NAV TECH DAYS 2016

mibuso.com

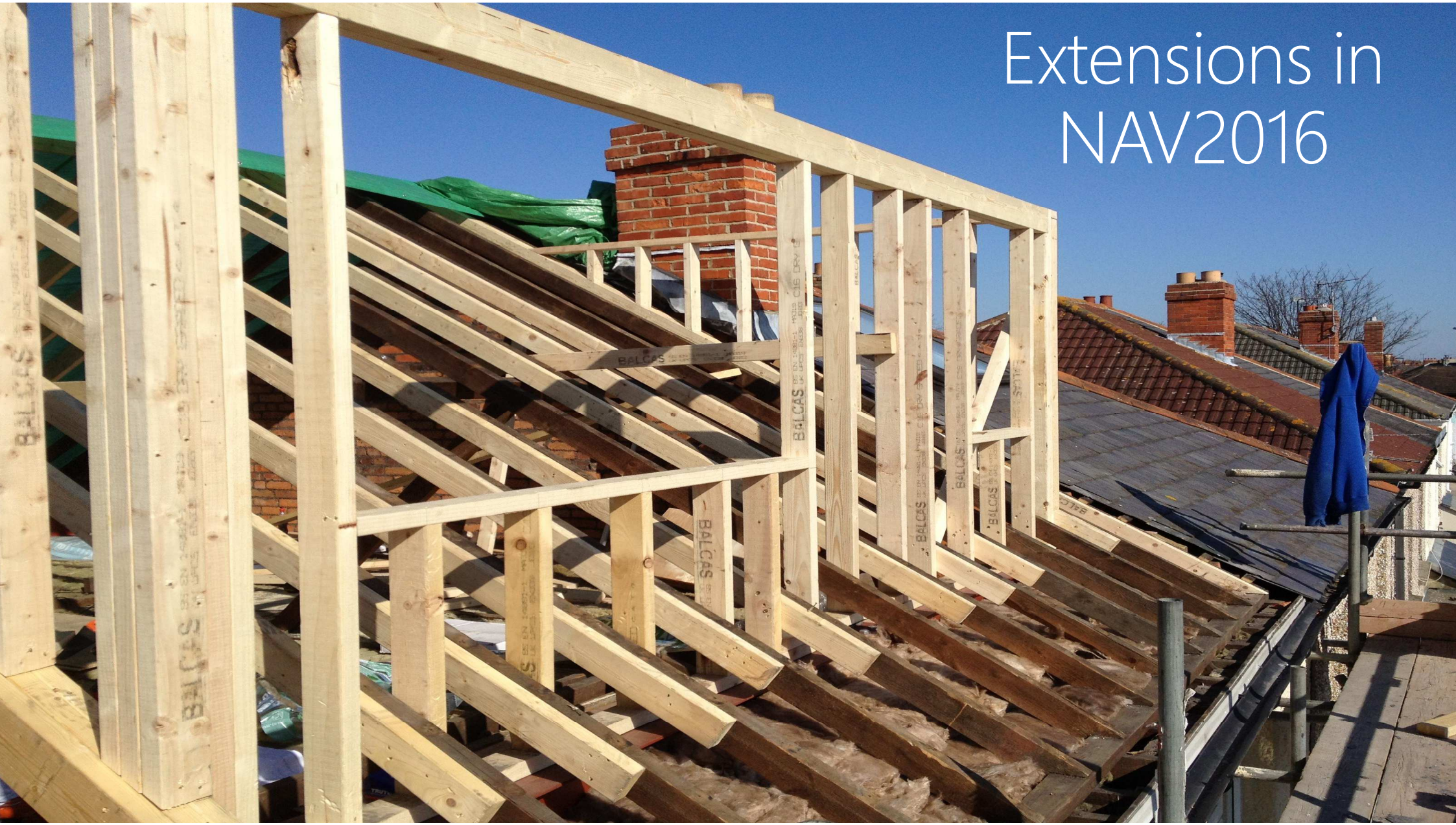
BEST PRACTICES IN DEVELOPING MICROSOFT DYNAMICS NAV 2017 EXTENSIONS

ERIC WAUTERS (WALDO)
IFACTO BUSINESS SOLUTIONS
CLOUD READY SOFTWARE


If you're happy & you know it
clap your... oh

Session objectives

Extensions in NAV2016



Extensions in NAV2016

- NO XMLPorts
- NO Queries
- NO Reports
- NO Add-Ins
- NO Web Services
- …

- Limited set of Object Types
  - Codeunits
  - Tables
  - Pages
- Changes to
  - Tables
  - Pages

NAV TECH DAYS 2016
mibuso.com

What's new in NAV2017

In a nutshell …

NAV 2017 Extension: New capabilities

NAV TECH DAYS 2016
mibuso.com

Additional Object Types:

- Reports
- XMLPorts
- Queries
- Custom report templates
- Reports

Default and starting data

Multilanguage captions

Support for .Net Framework Add-Ins

- .NET interop types executed on the server
- Client-side JavaScript
- WinForms extensibility control add-ins

Support for web services

Restore and backup data in extensions

- NAVAPP.RESTOREARCHIVEDATA
- NAVAPP.DELETEARCHIVEDATA

Updated PowerShell to publish Extensions

- Publish to SQL db on Azure SQL

Installing Extensions

- Users can install/uninstall from Extensions Mgt page.


Extension
Management

PowerShell

PowerShell

Create your own set of functions

Turn them into a Module

Make sure you use the same module on all environments

On top of this module, create script to maintain your Extension

Make these scripts part of your app

PowerShell

Set of scripts that "maintains" the extension/app:

- _Settings file
- Create DEV Environment
  - Auto naming of environments
  - Shared test instance
  - Shared orig instance
- Apply Deltas
  - Import deltas in the DEV environment
- Create PermissionSet
- Open DEV Environment
- Build and Deploy
- Backup App

PowerShell

Set of scripts that "maintains" the extension/app:

- _Settings file
- Create DEV Environment
  - Auto naming of environments
  - Shared test instance
  - Shared original instance
- Apply Deltas
  - Import delta in the DEV environment
- Create PermissionSet
- Open DEV Environment
- Build and Deploy
- Backup App

PowerShell – Create DEV Environment

Original
- Needed to create deltas

DEV database
- Development

TEST database
- Always test as an extension

PowerShell – Apply deltas

Update your DEV db with your latest developments by importing deltas
Agnostic to

- DB version
- CU Update
- …

Can also handle deletes

It's dangerous to work with Fob or Txt

PowerShell – Permission Sets

Very necessary
Export-NAVAppPermissionSet
Data

Incorporate it in a script
(same for webservices)

PowerShell – Build And Deploy

Never expect your Extension to work as normal development
ALWAYS test an Extension AS an Extension

Steps

- Create a NAVX file
  - Increment Build
  - Deltas
  - Permission sets
  - Create AddIn
  - …
- If it's installed on the TEST-environment: uninstall it
- Publish / install in test
- Open TEST

EXTENSION PUBLISHING – V1

PHASE 0: SANDBOX

- Create new DB containing copies of app DB tables
- Used for temporary import/export/compile steps

PHASE 1: BASE + DEPENDENCIES

PHASE 1A: IMPORT SOURCE

- Changed objects
  - Export [BASE]
  - Apply AMU deltas [DEPS]
  - Import merged [BASE + DEPS]
- New objects [DEPS]

PHASE 1B: COMPILE

PHASE 1C: EXPORT RUNTIME METADATA (ORIG)

- XML [BASE + DEPS]
- C# [BASE + DEPS]
- AL [BASE + DEPS]

PHASE 2: BASE + DEPENDENCIES + EXTENSION

PHASE 2A: IMPORT SOURCE

- Changed objects
  - Export [BASE + DEPS]
  - Apply AMU deltas [EXTENSION]
  - Import merged [BASE + DEPS + EXTENSION]
- New objects [EXTENSION]

PHASE 2B: COMPILE

PHASE 2C: EXPORT RUNTIME METADATA (MODIFIED)

- XML [BASE + DEPS + EXTENSION]
- C# [BASE + DEPS + EXTENSION]
- AL [BASE + DEPS + EXTENSION]

PHASE 3: FINAL METADATA

PHASE 3A: COMPARE RUNTIME METADATA (ORIG vs MODIFIED)

- Diff changed objects
  - XML -> *.xmldelta
  - C# -> *.csdelta
- Copy new objects
  - XML
  - C#
  - AL

PHASE 3B: COMMIT

- Put final metadata in the "real" database

PowerShell – Backup

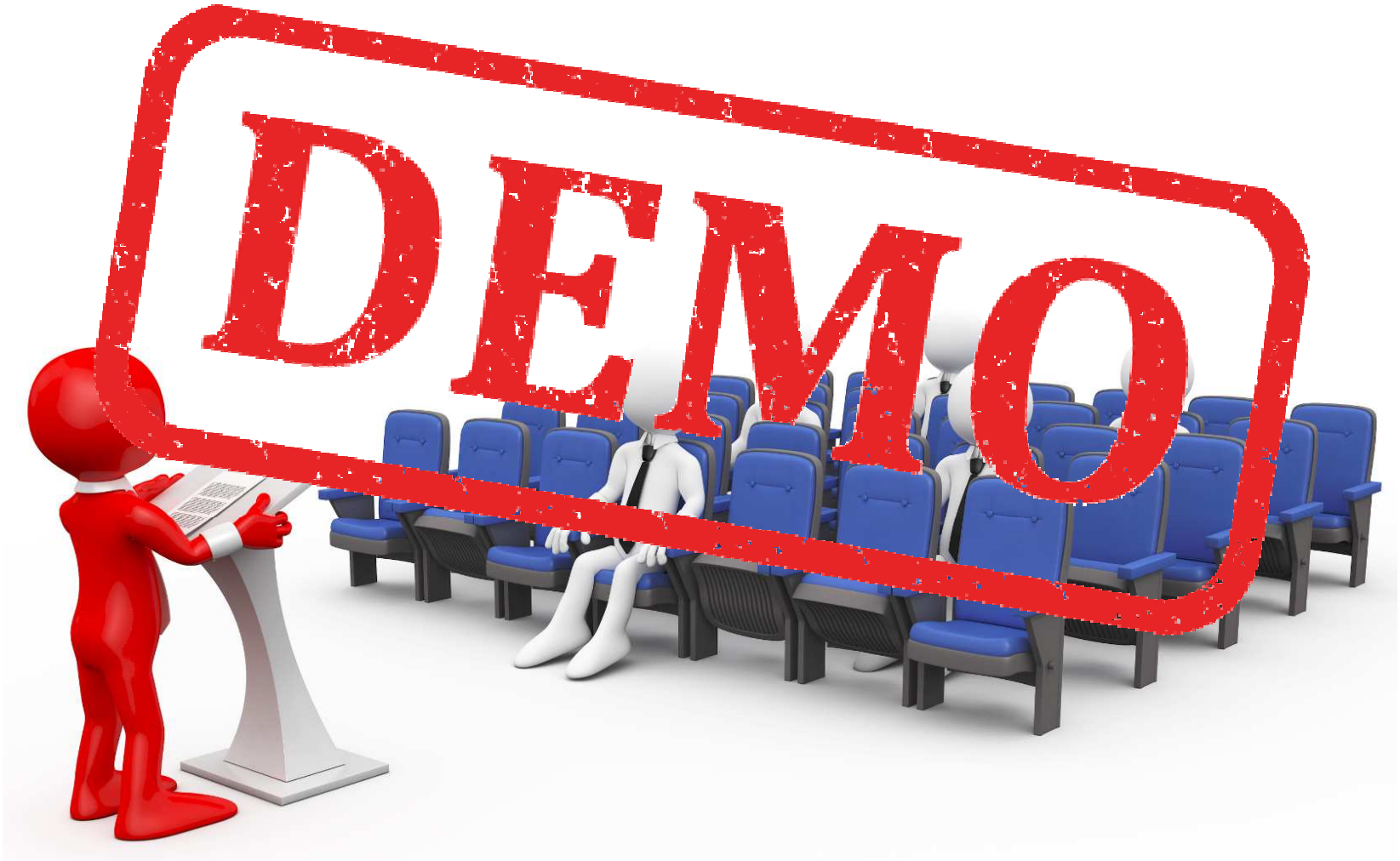
Structured way to "save" developments

- Enables SCM

Backup as much as might be useful:

- txt
- Split txt
- fob
- navx
- Deltas
- Reversed deltas
- manifest

WaldoNAVPad



PowerShell

You don't have to start from scratch …

GitHub (https://github.com/waldo1001/)

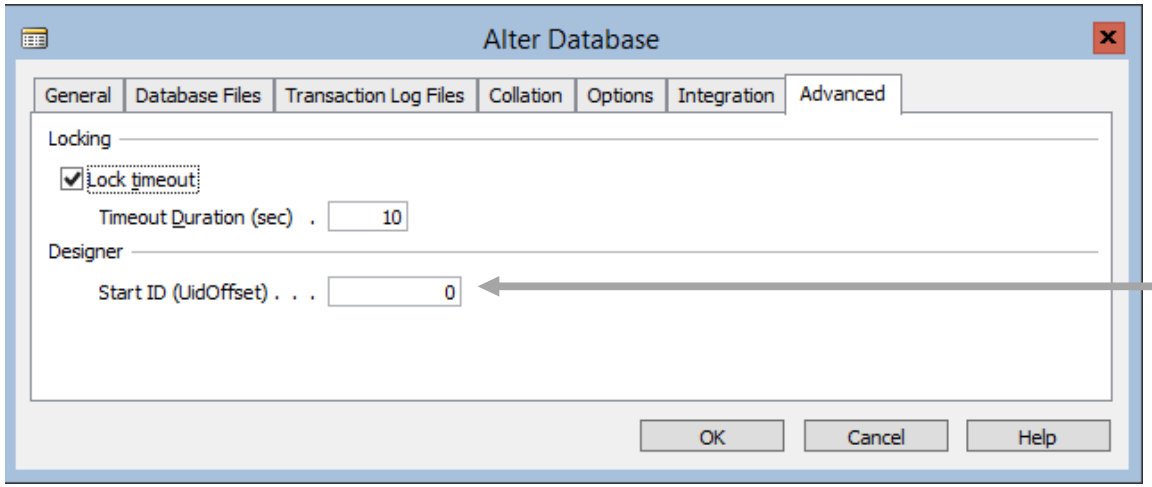
PowerShell Scripts:
https://github.com/waldo1001/**Cloud.Ready.Software.PowerShell**
WaldoNAVPad
https://github.com/waldo1001/**Waldo.NAV**

Development

Start with the basics

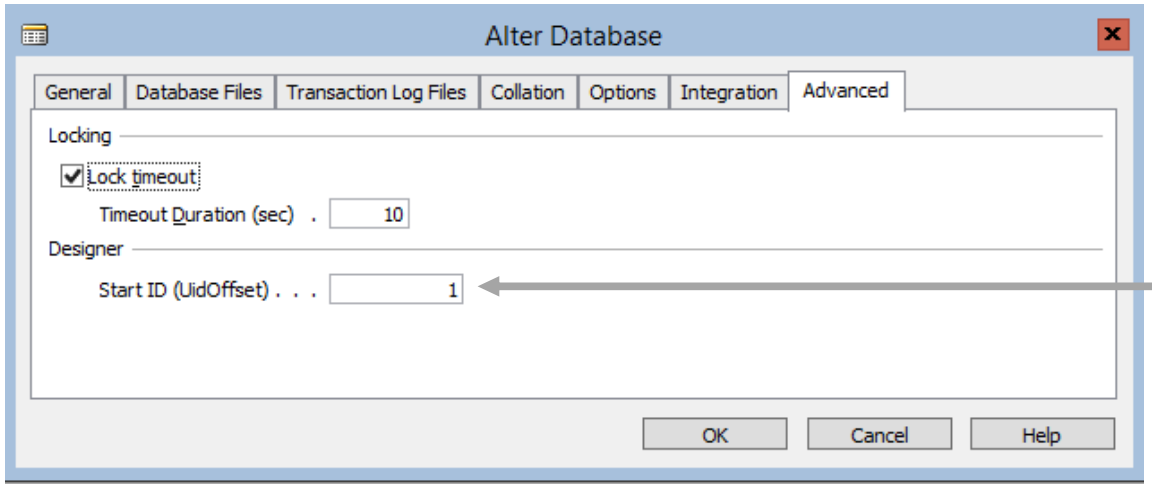
1. PowerShell scripts & settings
2. Build DEV environment
3. Import the basics:
   - Upgrade codeunit
   - Helper codeunits
   - Only in your numberrange!
   - Unique naming!
4. UidOffSet

Start ID (UidOffSet)



This Is NOT good!

Start ID (UidOffSet)



This Is NOT good either!

Start ID (UidOffSet)

When you try to publish an Extension that has conflicting ControlIDs:

- `Publish-NAVApp : Merge conflict found for object Page:22, Missing base object to apply modification, or duplicate new objects found`
- `At line:1 char:1`
- `...`

Start ID (UidOffSet)

Suppose the Extension was already published,
but at a later point, an action (with same ID) was added using normal development.
Now you try to install the Extension:

- `Install-NAVApp : An error occurred while applying changes from the 'Just4Test by Cloud Ready Software GmbH 1.0.0.16' app to the application`
- `object of type 'Page' with the ID '22'. The error was: The element '<Actions ID="1100084000" />' can only be added once in this context.`
- `At line:1 char:118`
- `+ ...`

Start ID (UidOffSet)

Suppose the Extension was already installed,
people were using it, but you now you add an action (with the same ID) using normal development:

All compiles in DEV!
All seems to work in RTC!

But the new action does not show ☹

Start ID (UidOffSet)

Suppose you come across that situation – you might want to uninstall the Extension.

- `unInstall-NAVApp : An error occurred while applying changes from the 'Just4Test by Cloud Ready Software GmbH 1.0.0.16' app to the application object of type 'Page' with the ID '22'. The error was:`
- `The element '<Actions ID="1100084000" />' can only be added once in this context.`
- `At line:1 char:118`
- `+ ..`

Start ID (UidOffSet) - Conclusion

Take care of the UidOffSet

from Day 1

Hour 1

Minute 1

Second 1

Millisecond 1

...

Include it in your PowerShell scripts

Manifests

App ID is really important
Make it part of your SCM
Include it in your PowerShell scripts

Dependencies vs Prerequisites

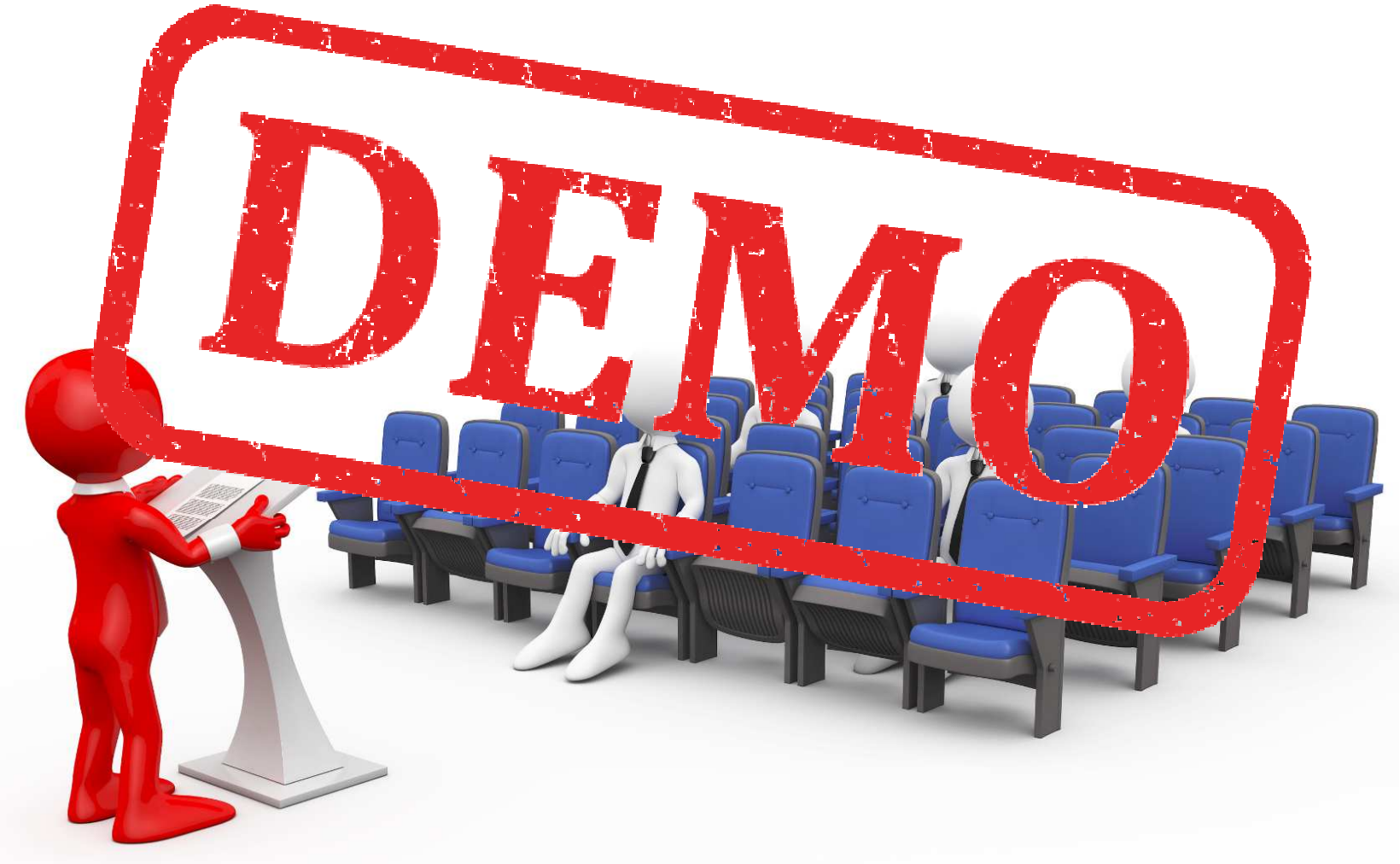
Prerequisites:

- a collection of **Object elements** that define what must be presently available in the hosting Microsoft Dynamics NAV deployment in order for this extension to function

Dependencies:

- a collection of Dependency elements that identifies **other extensions** that this extension has a specific dependency on

Extension Development Demo



Some other GOTCHA 's

CurrPage.SETSELECTIONFILTER
Never change 20000…-tables
No Documentation Trigger allowed
No code on default objects
- No code on new actions on default pages
- No new functions in default tables

Only new Objects in the ISV range!
- For Dynamics 365: 70-million range!
- Mind the UIDOffSet

Naming Conventions:
- Don't use too generic names – might conflict with other Extensions
- Work with Prefixes specific for your Extension

Menusuites
- Adds surprises

Don't change Option Values
No processing Reports

Missing Events - Approach

Different approach for:

Extensions for Dynamics 365	Extensions for your own product
In the Microsoft Cloud	On Premise Managed Service Your own cloud Infrastructure
Microsoft owns the base product	You own the base product
You're dependent from Microsoft's events	You can create any event you like

Missing Events

NAV TECH DAYS 2016
mibuso.com

e.g.: Converting Quote to Orders

How can you work around it:
Page Events
- No "Sender" (so, no CurrPage)
- No GlobalVAR

Context of Events
→ work with CallStack (James Pearson)

Missing Events - Suggestions

If you have suggestions for Microsoft:
http://connect.microsoft.com/dynamicssuggestions

Follow steps here:
https://blogs.msdn.microsoft.com/nav/2015/10/15/integration-events-in-microsoft-dynamics-nav-2016/

Events – considerations

Multiple Subscribers

Erroring events

- not always a compile error
- not always a runtime error
- It usually just ignores the event

Event Helper

Monitor

- the **number of subscriptions** on one publisher
- Monitor the **errors**
  - An issue in a subscriber might mean that a subscriber is ignored, while no error is being showed! → expected business logic is not executed!
- Uncompiled Objects (as a bonus)

On each login

Different behavior per db type

Event Helper

Monitor

- the **number of subscriptions** on one publisher
- Monitor the **errors**
  - An issue in a subscriber might be that a subscriber is ignored, while no error is being shown! – expected business logic not executed!
- Uncompiled Objects (as obvious)

On each login

Different behavior per db type

Debug Extensions

The upgrade process
An error (breaking execution)
A specific part in code (setting breakpoints)
Code Coverage

Debug Extensions

How?

- Manually set Break Points
- Debug Next → Break
- Break on Error

Debug Extensions

NAV TECH DAYS 2016
mibuso.com

How?

- Manually set Break Points
- Debug Next
- Break o Err

Code signing

Publish-NAVApp

- Default: NAVX needs to be signed
- For testing purposes
  - you can use –skipverification
  - Or: use Self-Signed certificate

DigiCertUtil (https://www.digicert.com/util/)

Data

Handling Initial Data

Extension should ensure necessary setup has been done

Extension install
- Recommended when handling **archive data**

OnNewCompany
- Recommended when **new companies** need to be handled later

Setup page
- Recommended when **user input** is necessary

It is not possible to remove certain setup data when you uninstall an extension. (e.g. Job Queue, Web Services, …)

Upgrade Extensions

NAV TECH DAYS 2016
mibuso.com

You always need an **upgrade codeunit**

Codeunit 60199 Phase Out Data Upgrade - C/AL Editor

```
1  Documentation()
2
3  OnRun()
4
5  OnNavAppUpgradePerDatabase()
6
7  OnNavAppUpgradePerCompany()
8  NAVAPP.RESTOREARCHIVEDATA(DATABASE::Item);
9
```

Upgrade Extensions

Use generic code is/if possible

Upgrade Extensions

Use generic code if possible

Versions

Manage Versions in code:

- NAVAPP.GETARCHIVEVERSION
- 1.2.0.0 > 1.10.0.0

```
IF
gVersionFunctions.IsLessThan(NAVAPP.GETARCHIVEVERSION,'1.
0.19.4') THEN BEGIN
  RestoreFieldsInModifiedTables(70006800,70006999);
  RestoreAppTables(70006802,70006803);
  DeleteAppTables(70006804,70006999);
END ELSE IF
gVersionFunctions.IsLessThanOrEqualTo(NAVAPP.GETARCHIVEVE
RSION,'1.0.20.2') THEN BEGIN
  RestoreOnlyExistingFieldsOnTables(70006802,70006802);
  DeleteAppTables(23,23);
  RestoreFieldsInModifiedTables(70006800,70006999);
  RestoreAppTables(70006800,70006801);
  RestoreAppTables(70006803,70006999);
END ELSE BEGIN
  RestoreFieldsInModifiedTables(70006800,70006999);
  RestoreAppTables(70006800,70006999);
END;
```


That's all Folks!

Key Takeaways

Let PowerShell help you to be as efficient as possible
Let PowerShell help you to not forget stuff
Test AS an Extension – never expect your software to work as normal development
Get familiar with the development considerations
Use events with care

Have fun!