

E2E: Run your solution in a NAV Container

This is an end 2 end description on how to get your solution up running on Docker as a code customized solution, enabling you to extend your solution with Extensions v2 and start gradually moving pieces of your solution from a code customized solution to extensions v2.

Note: This description is a happy-path description. You might have other processes for taking your solution from one version to another, creating deltas, importing deltas etc. Feel free to use those methods instead.

Contents

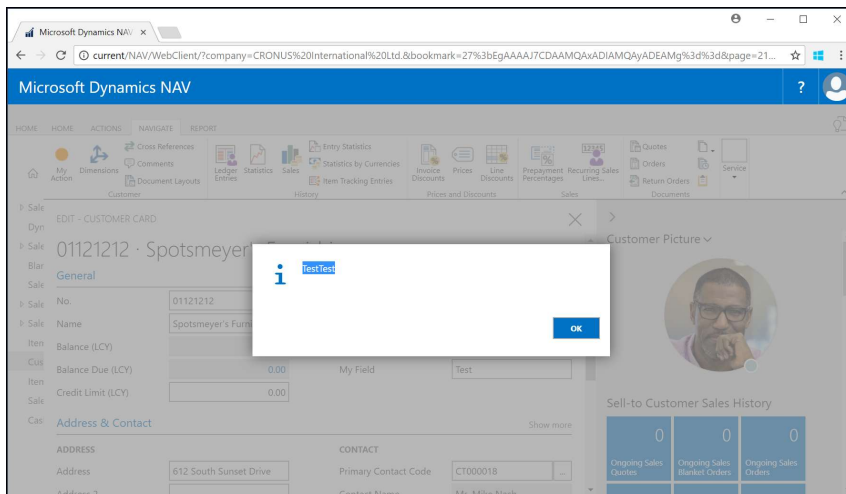
The soluton	2
Setup a development machine with docker.....	2
Install navcontainerhelper.....	2
Creating a CSIDE Development environment.....	3
Import your solution to your CSIDE environment.....	3
Export your solution as deltas	3
Remove the old CSIDE development environment.....	4
Create a “Tenerife” Development environment (with CSIDE)	4
Import deltas to the “Tenerife” development environment	5
“Fix” your solution	5
Install VS Code and the VSIX matching your container	6
Create an app depending on your code customized solution.....	6
Convert your code customized solution to an app	7
Export your app and tenant databases as .bacpac.....	7
Create a new local container based on .bacpac databases.....	7
Create a secure URLs for your .bacpac files	8
Running a container in Azure using Azure SQL.....	8

The solution

I have created a very simple solution, consisting of:

- A new field on the Customer table: "My Field"
- A Codeunit with a new function to multiply a text with a number
- A Run function in the Codeunit, which calls the function with the new field as a parameter
- Added this field to the Customer Card
- Added an action to the Customer Card to call the Codeunit

Number series used is 50100 and these 3 objects is saved as .fob based on NAV 2017 CU13 W1 in c:\temp\mysolution.fob.



If you want to try out this solution, you might be able to download the solution here:

<https://www.dropbox.com/s/mg6p7uut1szpncu/mysolution.fob?dl=0>

Setup a development machine with docker

You need a development machine, which has docker installed. Follow the process below to install docker or create an Azure Virtual machine which has docker pre-installed.

- Windows 10: <https://docs.docker.com/docker-for-windows/install/>
- Windows Server 2016: <https://docs.docker.com/install/windows/docker-ee/>
- Azure Virtual Machine: Create a Windows Server 2016 Datacenter – with containers

We recommend Windows Server 2016 for running docker.

Install navcontainerhelper

On your docker host computer, you need to install the latest version of the navcontainerhelper in PowerShell. Version 0.2.7.0 or later is required.

Navcontainerhelper is available here: <https://www.powershellgallery.com/packages/navcontainerhelper> and can be installed by starting PowerShell ISE and running:

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

If you already have navcontainerhelper installed, please use update-module to get the latest version.

Creating a CSIDE Development environment

To setup a CSIDE development container for NAV 2017 CU13 W1, you can use this script:

```
$mylicense = "c:\temp\mylicense.flf"
$oldimagename = "microsoft/dynamics-nav:2017-cu13-w1"
$oldcontainer = "old"

if ($credential -eq $null -or $credential -eq
[System.Management.Automation.PSCredential]::Empty) {
    $credential = get-credential -UserName $env:USERNAME `
        -Message "Please enter your windows credentials."
}

New-NavContainer -accept_eula `
    -containerName $oldcontainer `
    -auth windows `
    -credential $credential `
    -includeCSide `
    -licensefile $mylicense `
    -imageName $oldimagename `
    -updateHosts
```

Note: You need to replace the path of the license file with the location of your partner license.

The script will create a container, which is running Windows Authentication with your docker host. The container name is **old** and you will have a set of shortcuts on the desktop for launching Web Client, CSIDE, Windows Client etc.

When starting the container of a given version the first time, new-navcontainer will export all objects of the base app in order to be able to create deltas. This will take some time, but is needed later in the process.

Import your solution to your CSIDE environment

Running this script will import your solution to your CSIDE environment and compile the objects:

```
$mysolution = "c:\temp\mysolution.fob"
Import-ObjectsToNavContainer -containerName $oldcontainer `
    -objectsFile $mysolution

Compile-ObjectsInNavContainer -containerName $oldcontainer `
    -filter "modified=1"
```

Note: You need to replace the path of the solution file with the location of your .fob file.

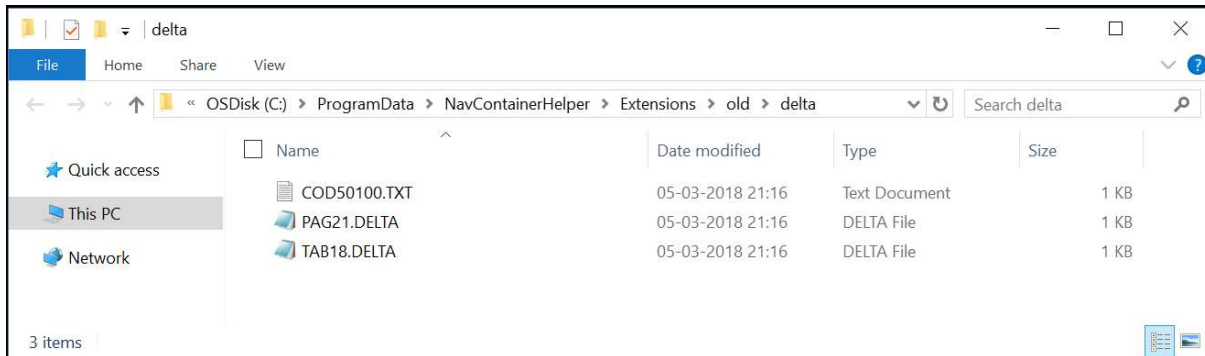
Export your solution as deltas

Navcontainerhelper contains a function, which can export your modifications as Deltas, run this:

```
Export-ModifiedObjectsAsDeltas -containerName $oldcontainer -openFolder
```

The openFolder flag means that the script will open the folder containing the solution. You should see this:

Run your solution in a NAV Container



Inspect the files to see that they contain only your modifications.

Remove the old CSIDE development environment

If you are running Windows 10, containers will be using HyperV isolation, which means that they are pre-allocating 4Gb memory for the container on startup. You might need to remove the old container before starting a new.

Note: Please remember to copy the delta folder to another location before removing the container as removing the container also removes the folder with data created by the container (including the delta folder). **I have copied the delta folder to c:\temp.**

Use:

```
Remove-NavContainer $oldcontainer
```

To remove the old container.

Create a “Tenerife” Development environment (with CSIDE)

Run this script to create a container using the latest version of the Developer Preview.

```
$olddeltaFolder = "c:\temp\delta"
$newimagename = "microsoft/dynamics-nav:devpreview-finus"
$newcontainer = "new"
New-NavContainer -accept_eula `
  -containerName $newcontainer `
  -auth windows `
  -credential $credential `
  -includeCSide `
  -licensefile $mylicense `
  -imageName $newimagename `
  -updateHosts `
  -multitenant `
  -enableSymbolLoading `
  -additionalParameters @"(-v ${olddeltaFolder}:c:\olddelta)"
```

The new container will also have shortcuts on the desktop for running the Web Client, CSIDE, Windows Client etc.

Note: The enableSymbolLoading flag will allow you to run development in VS Code side by side with CSIDE development.

Note: The additionalParameters parameter will share the folder containing the old deltas with the container.

Run your solution in a NAV Container

Note: The container is setup for multitenancy in order to be able to export app and tenant databases as bacpac later.

Note: After the container has started, it will display the URL with which you can download the .vsix file to use for Visual Studio code, use this URL to download the .vsix file.

Files:

<http://new:8080/a1-0.14.17461.vsix>

Import deltas to the “Tenerife” development environment

Run this script to import your deltas to the new container:

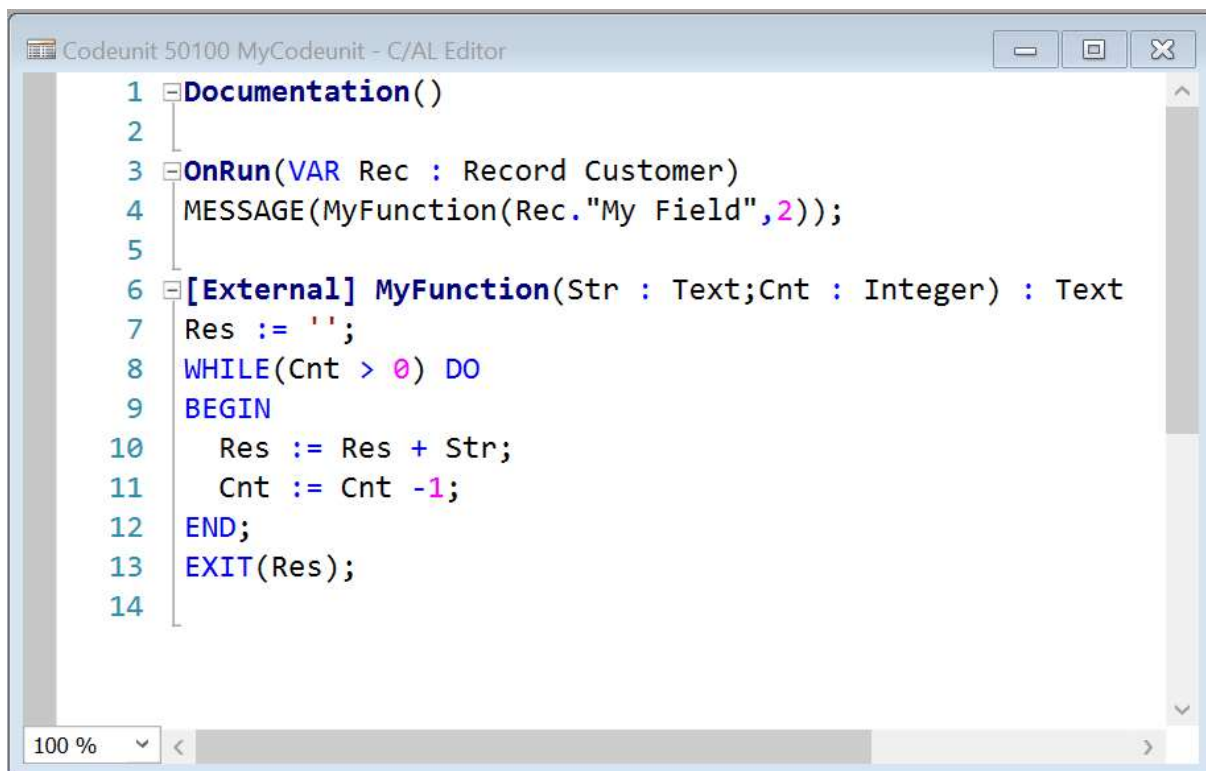
```
Import-DeltasToNavContainer -containerName $newcontainer -deltaFolder $olddeltaFolder
Compile-ObjectsInNavContainer -containerName $newcontainer `
    -filter "modified=1"
```

This will import your deltas and compile them.

“Fix” your solution

In my sample, I used NAV 2017 as base, meaning that ApplicationArea wasn't set on the control and the action on the page. Secondly, I also couldn't mark my function in the Codeunit as [external].

Start CSIDE for the new container and fix these issues (use Ctrl+F4 on the function to mark it as external).



```
Codeunit 50100 MyCodeunit - C/AL Editor
1 Documentation()
2
3 OnRun(VAR Rec : Record Customer)
4 MESSAGE(MyFunction(Rec."My Field",2));
5
6 [External] MyFunction(Str : Text;Cnt : Integer) : Text
7 Res := '';
8 WHILE(Cnt > 0) DO
9 BEGIN
10     Res := Res + Str;
11     Cnt := Cnt -1;
12 END;
13 EXIT(Res);
14
```

This topic is probably going to take much more time, when doing this on your own solution.

Install VS Code and the VSIX matching your container

Install Visual Studio code following the process here: <https://code.visualstudio.com/docs/setup/windows>

Install the .vsix from the file you downloaded from the new container.

Create an app depending on your code customized solution

Use **Ctrl+Shift+P** and select **AL: Go!**

Create a project using Your own server.

Modify **launch.json** with these values:

```
"server": "http://new",  
"serverInstance": "nav",  
"authentication": "Windows",  
"tenant": "default",  
"startupObjectId": 22
```

Use **Ctrl+Shift+P** and select **AL: Download Symbols**

Modify the HelloWorld.al to

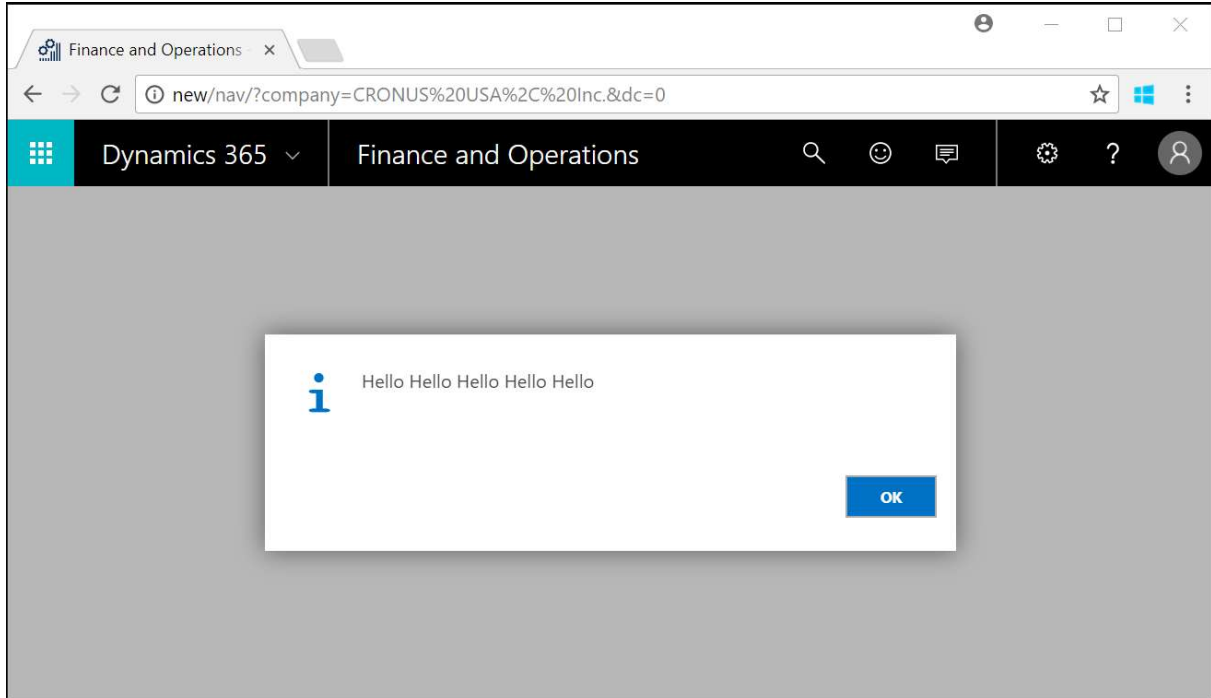
```
trigger OnOpenPage();  
var  
    mycodeunit: codeunit MyCodeunit;  
begin  
    Message(mycodeunit.MyFunction('Hello ',5));  
end;
```

Note: You will have the symbols for your codeunit and function available to use in Extensions V2 due to the enableSymbolLoading flag on the new container and the [external] flag on the function. You can now create events in CSIDE and use it in Extensions v2.

You can now gradually move things from your main solution to extensions v2, refactoring the app while going forward.

Run the solution using F5 and you should see:

Run your solution in a NAV Container



Convert your code customized solution to an app

Use this script to convert your solution to extensions v2

```
convert-ModifiedObjectsToAl -containerName $newcontainer -startId 50110 -openFolder
```

This should create deltas using the new syntax and send this to the txt2al tool to convert the objects to AL code.

Note: while writing this, it doesn't create a page extension based on the page delta, we are investigating why this is and fix it. The process should be the same though.

Export your app and tenant databases as .bacpac

Use this script to export your databases as .bacpac

```
Export-NavContainerDatabasesAsBacpac -containerName $newcontainer
```

You can specify a shared folder, to which the .bacpac files should be copied. If not, the .bacpac files will be exported and the folder in which both files are placed will be displayed.

Copy the .bacpac files to another location (ex. c:\temp\bacpac)

Create a new local container based on .bacpac databases

Use this script to create a new local container using the exported .bacpac files as databases:

```
$bacpacFolder = "c:\temp\bacpac"  
$localcontainer = "local"  
New-NavContainer -accept_eula `   
                  -containerName $localcontainer `   
                  -auth windows `   
                  -credential $credential `   
                  -includecside
```

Run your solution in a NAV Container

```
-licensefile $mylicense `
-imageName $newimagename `
-updateHosts `
-multitenant `
-enableSymbolLoading `
-additionalParameters @"(-v ${bacpacFolder}:c:\bacpac",
                        "-e appbacpac=c:\bacpac\app.bacpac",
                        "-e tenantbacpac=c:\bacpac\tenant.bacpac")
```

Note: The additionalParameters will share the folder in which the .bacpac files are and then set the appbacpac and the tenantbacpac parameters to files available in the container.

Now you can go back to the VS Code solution, change launch.json to use <http://local> instead of <http://new> as development server, download symbols and deploy your hello world solution to the new container.

Create a secure URLs for your .bacpac files

Follow the process described here: <https://blogs.msdn.microsoft.com/freddyk/2017/02/26/create-a-secure-url-to-a-file/> to upload your .bacpac files to a cloud storage and get secure URLs for the bacpacs.

I have uploaded the .bacpacs from this sample to Azure Storage and they can be downloaded here:

App Bacpac URL:

<https://directionswe.blob.core.windows.net/bacpacs/my/app.bacpac?st=2018-03-04T12%3A23%3A00Z&se=2020-01-01T12%3A23%3A00Z&sp=r&sv=2017-04-17&sr=b&sig=hi73XivZr2Q5c7wORYUEJ8vRk71LOCOqw9qQyqDTy6o%3D>

Tenant Bacpac URL:

<https://directionswe.blob.core.windows.net/bacpacs/my/tenant.bacpac?st=2018-03-04T04%3A23%3A00Z&se=2020-01-01T04%3A23%3A00Z&sp=r&sv=2017-04-17&sr=b&sig=bs%2Fvxq11vkvs%2Fjqys82ryG2cCiVbS67ou2fMAjxvdwy%3D>

Running a container in Azure using Azure SQL

Modify the contact email for Lets encrypt to your email and run this script to launch the Azure Portal with parameters pre-populated with Docker image and bacpac uris for app and tenant:

```
$contactemailforletsencrypt = "<your email address>"
$appbacpacuri =
"https://directionswe.blob.core.windows.net/bacpacs/my/app.bacpac?st=2018-03-04T12%3A23%3A00Z&se=2020-01-01T12%3A23%3A00Z&sp=r&sv=2017-04-17&sr=b&sig=hi73XivZr2Q5c7wORYUEJ8vRk71LOCOqw9qQyqDTy6o%3D"
$tenantbacpacuri =
"https://directionswe.blob.core.windows.net/bacpacs/my/tenant.bacpac?st=2018-03-04T04%3A23%3A00Z&se=2020-01-01T04%3A23%3A00Z&sp=r&sv=2017-04-17&sr=b&sig=bs%2Fvxq11vkvs%2Fjqys82ryG2cCiVbS67ou2fMAjxvdwy%3D"
$url = 'http://aka.ms/getnavext?navdockerimage='+[Uri]::EscapeDataString($newimagename)+
      '&appbacpacuri='+[Uri]::EscapeDataString($appbacpacuri)+
      '&tenantbacpacuri='+[Uri]::EscapeDataString($tenantbacpacuri)+
      '&sqlservertype=AzureSQL'+
      '&multitenant=Yes'+
      '&useletsencryptcertificate=Yes'+
      '&contactemailforletsencrypt='+[Uri]::EscapeDataString($contactemailforletsencrypt)
Start-Process $url
```

Note: using LetsEncrypt certificate for your Azure VM means that you do not have to install trust for a self signed certificate on your development machine.

Specify resource group, VM name and password and press purchase to deploy an Azure VM running the docker container using an Azure SQL database server with the two .bacpac files restored and used as app and tenant.

In the portal, you can select SQLExpress in order to use SQL Express instead of Azure SQL.

Run your solution in a NAV Container

After spinning up the Azure VM, you can modify launch.json with the URL of your server and UserPassword instead of windows. Example:

```
"server": "https://<publicdnsname>",  
"serverInstance": "nav",  
"authentication": "UserPassword",  
"tenant": "default",
```

Now use Download Symbols, authenticate with the NAV username (default admin) and the password you specified during Azure VM creation. Press F5 to deploy to your Azure development box.