How do we solve the issue of having dependencies to AppSource Apps

was “Discuss NuGet packages for Business Central and DevOps”

The problem!

- ISV’s or VARs developing apps with dependencies on AppSource apps cannot easily build and run automated testing for their app
- Partners have to spend time on providing and gathering these packages for their DevOps solution
- On-premises customers using AppSource apps need to go through the same hardship
- Partners also need the objects for AppSource apps in their developer license in order to work with it

Design with the end-goal in mind

This presentation doesn’t include any promises

everything are ideas and suggestions

Nirvana aka. The GOAL!

- “All AppSource apps” are available in a public Nuget server as runtime packages for all compatible BC versions.
- The NuGet server is owned and hosted by Microsoft and only Microsoft can deploy apps to the server (i.e. trusted)
- All NuGet packages have a full dependency tree included (of other AppSource apps, Base app, other Microsoft apps and platform)
- Partners can have private NuGet servers, if they elect to not be part of the public server
- NuGet packages can be used for on-premises as well as for your DevOps setup
- CRONUS license can run all AppSource apps
- With every new minor of Business Central, all NuGet packages are updated with an additional runtime package (if compatible)
- With every new submission of apps to AppSource, a new NuGet package is added with the new version number (for all BC versions)

With this, we could…

If you add a dependency to an AppSource app to your app.json, your DevOps setup should be able to resolve this automatically (including all dependencies, if a compatible version isn't already installed. These apps can also be force-updated)

You can spin up a Business Central NST on-premises (or container) with an empty database and just install any AppSource app, which then will pull all dependencies along

You can upgrade Business Central on-premises (or container) to a new version of an AppSource App (incl. all dependencies + platform)

There’s a long way… - let’s break it down 1 level

1. Get AppSource ranges in the CRONUS license…
2. Define NuGet package standard format
3. Allow BcContainerHelper, AL-Go and other DevOps solutions to support using this NuGet package format (both from private (partner) and public (MS) NuGet servers)
4. Add support in ALC to create runtime packages without the NST (will make generation of runtime packages possible – without this, every version takes ~10 minutes)
5. Allow BcContainerHelper, AL-Go and other DevOps solutions to support delivering to this NuGet package format (to a private or public NuGet Server)
6. Be able to determine runtime package emit version from BC application version and vice versa
7. Add support for properties in app.json for identifying home repo and generated by
8. Add a ResourceExposurePolicy to allow the app to be shared as NuGet (Fail validation if dependent apps are not allowed to be exposed?)
9. Generate NuGet packages when apps are updated in AppSource
10. Update NuGet packages when BC versions are created

1. Get the AppSource app range in CRONUS

- We will try to make this happen in 2023 wave 1
  - Obviously, we need LT approval for this, but it is really a small task for us

2. Define NuGet package standard format

- I have created a proof-of-concept prototype in BcContainerHelper
- A discussion on the topic is here:
  - https://github.com/microsoft/AL-Go/issues/261
  - https://github.com/microsoft/AL-Go/issues/262

3. Use BcNuGet packages

- Proof-Of-Concept functions has been added to ContainerHelper
  - Publish-BcNuGetPackageToContainer
  - Get-BcNuGetPackage

4. Use ALC to create runtime packages

- Speed up process and save the environment

Today, easy to code, but time consuming! and… - which versions are necessary?

```
$packages = @{}
'20.0','20.1','20.2','20.3','20.4','20.5','21.0','21.1' | % {
    $artifactUrl = Get-BCArtifactUrl -country us -version $_
    $destinationFolder = Convert-BcAppsToRuntimePackages `
        -containerName cnvt `
        -imageName "" `
        -artifactUrl $artifactUrl `
        -licenseFile $LicenseFile `
        -apps @("C:\...BingMaps.AppSource_3.1.254.0.app")
    $packages."$_" = $destinationFolder
}
```

Takes approx. 1 hour (45 minutes if artifacts and generic image is cached)

If all runtime appsource apps…

- Should have a runtime packages generated for an average of 6 BC versions – this would take 2250 compute hours (~94 days)
  - Partners would have to pay for this
  - With the optimized runtime package generation, it would be more like a few hours
- Every month, with a new BC version, you would add 375 hours (~15 days) of compute time
  - Partners would have to pay for this
  - With the optimized runtime package generation, this would be minutes

5. Generate BcNuGet packages

- Proof-Of-Concept functions has been added to ContainerHelper
  - New-BcNuGetPackage
  - Push-BcNuGetPackage