.
- Export objects in New Syntax.
- Create new Deltas (hence: in New Syntax format).
- Convert Deltas to AL files.

Finally

- You're set to go.
- Fix compilation issues (not part of this workshop).
- Hit **F5** and see what happens.

Beyond Tech

- NAV add-ons don't directly translate to Dynamics 365 Business Central.
- Add SaaSified User Experience.
 - Notifications
 - Assisted Setups and Wizards
 - Tooltips and Application Areas
- Integration With New Technologies.
 - PowerBI, PowerApps, Flow, CDS, CDM, AI, Machine Learning, IoT.
- Take into account Marketing Aspects.

Marketing Aspects

- Customer Journey
- Go-To-Market
- Landing Page
- Online Resources
- AppSource
- User Experience is much more important.
- Split Functionality in Basic, Advanced, Premium levels.

Event Recorder

Event Recorder

- Available from fall release 2018.
- Easily find the right events to subscribe to.
- How it is done:
 - Search for Event Recorder and start the page.
 - Select Record Events, Start.
 - Open a new Tab in your browser with Business Central Webclient.
 - Do whatever you want to record.
 - In the web browser, switch back to the tab with the Event Recorder.
 - Select Record Events, Stop.
 - Now you can see all recorded events.

Event Subscriber Code Snippet

Get easily the Subscriber Code for a recorded event by clicking *Get AL Snippet*.

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
6	Custom Event	2	Codeunit	Graph Mgt - Ge...	OnGetIsAPIEnabled		Codeunit	Graph Mgt - Ge...	IsAPIEnabled	Get AL Snippet
7	Custom Event	1	Table	Customer	OnAfterSetLastModifiedDateTi...		Table	Customer	SetLastModifiedDateTime	Get AL Snippet
8	Custom Event	1	Table	Customer	OnBeforeIsContactUpdateNeed...		Table	Customer	IsContactUpdateNeeded	Get AL Snippet
9	Trigger Event	1	Table	Customer	OnBeforeModifyEvent					Get AL Snippet
10	Custom Event	1	Codeunit	Graph Mgt - Ge...	OnGetIsAPIEnabled		Codeunit	Graph Mgt - Ge...	IsAPIEnabled	Get AL Snippet
11	Custom Event	1	Codeunit	"Global			Table	Customer	OnModify	Get AL Snippet
12	Custom Event	1	Codeunit	"Global			Codeunit	GlobalTriggerMa...	GetGlobalTableTriggerMask	Get AL Snippet
13	Custom Event	1	Codeunit	"Global			Table	Customer	OnModify	Get AL Snippet
14	Custom Event	1	Codeunit	CRM In			Codeunit	CRM Integration...	GetDatabaseTableTriggerSetup	Get AL Snippet
15	Custom Event	1	Codeunit	Graph M			Codeunit	Graph Mgt - Ge...	IsAPIEnabled	Get AL Snippet
16	Custom Event	1	Codeunit	Graph M			Codeunit	Graph Mgt - Ge...	IsAPISubscriptionEnabled	Get AL Snippet
17	Custom Event	1	Codeunit	GlobalT			Codeunit	GlobalTriggerMa...	GetDatabaseTableTriggerSetup	Get AL Snippet
18	Custom Event	1	Codeunit	Change Log Ma...	OnAfterIsAlwaysLoggedTable		Codeunit	Change Log Ma...	GetDatabaseTableTriggerSetup	Get AL Snippet
19	Custom Event	1	Codeunit	Change Log Ma...	OnAfterIsAlwaysLoggedTable		Codeunit	Change Log Ma...	IsAlwaysLoggedTable	Get AL Snippet

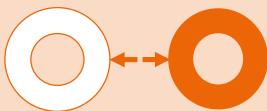
Apps for Dynamics 365

Agenda

- App types
- Customization

App types

Integrating and extending



Connect

Connected Apps Connect another service with Dynamics 365 Business Central

API's



Add-On

Add-on Apps enrich the Dynamics 365 Business Central

Extensions 2.0



Embed

Embed Apps provides an end-to-end solution for an industry or micro-vertical

Extensions 2.0,
API's, etc. ...

Existing story: Add-ons

- Easily deployable customizations through extensions.
- Enrich business logic, UI, etc from inside .
- Supported in SaaS, PaaS and on-prem.
- Stronger v2 story with Visual Studio Code and In Client Designer.
- Surfaced in and installed from the [AppSource](#) marketplace.

Introducing: Connect Apps

- Allows easy data integration using new REST APIs.
 - Connect from outside, e.g., a different SaaS service, to read and write data.
 - Connection set up from within the outside service, not in Dynamics 365 Business Central.
 - API standardized across all cloud tenants in Dynamics 365 Business Central, supporting volume integrations.
- Connect through multiple endpoints.
 - Cloud Dynamics 365 Business Central tenants - api.businesscentral.dynamics.com.
 - Microsoft product wide - [Microsoft Graph](https://developer.microsoft.com/en-us/graph/) (<https://developer.microsoft.com/en-us/graph/>).
- Relies on Azure Active Directory (AAD) authentication.
- Will become part of AppSource as well.

Customizations

Customizations

Cloud

- Only through extensions.
- No code customizations.
 - Lift & shift approach.
- Three different object ranges.
- AppSource for public apps.
- Tenant customizations.

On-premises

- Supports extensions.
- Code customizations of BaseApp.
- Two different object ranges.
- No AppSource.

Object ranges in Business Central

Object range	Extension type	Available in AppSource	Available on-premise
50.000 – 99.999	Tenant customization		X
1.000.000 – 60.000.000	ISV number range	X	X
70.000.000 – ...	AppSource number range	X	

Publishing extensions

- AppSource
 - Benefit marketing power of Microsoft.
 - Specific procedure and validation process.
 - Also applies to solutions in ISV range.
- Upload directly in tenant.

Publishing on AppSource

- Member of Microsoft Partner Network.
- Sign up for SPA or PRA.
 - SPA for selling partners.
 - PRA for non-selling partners.
- Request a number range.
 - Decide for which platform.
- Create a developer account.
- Get access to Cloud Partner Portal.
- Develop the app.
- Publish the app through Cloud Partner Portal.

Developing the app

- Three options:
 - Sandbox environment
 - Docker based environment on Azure
 - Local docker based environment

Technical aspects

- Code analysis test
- Examples of critical errors (must be resolved):
 - Using file system
 - Using ASSERTERROR
 - Not encrypting sensitive table data
- Examples of important errors (should be resolved):
 - Temporary files not cleaned up
 - Using files without Codeunit 419
 - Not working with temporary files
 - Using specific time zone

Technical aspects

- Comply with Dynamics 365 Business Central user-experience.
- Make use of Application Area.
- Provide tooltips, a setup and user guide.
- Must contain translations for targeted country/region languages.
- Must provide automatic test package.

“Ready to Go” program

- Resource library on Dynamics Learning Portal.
- ISV Development Centers and Master VARs provide validation workshops.
- Microsoft Collaborate portal

SaaSified User Experience

Session objectives

- SaaSified User Experience
- Assisted Setup & Wizards
- Notifications
- Application Areas

SaaSified User Experience

Customer Journey

- Goal: getting end-users to subscribe to your product.
- You can lose potential customers on every step of the way.
- Prospects might not find your app / landing page.
- Prospects might not want to give their e-mail address to claim their trial period.
- Prospects might find your app too complicated once they've downloaded it.
- You need to gently draw your prospects deeper into your app until they make the decision to subscribe to it.

SaaSified User Experience

- Quick adoption.
- Lead users to features.
- Step by step guidance through setup.
- Contextual information.
- Hide irrelevant fields.
- Simplify the user experience.

An App Look & Feel

- Apps are different from on prem verticals / horizontals.
- We need to invest as much time and effort into the user experience as we need to invest into functionality – or probably more.
- Less is more; simplicity is the key.
- Every user recognizes an app UI immediately.
- It is intuitive.
- It is visual.
- It can be handled without any training.

SaaSified techniques

Wizards

- Step by step guidance through a setup process

Assisted Setup

- List of setups and information about status

Notifications

- Contextual information at the right moment

Application Areas

- Simplifying the user experience

Assisted Setup & Wizards

Wizards

What is a wizard?

- Crucial part in Assisted Setup.
- Series of user input screens or steps.
- Hides irrelevant options.
- User can navigate with Next and Previous buttons.
- Web Client shows buttons with specific styling.
- Based on a page of type NavigatePage.

How to create a Wizard – Step 1

- Create new page of type NavigatePage:
 - Create new *.al file in VSCode, use the 'tpage' snippet.
 - Set the Id and PageName.
 - Set PageType property to NavigatePage.
 - Set SourceTable property to **Company Information**.

```
page 50130 "BSB Company Information Wizard"  
{  
    PageType = NavigatePage;  
    SourceTable = "Company Information";  
    Caption = 'Setup Company Information';  
    SourceTableTemporary = true;
```

Step 2 – Add Steps

- Add a Group for each Step

```
layout
{
    area(content)
    {
        group(Step1)
        { ...
        }

        group(Step2)
        { ...
        }

        group(Step3)
        { ...
        }
    }
}
```

- Create a global integer type variable 'CurrentStep'

```
var
    10 references
    CurrentStep: Integer;
```

- On each group, set the Visible property with an expression

```
group(Step1)
{
    Visible = CurrentStep = 1;
```

Step 3 – Add content

- Create a sub-group inside the steps and set the caption
- Set InstructionalTextML to provide guidance
- Add fields to the sub-group

```
group(Step1)
{
    Visible = CurrentStep = 1;
    0 references
    group(CompanyName)
    {
        Caption = 'Company Name';
        InstructionalText = 'Provide the name of your company';
        0 references
        field(Name; Name)
        {
            ApplicationArea = All;
        }
    }
}
```

Step 4 – Add Actions

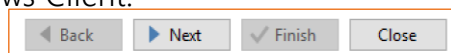
- Add three actions to facilitate navigation
 - Names
 - ActionBack, ActionNext, ActionFinish
 - Captions
 - Back, Next, Finish
- For each action, set the InFooterBar property to true
 - In combination with NavigatePage value for the PageType property, this shows actions as navigation buttons

```
actions
{
    0 references
    area(processing)
    {
        0 references
        action(ActionBack)
        {
            ApplicationArea = All;
            Caption = 'Back';
            InFooterBar = true;

            trigger OnAction()
            begin ...
            end;
        }
        0 references
        action(ActionNext)
        { ...
        }
        0 references
        action(ActionFinish)
        { ...
        }
    }
}
```

Step 4 – Add Actions

- Create three global variables of type Boolean
 - ActionBackAllowed, ActionNextAllowed, ActionFinishAllowed
- On each action, set the Enabled property to the appropriate global variable
- On each action, set the Image property
 - Back: PreviousRecord
 - Next: NextRecord
 - Finish: Approve
- In the Windows Client:



```
action(ActionBack)
{
    ApplicationArea = All;
    Caption = 'Back';
    Enabled = ActionBackAllowed;
    Image = PreviousRecord;
    InFooterBar = true;

    trigger OnAction()
    begin ...
    end;
}
```

Step 5 – Add Code for Navigation

- Create a function SetControls

```
local procedure SetControls()
begin
    ActionBackAllowed := CurrentStep > 1;
    ActionNextAllowed := CurrentStep < 3;
    ActionFinishAllowed := CurrentStep = 3;
end;
```

- Set starting values in the OnOpenPage trigger

```
trigger OnOpenPage()
begin
    CurrentStep := 1;
    SetControls();
end;
```

- Create a function TakeStep

```
local procedure TakeStep(Step: Integer)
begin
    CurrentStep += Step;
    SetControls();
end;
```

Step 5 – Add Code for Navigation

- Create function StoreCompInfo

```
local procedure StoreCompInfo()
var
    CompInfo: Record "Company Information";
begin
    if not CompInfo.Get() then begin
        CompInfo.Init();
        CompInfo.Insert();
    end;

    CompInfo.TransferFields(Rec, false);
    CompInfo.Modify(true);
end;
```

- Create function FinishAction

```
local procedure FinishAction()
begin
    StoreCompanyInfo();
    CurrPage.Close();
end;
```

Step 5 – Add code for navigation

- Add OnAction triggers to the navigation actions

```
action(ActionBack)
{
    ApplicationArea = All;
    Caption = 'Back';
    Enabled = ActionBackAllowed;
    Image = PreviousRecord;
    InFooterBar = true;

    trigger OnAction()
    begin
        TakeStep(-1);
    end;
}
```

```
action(ActionNext)
{
    ApplicationArea = All;
    Caption = 'Next';
    Enabled = ActionNextAllowed;
    Image = NextRecord;
    InFooterBar = true;

    trigger OnAction()
    begin
        TakeStep(1);
    end;
}
```

```
action(ActionFinish)
{
    ApplicationArea = All;
    Caption = 'Finish';
    Enabled = ActionFinishAllowed;
    Image = Approve;
    InFooterBar = true;

    trigger OnAction()
    begin
        FinishAction();
    end;
}
```

Step 6 – Add Image Header

- Create two global record variables for the Media Repository table and for the Media Resources table.

```
MediaRepositoryStandard: Record "Media Repository";
2 references
MediaRepositoryDone: Record "Media Repository";
2 references
MediaResourcesStandard: Record "Media Resources";
3 references
MediaResourcesDone: Record "Media Resources";
```

Step 6 – Add Image Header

- Create two new groups above the "Step 1" Group
- Add one field to each group
- Set SourceExpr to the Image field of the Media Resources variables
- Set ShowCaption property to 'false'
- Note the expression in the Visible property

```
group(StandardBanner)
{
    Editable = false;
    Visible = TopBannerVisible AND (CurrentStep < 3);
    0 references
    field(MediaResourcesStandard; MediaResourcesStandard."Media Reference")
    {
        ApplicationArea = All;
        ShowCaption = false;
    }
}
0 references
group(FinishedBanner)
{
    Editable = false;
    Visible = TopBannerVisible AND (CurrentStep = 3);
    0 references
    field(MediaResourcesDone; MediaResourcesDone."Media Reference")
    {
        ApplicationArea = All;
        ShowCaption = false;
    }
}
```

Step 6 – Add image header

- Create a new function to load the top banner images

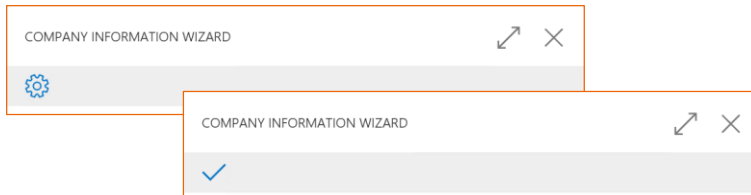
```
local procedure LoadTopBanners()
begin
    if MediaRepositoryStandard.GET('AssistedSetup-NoText-400px.png', Format(CurrentClientType())) and
       MediaRepositoryDone.GET('AssistedSetupDone-NoText-400px.png', Format(CurrentClientType()))
    then
        if MediaResourcesStandard.GET(MediaRepositoryStandard."Media Resources Ref") and
           MediaResourcesDone.GET(MediaRepositoryDone."Media Resources Ref")
        then
            TopBannerVisible := MediaResourcesDone."Media Reference".HasValue();
        end;
    end;
end;
```

- Call this function from the OnInit trigger

```
trigger OnInit()
begin
    LoadTopBanners();
end;
```

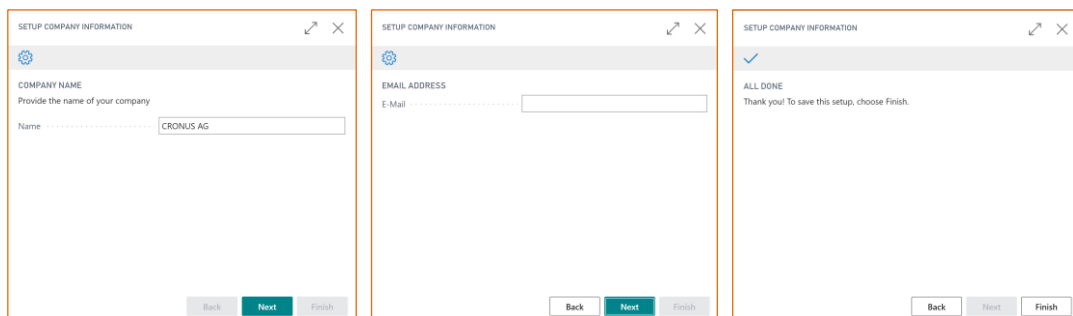

Step 6 – Add image header

- The media will render as a header as long as it is alone in the top-level group
- Image headers have gray background and are left-aligned



Exercise: Wizard page

- Create page 50130 **Company Information Wizard** using snippet **tpagewizard3stepswaldo**.



Assisted Setup & Wizards

What is Assisted Setup?

- List of setup scenarios.
- Use a wizard with relevant options.
- Information about status.
- Predefined set of setup scenarios.
 - Cash Flow Forecast, Approval Workflows, Email, CRM Connection, ...
- Possible to add new setup scenarios.

Use Assisted Setup

Available from Role Centers or search for Assisted Setup

CRONUS AG Verkauf ▾ Einkauf ▾ Lagerbestand ▾ Gebuchte Belege ▾ Einrichtung und Erweiterungen ▾ ☰						
Unterstützte Einrichtung: Alle ▾ 🔍 Suchen 📄 In Excel öffnen Weitere Optionen 🔍 ☰ 📄 🔗						
Name	Abgeschlossen	Hilfe	Video	Gruppe	Übersetzter Name	
Set up Cloud Migration	☐	–	–	Erste Schritte mit Dyn...	–	
Mein Unternehmen einrichten	☐	–	–	Erste Schritte mit Dyn...	Mein Unternehmen einrichten	
Genehmigungswflows einrichten	☐	–	Ansehen	Einstellungen an Ihre ...	Genehmigungswflows einrichten	
E-Mail einrichten	☐	Lesen	Ansehen	Einstellungen an Ihre ...	E-Mail einrichten	
Ihren Unternehmensposteingang in Outlook einrichten	☐	Lesen	Ansehen	Einstellungen an Ihre ...	Ihren Unternehmensposteingang in Outlook einrichten	
Geschäftsdaten migrieren	☐	Lesen	Ansehen	Erste Schritte mit Dyn...	Geschäftsdaten migrieren	
E-Mail-Protokollierung einrichten	☐	–	Ansehen	Einstellungen an Ihre ...	E-Mail-Protokollierung einrichten	

Add a new setup scenario

- Objects involved:
 - Codeunit 1990 **Guided Experience**
- Register Wizard for Assisted Setup.
 - Event OnRegisterAssistedSetup:
 - Subscribe to this event.
 - Call GuidedExperience.InsertAssistedSetup() to add wizard page of your extension.
 - Call GuidedExperience.AddTranslationForSetupObjectTitle() to add title translations.
 - Call GuidedExperience.AddTranslationForSetupObjectDescription() to add description translations.
- Wizard should update the status in the **Guided Experience Item** table.
 - Event OnAfterRunAssistedSetup:
 - Subscribe to this event.
 - To store the updated status using GuidedExperience.CompleteAssistedSetup().

Exercise: Register Wizard in Assited Setup

- Create codeunit 50130 **BSB Company Info. Wizard Meth**
- Register **BSB Company Information Wizard** in **Assisted Setup** by using snippet **tassistedsetupwaldoafterv15**.

Notifications

What are Notifications?

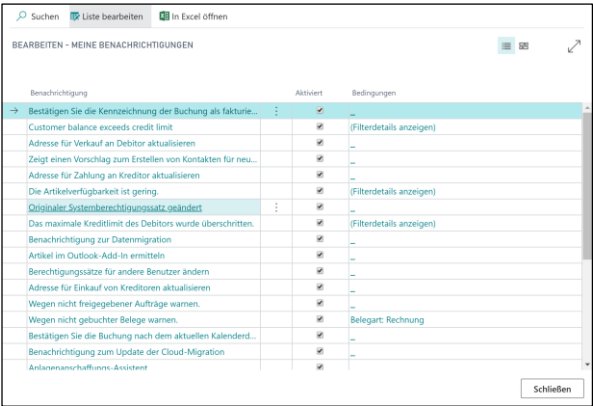
- Non-intrusive message.
- Contextual
- Optional actions (max 4).
- User can continue to work.
- Works with any client.

Notification – Example

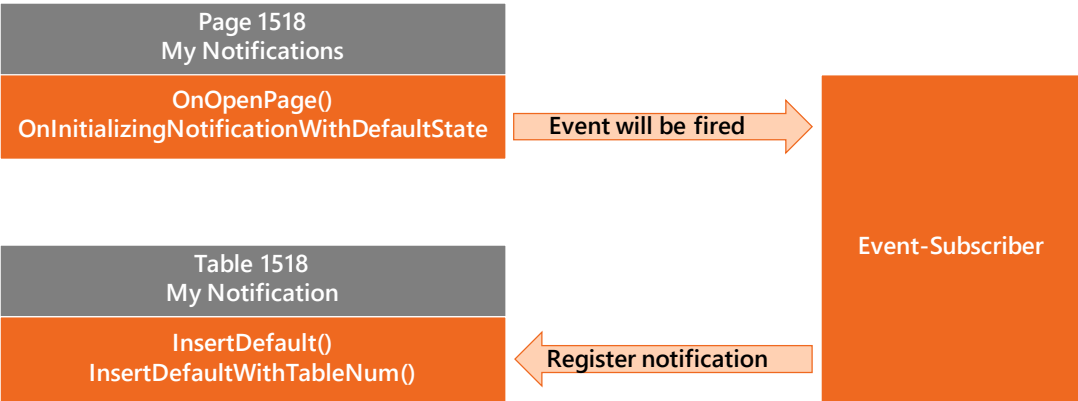
The screenshot shows a notification within the Microsoft Dynamics 365 Business Central interface. At the top, it says 'VERKAUFSAUFRAG | ARBEITSDATUM: 28.01.2021'. Below this is the title '101016 · Möbel-Meller KG'. A light blue notification bar contains the message: 'X Der verfügbare Lagerbestand für Artikel 70000 ist geringer als die eingegebene Menge an dieser Position.' with a 'Details anzeigen' link. Below the notification bar is a process bar with steps: 'Prozess', 'Freigeben', 'Buchen', 'Vorbereiten', 'Auftrag', 'Genehmigung anfordern', 'Drucken/Senden', 'Navigieren', and 'Weitere Optionen'. The 'Allgemein' tab is selected, showing fields for 'Debitorenname' (Möbel-Meller KG) and 'Fälligkeitsdatum' (28.02.2021). On the right, there are links for 'Details' and 'Anhänge (0)', and a section for 'Verkaufshistorie für Verkauf an ...'.

Personal settings

- Page My Notifications
 - From MenuSuite or My Settings Page in Web Client
- Personal settings to disable or filter Notifications
- Expandable list



Implementation – My Notifications



How to create a Notification

- New C/AL Types
 - Notification
 - NotificationScope
- Developers
 - Have full control over the context
 - Can include data
 - Can include actions
 - Can recall a notification
- User can personalize if developer adds Notification to My Notifications page

- AddAction
- GetData
- HasData
- Id
- Message
- Recall
- Scope
- Send
- SetData

Code example of a Notification

```
CreditLimitNotif.Id(GetCreditLimitNotifID());
CreditLimitNotif.Scope(NotificationScope::LocalScope);
CreditLimitNotif.Message(
    StrSubstNo(CreditLimitNotifMsg,
        Cust.FieldCaption("Balance (LCY)"), Cust."Balance (LCY)",
        Cust.TableCaption(), Cust."No.", Cust.Name,
        Cust.FieldCaption("Credit Limit (LCY)"));
CreditLimitNotif.SetData('CustNo', Cust."No.");
CreditLimitNotif.AddAction('Edit Customer', Codeunit::"Demo Cred. Limit Notification", 'OpenCustomerCard');
CreditLimitNotif.Send();
```

```
procedure OpenCustomerCard(Notif: Notification)
var
    Cust: Record Customer;
begin
    Cust.Get(Notif.GetData('CustNo'));
    Page.Run(Page::"Customer Card", Cust);
end;
```

Creating Notifications – Tips

- Always use a unique ID.
 - Can be any GUID.
 - Create a GUID online www.guidgen.com or in VS Code.
- AddAction() points to a function in a Codeunit.
 - Function must be global.
 - Function must accept a Notification as parameter.
- Use SetData() and GetData() to exchange information between the sending and handling function.

Exercise optional: Notification

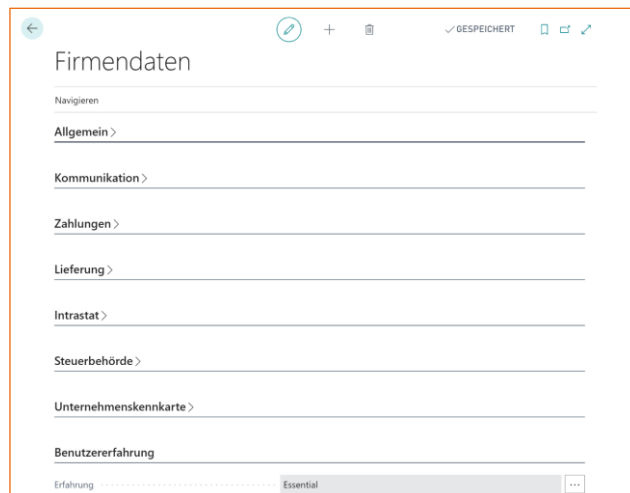
- Create a new codeunit 50131 **BSB Company Information Notif.**
- Implement a notification that, when opening a **Sales Order**, checks whether the **Company Name** and **E-Mail** address are entered in the **Company Information**.
If not, a corresponding notification is issued with an action to open the **Company Information Wizard**.
- Register the notification in **My Notifications**.
You can use the snippet rMyNotifications for this.

Application Areas

What are Application Areas?

- Designed to simplify the user experience.
- By hiding irrelevant controls from the user.
 - Page fields
 - Page actions
 - Report RequestPage options
- Enabled on Dynamics 365, cannot be disabled.
- Mandatory for Dynamics 365 Apps.

Setup Experience Tier in Company Information



Setting Application Areas

- Property has a comma separated text.
 - In C/SIDE, each Application Area is preceded by a hash tag.
- Only controls in the enabled Application Area will be displayed.
- Application Area "All" is always displayed.
- When no Application Area is set, nothing is displayed!

Scope

- Currently in development – only meant for Dynamics 365 Business Central
- You will be able to add your own Application Areas

Integrating with the outside world

Agenda

- HTTP API
- JSON API
- XML API

HTTP API

Lesson 1

About HTTP API

- Fills the gap for accessing HTTP-based web services without DotNet interop.
- Allows you to declare web request/response specific variables in AL.
- Objects are automatically initialized, there is no need to call constructors.
- AL-friendly syntax and usage.

HTTP API types

Type	Description
HttpClient	Allows invoking HTTP methods (GET, POST, etc.) against a URL.
HttpContent	Contains the data to be sent through an HttpClient with a method that supports sending content.
HttpHeaders	Contains headers for either the entire request or for content.
HttpRequestMessage	Represents a request message to be sent to a URL through an HttpClient.
HttpResponseMessage	Represents a response returned by a web server in response to a call placed through an HttpClient.

These types are direct replacements for .NET types of the same name, from the System.Net.Http namespace.

Read more at: <https://msdn.microsoft.com/en-us/library/system.net.http>

JSON API

About JSON API

- Fills the gap for building, navigating, searching, or manipulating JSON without DotNet interop.
- Allows you to declare JSON variables directly in AL.
- Allows constructing JSON from string, or converting JSON to string.
- Can be used together with HTTP API in conjunction with REST web services.
- JSON types can be used as return values from functions.

JSON API types

Type	Description
JsonObject	Represents a JSON object, which is a collection of JSON key-value pairs.
JsonArray	Represents a JSON array, which is a collection of JSON objects.
JsonToken	Represents any syntactical element of JSON, and can be an object, an array, or a value.
JsonValue	Represents a value of a JSON key-value pair.

- These types are direct replacements for .NET types of the similar name (J instead of Json, so JObject, JArray...), from Newtonsoft.Json.Linq namespace, from the Newtonsoft.Json.NET library, that is a part of NAV stack since version 2013 R2.
- Read more at: <https://www.newtonsoft.com/json>

Constructing JSON

```
JObj.Add('text', 'Hello, World!');  
JObj.Add('number', 3.1415926);  
JObj.Add('datetime', DateTime.Now);  
JObj.Add('boolean', false);  
JArr.Add(JObj);  
JArr.WriteTo(TextWriter);  
Message(TextWriter);
```



[{"text":"Hello, World!","number":3.1415926,"datetime":"2020-01-06T00:53:07.4690000Z","boolean":false}]

OK

XML API

Lesson 3

About XML API

- Fills the gap for handling and manipulating XML without DotNet interop.
- Allows you to declare XML variables directly in AL.
- Can be used together with HTTP API in conjunction with XML-based REST web services, or SOAP web services.
- XML types can be used as return values from functions

XML API types

Type	Description
XmlAttribute	Represents an XML attribute.
XmlNode	Represents any XML node.
XmlElement	Represents an XML element.
XmlDocument	Represents an XML document.
XmlNamespaceManager	Allows managing namespaces for an XML document.
... a lot more	

These types are direct replacements for .NET types of the same name, from the System.Xml namespace.

Read more at: <https://msdn.microsoft.com/en-us/library/system.xml>

Source Control Management

Agenda

- Overview of source control management.
- Setting up Git support.
- Managing repositories.
- Working with code.
- Working with branches.
- Synchronizing changes.
- Merging conflicts.

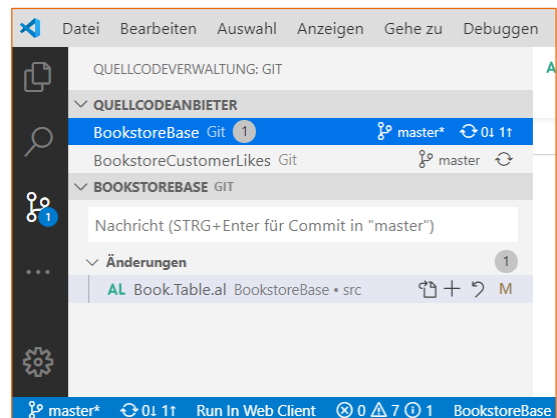
Overview

Source Control Management in VS Code

- Integrated feature of Visual Studio Code.
- Native support of Git.
- SCM providers available for many other technologies:
 - VSTS
 - Mercurial
 - SVN
 - Perforce

SCM Providers

- Supported through extensions.
- Multiple providers can be used at the same time.
- VS Code provides insight into which provider is used, and easy way of switching between different providers.



Git support

- VS Code supports invoking base git commands from within its UI.
 - Commands are always forwarded to local OS.
 - To execute commands from Code:
 - Git support must be installed on the machine.
 - Git must be supported on command line.
- Until Git is installed on the machine, VS Code cannot do anything, even with a git-based SCM provider (such as VSTS).

Setting up Git Support

Installing Git support

- Install a third-party Git tool for command line.
- Make Git available on command line through PATH (done automatically with Windows installer).
- Configure Git tools on your machine by providing username and e-mail configuration options.
- Install Git from <https://git-scm.com/download/win>.

Exercise: Installing Git support

- Download installation from <https://git-scm.com/download>.
- Start installation.
- In the Choosing HTTPS transport backend page, select the Use the native Windows Secure Channel library option.
- In the Configuring the terminal emulator to use with Git bash page, select the Use Windows default console window option.

Managing repositories

After Git support is installed

- Git: commands in command palette are working.
- Commands are populated in the More button in the Source Control pane.
- For this to work, a repository must be initialized.
 - Having Git installed on machine is not enough.
 - You must have a Git-aware workspace.

Some Git terminology

Term	Meaning
Repository	A project folder that contains all files that belong to an individual project.
Remote	A version of a project that is hosted on a remote server, that can then be synced with a local repository.
Clone	A local copy of a remote repository that allows you to work offline.
Push	Sending committed changes from local repository to remote.
Pull	Retrieving the changes from remote repository and merging them to local repository.
Fetch	Same as pull, without merging – this can be done manually later.
Branch	A version of repository different than master, allowing you to work without disrupting the "live" version.
Merge	Applying changes done in one branch onto another branch, most commonly master.

Initializing a Git repository

There are two ways of initializing a local Git repository:

- Clone an existing Git repository from a remote Git source (GitHub, VSTS, some other...).
- Initialize a repository, then push it to a remote repository.

Clone a remote repository

- Open VS Code.
- Access Command Palette.
- Enter "Git: Clone".
- Enter the path of Git remote (for example: <https://github.com/vjekob/test1.git>).
- Enter the path of the local parent directory where the child directory for the cloned repository will be created.
- When prompted to open the cloned repository, click Open Repository.

Creating and linking

- Create a local folder.
- Open VS Code.
- Access Command Palette, and run "Git: Initialize Repository".
- Browse to the folder created in the first step and click Initialize Repository.
- Open that folder in VS Code.
- In Terminal window, enter the following command:
`git remote add origin url_of_remote_repository.`
- Synchronize.

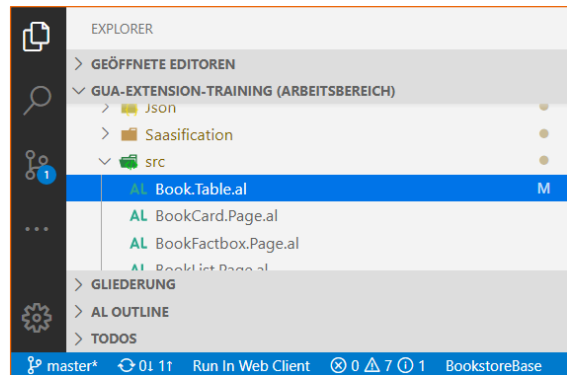
Which way to choose?

- Clone
 - There is an existing repository.
 - You are configuring a new environment to connect to an existing project.
 - You want to fork an existing project that you want to spin off.
- Initialize locally first
 - You are starting a new project.
 - You don't want to work with a remote, and want to work in local-only mode.
 - You have local project not under SCM, that you want to start managing now.
- Keep in mind: You can initialize a local repository over an already existing folder with files!

Working with code

Working with code (1)

- Git automatically tracks changes and shows information about changes and status on screen.
- Indicators show:
 - Number of changed files.
 - Current branch.
 - "Dirtiness" status of current branch.
 - Number of outgoing pushes and incoming pulls.



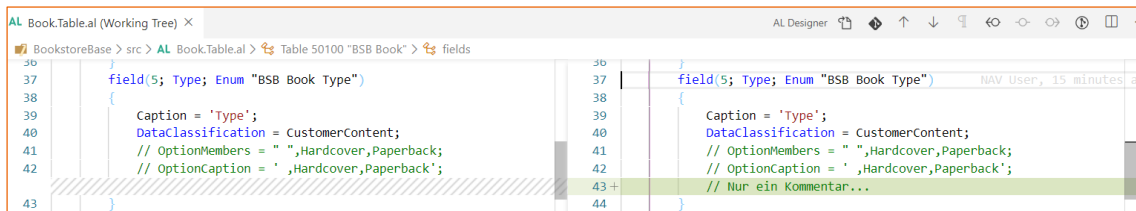
Working with code (2)

- Gutter indicators in the editor indicate where the changes were made in the file (blue blocks in the scroll bar).



Working with code (3)

- Changes can be viewed for each modified file.
- Changes can be edited for modified files directly in the comparison editor.



Committing code to repository

- Many Git actions require repository to not be dirty.
- After work on code is done, changes should either be:
 - Staged: to prepare them for committing, but not yet fully commit them to a repository. Staged changes can still be unstaged and re-edited.
 - Committed: fully committed to a local repository. Committed changes cannot be directly edited (only through adding more changes on top of them).
- Both staging and committing are local.
- To propagate changes to remote, use push or sync.

Working with branches

Working with branches

Command Palette command	Description
Git: Create Branch...	Create a new branch from the current branch.
Git: Delete Branch...	Delete a target branch (it can also be current, but choice is made about what to delete before actually deleting anything).
Git: Merge Branch...	Merges another branch onto a current branch.
Git: Publish Branch...	Publishes current branch onto a specified remote (there can be multiple remotes).

Synchronize changes and merging conflicts

Synchronizing changes

Action	Description
Synchronize changes	<p>Pull all pending incoming pulls, push all pending outgoing pushes. Merge is performed in the same operation.</p> <p>To perform:</p> <ul style="list-style-type: none">• Git: Sync• Click the Sync action in the status bar.• Invoke Sync from the More menu in Source Control pane.
Push	<p>Push pending outgoing pushes to a remote.</p> <p>To perform:</p> <ul style="list-style-type: none">• Git: Push• Git: Push to...• Push / Push to... in the More menu in Source Control pane.
Pull	<p>Pull all pending incoming pulls from a remote.</p> <p>To perform:</p> <p>Git: Pull / Git: Pull from... (or similar from the More menu).</p>

Merging Conflicts

- Changes are merged automatically.
- Changes on the same source file are indicated as conflicts.
- Conflicts indicators are injected into the source file, and have to be resolved before the file is usable again.

```
Aktuelle Änderung akzeptieren | Eingehende Änderung akzeptieren | Beide Änderungen akzeptieren | Änderungen vergleichen
<<<<<< HEAD (Aktuelle Änderung)
// OptionMembers = " ",Hardcover,Paperback;
// OptionCaption = ' ',Hardcover,Paperback';
// Nur ein Kommentar...
=====
OptionMembers = " ",Hardcover,Paperback;
OptionCaption = ' ',Hardcover,Paperback';
// Ein weiterer Kommentar
>>>>>> aacf5cabb4a8c68652aae0bd030d4c362f7a82d2 (Eingehende Änderung)
```

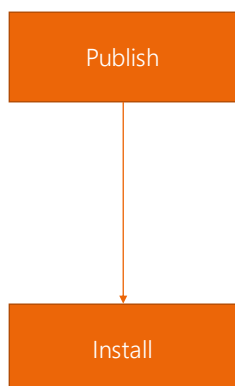
Deploying, Managing and Upgrading the Extension

Scenarios

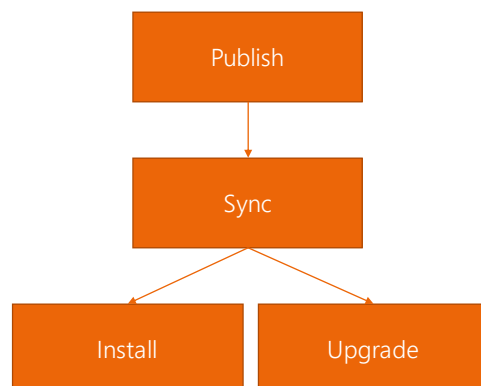
- We will cover:
 - Publish and install an Extension for the first time.
 - Upgrade an Extension.
 - Remove an Extension.
- We won't cover:
 - Joint upgrades (upgrades of multiple extensions at the same time).

Available steps

- Extensions V1

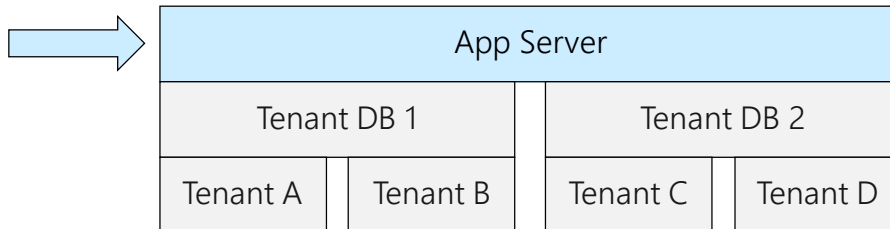


- Extensions V2



1 – Publish-NAVApp

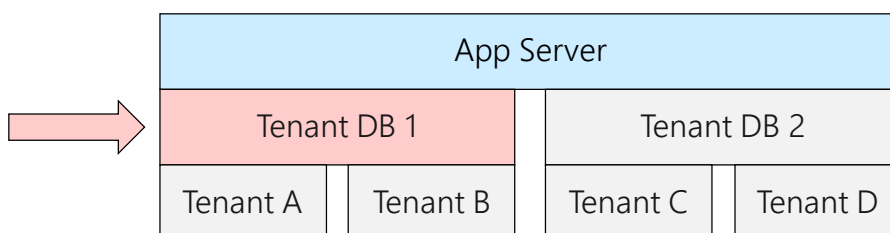
- Publish your .app to a certain ServerInstance.



- Note: Unpublish with Unpublish-NAVApp.

2 – Sync-NAVApp

- Applies schema changes on database level to the tenant.
- Database is up-to-date, but app won't use it.



- Note: you can use the Sync-NAVApp also to "clean up" the data (Mode: "Clean").

Extensions V2 Schema (from BC23)

- The Sync will create companion tables for all extensions that changed the schema of default tables.

Customer\$437dbf0e-84ff-417a-965d-ed2bb9650972

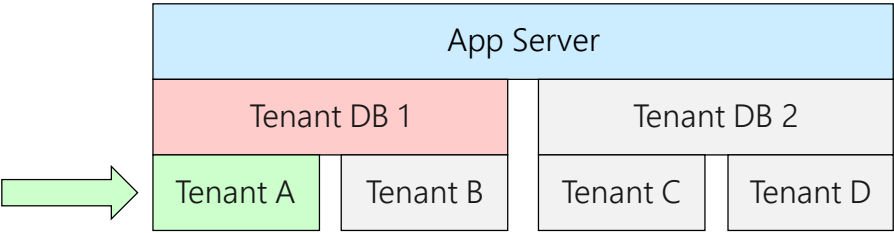
No_	Name
-----	------

Customer\$437dbf0e-84ff-417a-965d-ed2bb9650972\$ext

No_	Shoe_Size\$2e454762-...	Hair_Color\$9d8ed184-...
-----	-------------------------	--------------------------

3 – Install-NAVApp

- Applies at tenant level.
- Prepares data in companion table.
- Executes Install Code.



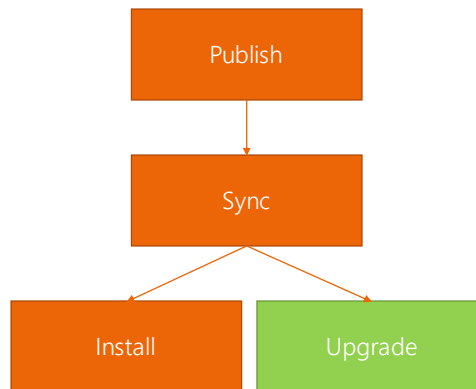
- Note: Not possible to run when there is data from an extension with a lower version.

4 – Start-NAVAppDataUpgrade

- Performs data upgrade for the app.
- Only when there is data from a previously installed version of the app.
- Runs the upgrade-logic that is defined by the upgrade codeunits in the extension (optional).
- Uninstalls the current extension version, and enables the new version instead.

Publish and install an extension

Steps



1. Publish

- Makes the app available on the server.

2. Sync

- Applies schema changes on database level.

3. Install

- Package data
- Web services
- Permission sets
- Report layouts
- Wiring up the metadata.

- Ensuring companion table integrity on install.
- Leaving data in actual tables on uninstall.
- Authoring install code.

3. Install - Code

- What do I need to know?
 - It lives in install codeunits.
 - It is optional.
 - It uses the same APIs as upgrade code to query state.
- Two possible cases
 - Fresh Install = no previous install/data (data version = 0.0.0.0).
 - Reinstall = previous install/data (data version = actual version).

3. Install - Code

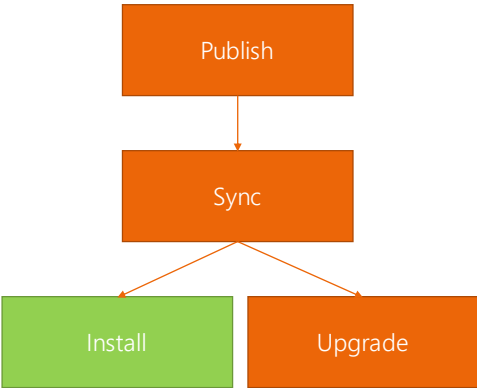
```
codeunit 50100 MyInstallCodeunit
{
    Subtype = Install;

    trigger OnInstallAppPerCompany();
    begin
        // your logic goes here
    end;

    trigger OnInstallAppPerDatabase();
    begin
        // your logic goes here
    end;
}
```

Upgrade an extension

Steps



1. Publish

- Makes the app available on the server.
- This app has a newer version than the previously installed app.
- If version is not different than a published app, the publish process ends in error.

2. Sync

- Applies schema changes on database level.
- But now, there is data, and possibly schema changes.

2. Sync – Principles

- Versioning
 - First synced version = baseline
 - Highest synced version = current maximum
 - Installable range = [Baseline, Maximum]
 - Sync can only be done against newer versions.
- Compatibility
 - A new version must be backwards compatible with each version back to the baseline.
 - No destructive changes are supported (for now).

2. Sync – Rules

- At Sync Time:
 - Changes are computed between the current and target versions.
 - The changes are evaluated against the rules.
 - If any rules are violated, an error is raised and the sync is aborted.

2. Sync – Rules

- Supported Changes
 - Adding fields/keys.
- Unsupported Changes
 - Removing fields/keys.
 - Renaming fields/keys.
 - Changing datatypes.
 - Changing key order.
 - ...Pretty much everything not on the supported list.

2. Sync – Modes

- Add
 - Processes and applies non-destructive changes.
 - Leverages the validation rules to check for unsupported changes.
 - The correct mode to use for production scenarios.
- Clean
 - Removes ALL schema for a given extension (all versions).
 - A mode to consider in dev/test scenarios.

3. Upgrade

- Cmdlet: Start-NAVAppDataUpgrade
- Performs data upgrade for the app.
- ONLY for upgrading app from old to new version.
- Automated restore.
- AL APIs for contextual info.
- Joint/Combined upgrade possible (Dynamics 365 – not covered today).

3. Upgrade – Principles

- Automatic Data Restore.
- No data changes need between versions = No data work needed.
- Upgrade - Executed within a single SQL transaction.
- Execution Context & Data Versions.
- Using the new API's to determine the context of running code and data version.

3. Upgrade – Code Hooks

- OnCheckPreconditions
 - Verify state to determine if upgrade should execute.
 - Errors thrown here will prevent the upgrade from running.
- OnUpgrade
 - Perform the bulk of the upgrade work modifying data for the new version
- OnValidate
 - Validate any state necessary to assert that the upgrade was successful
 - Errors thrown here will rollback the upgrade transaction

3. Upgrade – Code

```
codeunit 50100 MyUpgradeCodeunit
{
    Subtype = Upgrade;

    trigger OnCheckPreconditionsPerCompany(); begin end;
    trigger OnCheckPreconditionsPerDatabase(); begin end;
    trigger OnUpgradePerCompany (); begin end;
    trigger OnUpgradePerDatabase(); begin end;
    trigger OnValidateUpgradePerCompany (); begin end;
    trigger OnValidateUpgradePerDatabase(); begin end;
}
```

What are these API's all about?

Module API

What is it?

- New API that can provide introspective information about the Base & Extensions from within AL.

Execution Context

- The mode in which a given module is executing in, or the operation being perform within the current session.

Module Info

- Information about a specific module such as Name, Publisher, or Current Data Version.

Module API – Execution Context

Option Values

- Normal – Code is executing under standard circumstances.
- Install – Code is executing in the context of an install.
- Upgrade – Code is executing in the context of an upgrade.

Usage

- Get the general execution context of the current session:
`SESSION.ExecutionContext`
- Get the context of the currently executing module:
`SESSION.GetCurrentModuleExecutionContext`
- Get the context of a specific module:
`SESSION.GetModuleExecutionContext(ModuleID : GUID)`

Module API – ModuleInfo Type

Type Members

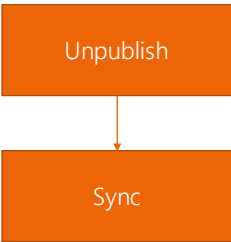
Property	Description
Id	The GUID of the module.
Name	The name of the module.
Publisher	The publisher of the module.
App Version	The current version of the module.
Data Version	The current version of the stored data for this module.
Dependencies	List of the direct dependencies of the module.

Usage

- Get the info of the currently executing module:
`NAVAPP.GetCurrentModuleInfo (var info : ModuleInfo)`
- Get the info of a specific module
`NAVAPP.GetModuleInfo(Module ID : GUID, var info : ModuleInfo)`

Remove an extension

Steps



Testing Codeunits

Agenda

- Identify testing areas
- Import Test Toolkit
- Set up test project
- Writing the test
- Run the test

Identify testing areas

Identify testing areas

- Cover all setup and usage scenarios as described in the user scenario document.
- Use the CRONUS demo company.
- Include additional data in the test.
- Include tests to verify that the extension works for a non-SUPER user.
- No requests to external services.
 - Mock your external calls to prevent this.

Bookshelf testing app

- Customer Card page – verify that the page has the field “Favorite Book No.”.
- Book List page – verify that the page behaves as expected.
- Book Card page – verify that the page behaves as expected.

Import Test Toolkit

Why using the Test Toolkit

- The Test Toolkit is used to automate and run the test.
- The Test Toolkit includes:
 - Codeunits with test functions.
 - Codeunits with generic and application-specific functions.
 - Application objects for running application tests.

Test Toolkit

- The Test Toolkit Test Libraries consists of 5 apps, which are included on the latest Docker images in the C:\Applications folder:
 - Microsoft_Any.app
 - Microsoft_Library Assert.app
 - Microsoft_System Application Test Library.app
 - Microsoft_Tests-TestLibraries.app
 - Microsoft_Test Runner.app
- The source for these applications are also included in .zip files.

Import the Test Toolkit

- PowerShell module navcontainerhelper has a function for this.
 - Run **Import-TestToolkitToBCContainer** function.
 - Alternatively use the option **-includeTestToolkit** with the **New-BCContainer** function.
 - For both above: Unless you specify **-includeTestLibrariesOnly**, importing the test toolkit will include all tests. This will take some time.

Set up Test Project

Set up a Test Project

- Create a new project for the tests.
- Specify the dependency between the test project and the project that you want to test.
- In launch.json set "startupObjectId": 130451.
- In app.json, add dependencies on the Test Framework apps (see next slide).
- Download the symbols.
- See also: <https://freddysblog.com/2019/08/30/running-tests-in-15-x-insider-containers/>

Test Toolkit Dependencies for the Test Projekt (1)

```
{
  "appId": "dd0be2ea-f733-4d65-bb34-a28f4624fb14",
  "publisher": "Microsoft",
  "name": "Library Assert",
  "version": "15.0.0.0"
},
{
  "appId": "e7320ebb-08b3-4406-b1ec-b4927d3e280b",
  "publisher": "Microsoft",
  "name": "Any",
  "version": "15.0.0.0"
},
```

Test Toolkit Dependencies for the Test Projekt (2)

```
{  
  "appId": "9856ae4f-d1a7-46ef-89bb-6ef056398228",  
  "publisher": "Microsoft",  
  "name": "System Application Test Library",  
  "version": "15.0.0.0"  
},  
{  
  "appId": "5d86850b-0d76-4eca-bd7b-951ad998e997",  
  "publisher": "Microsoft",  
  "name": "Tests-TestLibraries",  
  "version": "15.0.0.0"  
}
```

Writing the test

Writing the test – using tags

- Use tags to describe the tests.
 - In comment lines.
- [FEATURE] [<tag1>] [<tagn>]
 - Name of the feature, application area, can be set for whole codeunit.
- [SCENARIO <id>] <test description>
 - Link to work item.
 - Test description contains short description of test.

Writing the test – using tags

GIVEN – WHEN – THEN tags

- GIVEN – describes one step in setting up the test.
- WHEN – describes the action under test.
- THEN – the verification.

Test Codeunit

- Codeunit must have Subtype property set to Test.
- Test methods decorated with [Test].
- OnRun is executed first, then all test methods.
- Each test method is in a separate database transaction.
 - Optionally use the [TransactionModel] attribute.
- The outcome is SUCCESS or FAILURE.
 - If any error is raised by the test code or the code that is being tested, then the outcome is FAILURE.
 - Other Test methods will still continue.

Extra features

- Test pages
 - Mimic actual pages.
 - Not UI on a client computer.
 - Simulate user interaction.
- UI handlers
 - Handle user interaction.
 - MessageHandler, ConfirmHandler, StrMenuHandler, PageHandler, ModalPageHandler, ReportHandler, RequestPageHandler
 - Unhandled UI causes failure.

AssertError statement

- Test code under successful and failure conditions.
- AssertError is used to test under failing conditions.
- Indicates that an error is expected.

Run the test

Steps to run the test

- Publish the test app.
- Open the **AL Test tool** (Page 130451).
- Choose Get Test Codeunits and then Select Test Codeunits.
- Select the test codeunit and click OK.
- Choose Run or Run Selected.

Closing

How to stay up-to-date

Description	URL
Follow the Getting Started article	https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-get-started
Read the Developer Reference	https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/
Read and contribute on GitHub	https://github.com/microsoft/al/issues

Development Resources

Description	URL
Dev tools bugs and suggestions	https://github.com/microsoft/AL
Event requests, application enhancements	https://github.com/microsoft/ALAppExtensions
Business Central Learning Videos	https://aka.ms/businesscentralvideos
AL Language change log	https://marketplace.visualstudio.com/items/ms-dynamics-smb.al/changelog

Vielen Dank!

www.getanduse.academy

