Dependency Flow Onboarding for Repos Not on Arcade

Overview

Dependency flow is the method by which .NET repos consume other product repo's assets. In order to participate in dependency flow, a repo must produce a manifest of what assets / packages that repo produces and then publish that manifest to the Build Asset Registry (B.A.R.). This document is intended to provide guidance for repos which are not using the Arcade Sdk but still need to participate in dependency flow. The end goal is that the repo is able to produce a manifest and publish that to B.A.R.

To do so, a repo follows almost the same process as normal arcade publishing. The repo directly uses the `PushToAzureDevOpsArtifacts` task in the Microsoft.DotNet.Build.Tasks.Feed package (available from the dotnet-eng feed - `https://pkgs.dev.azure.com/dnceng/public/_packaging/dotnet-eng/nuget/v3/index.json`). This task uploads the artifacts to build storage and generates a manifest describing the build. The manifest is the used to publish the new build to the build asset registry (BAR).

Creating a manifest

To create a manifest, use the PushToAzureDevOpsArtifacts, providing the feed that packages pushed to. The below target performs this task.

```
<UsingTask TaskName="Microsoft.DotNet.Build.Tasks.Feed.PushToAzureDevOpsArtifacts" AssemblyF

<Target Name="PublishToBuildAssetRegistry">
  <PropertyGroup>
    <AssetManifestFileName>Assets.xml</AssetManifestFileName>
    <AssetManifestPath>$(ArtifactsLogDir)AssetManifest\$(AssetManifestFileName)</AssetManife
  </PropertyGroup>

  <Error Condition="!Exists($(NuGetClientNupkgsDirectoryPath))" Text="The package directory
  <Error Condition="Exists($(AssetManifestPath))" Text="The manifest file '$(AssetManifestPa

  <CreateItem Include="$([System.IO.Path]::Combine($(NuGetClientNupkgsDirectoryPath), '*.nup
    <Output TaskParameter="Include" ItemName="ItemsToPush" />
  </CreateItem>

  <Error Condition="'@(ItemsToPush)' == ''" Text="No packages to push." />

  <Error Condition="'$(BUILD_BUILDNUMBER)' == ''" Text="The BUILD_BUILDNUMBER property is re
  <Error Condition="'$(ArtifactsLogDir)' == ''" Text="The ArtifactsLogDir property is requir
  <Error Condition="'$(BUILD_SOURCEBRANCH)' == ''" Text="The BUILD_SOURCEBRANCH property is
  <Error Condition="'$(BUILD_SOURCEVERSION)' == ''" Text="The BUILD_SOURCEVERSION property i
```

```
  <Error Condition="'$(BUILD_REPOSITORY_URI)' == ''" Text="The BUILD_REPOSITORY_URI property
  <Error Condition="'$(MaestroAccessToken)' == ''" Text="The MaestroAccessToken property is
  <Error Condition="'$(MaestroApiEndpoint)' == ''" Text="The MaestroApiEndpoint property is
  <Error Condition="'$(SYSTEM_TEAMFOUNDATIONCOLLECTIONURI)' == ''" Text="The SYSTEM_TEAMFOUN
  <Error Condition="'$(SYSTEM_TEAMPROJECT)' == ''" Text="The SYSTEM_TEAMPROJECT property is
  <Error Condition="'$(BUILD_BUILDID)' == ''" Text="The BUILD_BUILDID property is required."
  <Error Condition="'$(SYSTEM_DEFINITIONID)' == ''" Text="The SYSTEM_DEFINITIONID property i

  <Message Text="Publishing %(ItemsToPush.Identity)" Importance="normal" />

  <ItemGroup>
    <ManifestBuildData Include="InitialAssetsLocation=https://dev.azure.com/url/to/my/feed/i
    <ManifestBuildData Include="AzureDevOpsBuildId=$(BUILD_BUILDID)" />
    <ManifestBuildData Include="AzureDevOpsBuildDefinitionId=$(SYSTEM_DEFINITIONID)" />
    <ManifestBuildData Include="AzureDevOpsProject=$(SYSTEM_TEAMPROJECT)" />
    <ManifestBuildData Include="AzureDevOpsBuildNumber=$(BUILD_BUILDNUMBER)" />
    <ManifestBuildData Include="AzureDevOpsRepository=$(BUILD_REPOSITORY_URI)" />
    <ManifestBuildData Include="AzureDevOpsBranch=$(BUILD_SOURCEBRANCH)" />
  </ItemGroup>

  <PushToAzureDevOpsArtifacts
    ItemsToPush="@(ItemsToPush)"
    ManifestBuildData="@(ManifestBuildData)"
    ManifestRepoUri="$(BUILD_REPOSITORY_NAME)"
    ManifestBranch="$(BUILD_SOURCEBRANCH)"
    ManifestBuildId="$(BUILD_BUILDNUMBER)"
    ManifestCommit="$(BUILD_SOURCEVERSION)"
    AssetManifestPath="$(AssetManifestPath)"
    PublishingVersion="3" />
</Target>
```

Publish the manifest to BAR

Publishing a single manifest

If you only have one Azure DevOps job that publishes assets, then you can add this task into your build steps.

Publish manifest to BAR

```
powershell -ExecutionPolicy Bypass -Command "eng\common\sdk-task.ps1
-task PublishBuildAssets -restore -msbuildEngine dotnet
/p:ManifestsPath='$(Build.StagingDirectory)/Download/AssetManifests'
/p:BuildAssetRegistryToken=$(MaestroAccessToken) /p:MaestroApiEndpoint=https://maestro-
```

`MaestroAccessToken` is available by referencing the “Publish-Build-Assets” variable group in dnceng/internal.

Publishing multiple manifests

If you have multiple jobs that publish assets, then you need to publish the generated manifests from each of those legs. In Arcade, this is done by publishing each of the manifests as Artifacts to Azure DevOps and then having a final job that runs and downloads the manifests / publishes them all to BAR

Was this helpful?