.NET - Prepare and deliver the source code

Summary: This section describes how to prepare and deliver the source code of your .NET application.

Information about discovery

Discovery is a process that is actioned during the packaging process whereby CAST will attempt to automatically identify projects within your application using a set of predefined rules. This discovery process also allows CAST AIP to set the initial analysis configuration settings explained in **.NET - Analysis configuration**. Discoverers are currently **embedded in CAST AIP**:

- **Microsoft Visual Studio .NET Discoverer**

You should read the relevant documentation for each discoverer (provided in the link above) to understand how the source code will be handled.

Source code delivery using CAST AIP Console

See **Application onboarding** and **Application onboarding - prerequisites** for more detailed information about the steps you should take to deliver your source code.

Prepare the application source code

AIP Console expects the application source code to be delivered either via a **ZIP file** or via a **source code located in a folder** configured in AIP Console. Whichever option you chose, you should include in the ZIP/source code folder all of your .NET application source code. CAST highly recommends placing all the relevant files in a folder and using sub-folders where necessary. You can deliver other technologies at the same time (for example, database DDL). If you are using a ZIP/archive file, zip the folders in the "temp" folder as shown in the image below - but do not zip the "temp" folder itself, nor create any intermediary folders:

```
D:\temp
        |-----DotNET
        |-----OtherTechno1
        |-----OtherTechno2
```

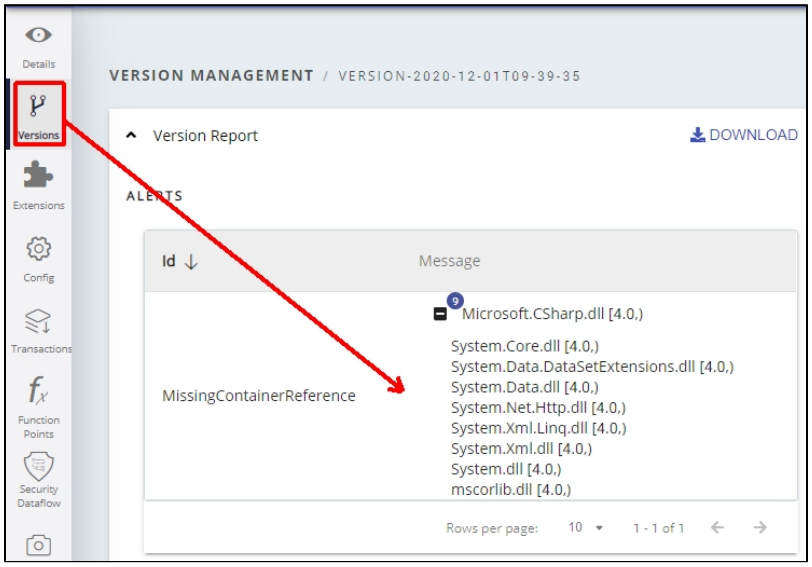
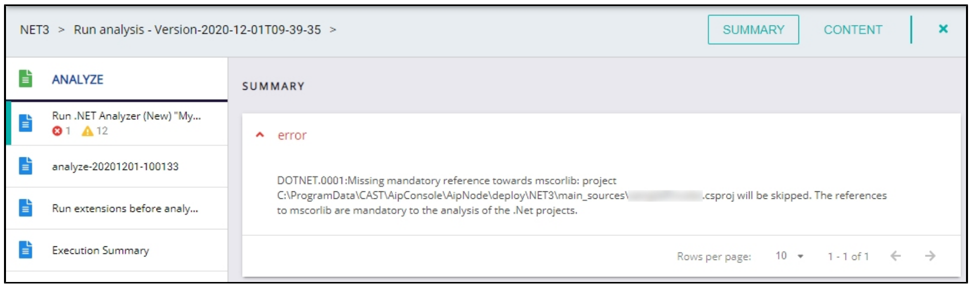
Any additional framework specific source code (such as **Entity Framework**, **Silverlight Framework**, **WCF**, **WPF**, **NoSQL** should also be provided in the ZIP/archive file or the source code folder.

What about delivering framework/external/custom assemblies?

The .NET Analyzer needs to know the location of any **assemblies** such as the **.NET Framework assemblies, external assemblies (third-party DLLs)** and other **custom assemblies** that are used by your application source code. There are various ways to declare the location of these assemblies when using AIP Console, but the action you choose also depends on the release of the .NET Analyzer you are using. This is explained in more detail below:

.NET Framework assemblies

.NET Framework assemblies are used by all .NET applications and therefore the .NET Analyzer needs to have access to these assemblies in order to resolve references correctly. Here there is a choice of options:

Using .NET Analyzer 1.1	When using .NET Analyzer **1.1**, the .NET Framework assemblies are provided in the extension itself (as listed in the section **Dependent fra** **extension** in the **extension documentation**) and they will be used to resolve references to specific assemblies that have been used by the declare the location of these .NET Framework assemblies, however, if you do not declare them, **"missing references" type alerts** will be g example as shown below. These alerts can be **safely ignored**. The resulting analysis will use the .NET Framework assemblies delivered in *Click to enlarge*  Details Versions Extensions Config Transactions Function Points Security Dataflow DOWNLOAD VERSION MANAGEMENT / VERSION-2020-12-01T09-39-35 Version Report ALERTS Id — Message MissingContainerReference — Microsoft.CSharp.dll [4.0.) System.Core.dll [4.0.) System.Data.DataSetExtensions.dll [4.0.) System.Data.dll [4.0.) System.Net.Http.dll [4.0.) System.Xml.Linq.dll [4.0.) System.Xml.dll [4.0.) System.dll [4.0.) mscorlib.dll [4.0.) Rows per page: 10 1 - 1 of 1 If you prefer not to have missing references type alerts, then you can, optionally, declare the location of the .NET Framework assemblies us Node responsible for analyzing your application, ensuring that the AIP Node can access the declared location (i.e. a folder on the AIP Node used **in priority** over the .NET Framework assemblies provided in the extension itself.
Using .NET Analyzer 1.0	When using .NET Analyzer **1.0**, the .NET Framework assemblies are NOT provided in the extension, therefore you MUST declare the locati `node-app.properties` file, ensuring that the AIP Node responsible for analyzing your application can access the declared location (i.e. a you do not declare them, then the analysis will fail with a **missing mandatory reference error**:  NET3 > Run analysis - Version-2020-12-01T09-39-35 > SUMMARY CONTENT ANALYZE Run .NET Analyzer (New) "My... 1 12 analyze-20201201-100133 Run extensions before analy... Execution Summary SUMMARY error DOTNET.0001:Missing mandatory reference towards mscorlib: project C:\ProgramData\CAST\AipConsole\AipNode\deploy\NET3\main_sources\[redacted].csproj will be skipped. The references to mscorlib are mandatory to the analysis of the .Net projects. Rows per page: 10 1 - 1 of 1

External assemblies (third-party DLLs)/custom assemblies

When using .NET Analyzer **1.1**, CAST provides some specific frameworks and third party packages in the extension itself which will automatically be used. Therefore if your source code uses these specific frameworks and third party packages, it is not necessary to deliver these items. However, **missing library/assembly alerts** for these items will be generated during the delivery. The alerts can be safely ignored if the alert references an item that CAST provides in the extension.

See the section **Dependent frameworks and third-party packages provided in the extension documentation** for more information.

If your application uses external assemblies provided by third parties or your own custom assemblies, the .NET Analyzer also needs to know the location of them:

- if these assemblies **are stored in the correct location as specified in the project definition file** and are delivered with the application (in the **ZIP file** or via a **source code located in a folder**), then the .NET Analyzer will find them without you needing to do anything.
- if these assemblies **are not stored in the correct location**, then there are two options available to you to ensure that the .NET Analyzer is aware of their location:

Declare the location in aip-node-app.properties	This option involves: • placing the assemblies in a folder that the AIP Node responsible for analyzing your application can access, i.e. a folder on the AIP Node or a shared network location. • declaring this folder using the `aip-node-app.properties` file on the AIP Node. When the analysis is run, the .NET Analyzer will search the folder declared in `aip-node-app.properties` and find the assemblies required by your application source code.
Deliver with the application source code	This option involves: • delivering the assemblies in a dedicated folder (typically this is a sub folder or the "bin" folder) together with the application source code (i.e. in the **ZIP file** or via a **source code located in a folder**). • ensuring that the `scanner.detect.dotnet.assemblies` option is set to **true** the `aip-node-app.properties` file on the AIP Node. When the analysis is run, the .NET Analyzer will attempt to find the assemblies required by your application source code. This option is **limited in scope** however and if in doubt, you should use the alternative option described above (**Declare the location in aip-node-app.properties**): only one folder in the delivered source code will be detected by the .NET Analyzer - the folder containing the largest number of assemblies (dll files). Therefore if your assemblies are distributed in multiple folders throughout the source code, this method is not recommended.

What about Nuget package dependencies?

An extension (**NuGet Resources Extractor**) has been published that provides the means to configure an automatic extraction of **NuGet package dependencies** from a NuGet repository specifically for .NET application source code. In other words, NuGet package based source code that resides in a simple local or **nuget.org** location. For example, when your .NET application contains **.csproj files** which have package references defined, you can use this extractor to extract those NuGet packages from the NuGet repository.

Out of the box, if a **.csproj** file is detected in the delivered source code, the extension will be downloaded and installed as part of the analysis process. If the .csproj file contains **package dependency references**, these references will automatically be accessed and included in the analysis. Example package references shown below:

```
<ItemGroup>
    <!-- ... -->
    <PackageReference Include="Contoso.Utility.UsefulStuff" Version="3.6.0" />
    <!-- ... -->
</ItemGroup>
```

The extractor will extract all NuGet package dependencies and place them inside a folder called "**nugetPck**" folder located in the Deploy folder.
The extractor is driven by the `%PROGRAMDATA%\CAST\AipConsole\AipNode\aip-node-app.properties` file attribute `scanner.nuget.repository`:

```
# HTTP V3 Nuget repository to download package dependencies https://api.nuget.org/v3/index.json or file system
like file://C:/Users/johndoe/.nuget/packages
scanner.nuget.repository=https://api.nuget.org/v3/index.json
```

Declaring options/locations in aip-node-app.properties

To declare the location, edit the following file with a text editor on the AIP Node responsible for analyzing your Application:

```
%PROGRAMDATA%\CAST\AipConsole\AipNode\aip-node-app.properties
```

Locate the following sections:

```
# ==============
# .NET assembly configuration
# --------------
# Provide a list of locations to use for .NET Assemblies to be given to the code scanner
# Provide a full path separated by ';' like
#scanner.dotnet.assembly.locations=C:/dotNet/v4.0;C:/dotNet/v4.5;C:/dotNet/v3.5
scanner.dotnet.assembly.locations=
# enable this option to detect the dotnet assemblies in the source code(feature in beta)
scanner.detect.dotnet.assemblies=false
```

Now edit either/both of the options as appropriate:

scanner.dotnet.assembly.locations	Use this option to declare the assemblies as required. For example to declare .NET Framework assemblies installed the AIP Node you can add `C:/Windows/Microsoft.NET/Framework64/v4.0.30319`. Or to declare custom assemblies in a shared folder: `S:/shared/assemblies`:
	`#scanner.dotnet.assembly.locations=C:/dotNet/v4.0;C:/dotNet/v4.5;C:/dotNet/v3.5`
	`scanner.dotnet.assembly.locations=C:/Windows/Microsoft.NET/Framework64/v4.0.30319;S:/shared/assemblies`
	• Path must use **single forward slashes** or **double back slashes** - the single back slash is not valid
	• Multiple paths must be separated by **semi-colons**.

scanner.detect.dotnet.assemblies	Set this option to **true** if you have decided to deliver assemblies in the **ZIP file** or via a **source code located in a folder** and these assemblies are not in the location defined in the project definition file.

Save the **aip-node-app.properties** file and then restart the AIP Node package in order for the configuration to be taken into account.

Source code delivery using legacy CAST Delivery Manager Tool

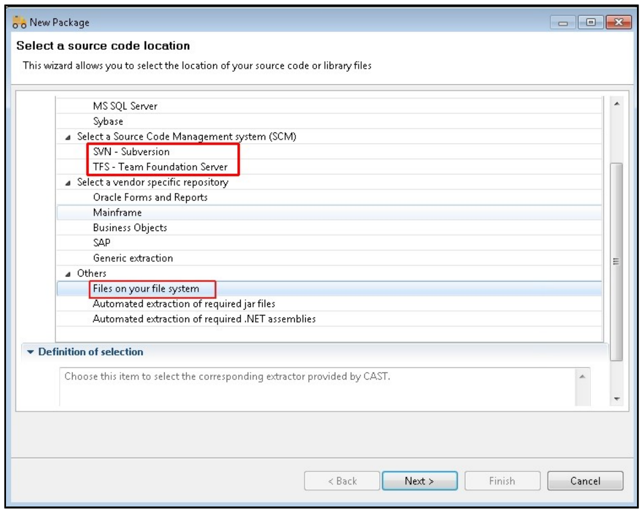
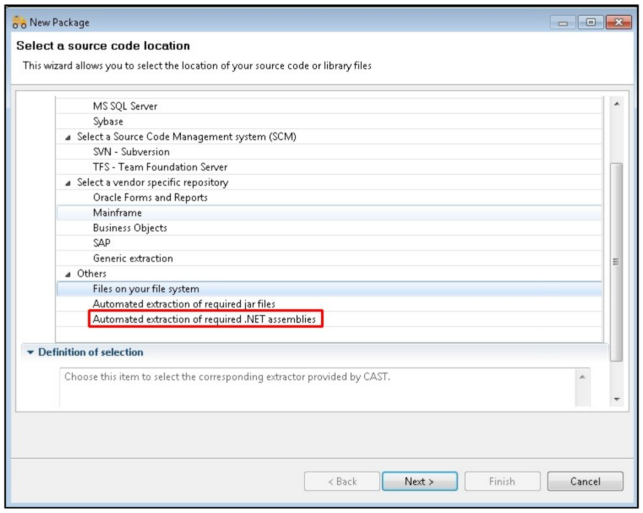
How do I add a source code package to my delivery

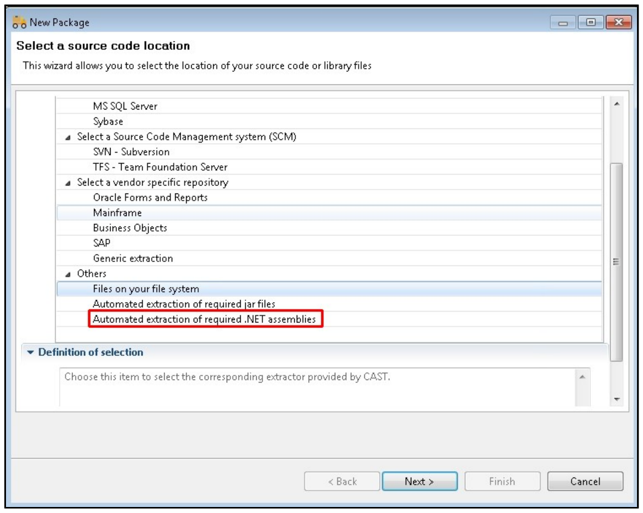
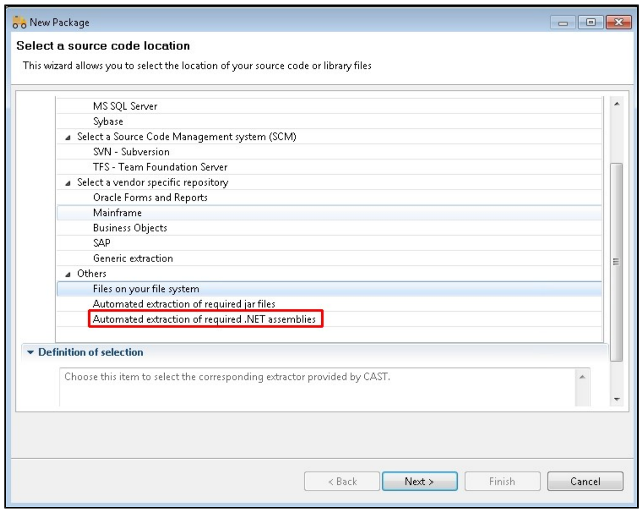
See How do I add a source code package to my delivery.

What you should package?

When creating packages to discover your .NET application you should create them as listed below:

Package no.	Package name /type	Mandatory?		Location /path	Notes
		.NET Analyzer v. 1.0	.NET Analyzer v. 1.1		

1	Source code	✓	✓	Source code root folder	Use the "**Files on your file system**" / **SVN** / **TFS** options in the CAST Delivery Manager Tool: *Click to enlarge*  New Package Select a source code location This wizard allows you to select the location of your source code or library files MS SQL Server Sybase Select a Source Code Management system (SCM) SVN - Subversion TFS - Team Foundation Server Select a vendor specific repository Oracle Forms and Reports Mainframe Business Objects SAP Generic extraction Others Files on your file system Automated extraction of required jar files Automated extraction of required .NET assemblies Definition of selection Choose this item to select the corresponding extractor provided by CAST. < Back Next > Finish Cancel Note that additional framework specific source code (such as Entity Framework, Silverlight Framework, WCF, WPF, NoSQL (MongoDB)) should be provided as part of this package.
2	External assemblies (third party DLL)	If required, one package per vendor.		N/A: depends on the third party	Use the "**Automated extraction of .NET assemblies on your file system**" option in the CAST Delivery Manager Tool: *Click to enlarge*  New Package Select a source code location This wizard allows you to select the location of your source code or library files MS SQL Server Sybase Select a Source Code Management system (SCM) SVN - Subversion TFS - Team Foundation Server Select a vendor specific repository Oracle Forms and Reports Mainframe Business Objects SAP Generic extraction Others Files on your file system Automated extraction of required jar files Automated extraction of required .NET assemblies Definition of selection Choose this item to select the corresponding extractor provided by CAST. < Back Next > Finish Cancel **When to create this package** This package should only be created when your application uses third party assemblies. Note that in .NET Analyzer v. **1.1**, CAST provides some specific frameworks and third party packages with the extension which will automatically be used. Therefore if your source code uses these specific frameworks and third party packages it is not necessary to create a specific package to deliver these items. However because it is not necessary to create a package, **missing library/assembly alerts** for these items will be generated during the packaging action in the CAST Delivery Manager Tool. The alerts can be safely ignored if the alert references an item that CAST provides in the extension. See the section **Dependent frameworks and third-party packages provided in the extension** documentation for more information.

3	.NET framework (system assemblies)	✓	✗ (not required in v. 1.1)	C:\Windows\Microsoft.NET\Framework64\<use the version used by your projects>	Use the "**Automated extraction of .NET assemblies on your file system**" option in the CAST Delivery Manager Tool: *Click to enlarge* New Package **Select a source code location** This wizard allows you to select the location of your source code or library files MS SQL Server Sybase Select a Source Code Management system (SCM) SVN - Subversion TFS - Team Foundation Server Select a vendor specific repository Oracle Forms and Reports Mainframe Business Objects SAP Generic extraction Others Files on your file system Automated extraction of required jar files Automated extraction of required .NET assemblies Definition of selection Choose this item to select the corresponding extractor provided by CAST. < Back · Next > · Finish · Cancel If **"C:\Windows\Microsoft.NET\Framework64\<use the version used by your projects>**" is NOT available on the machine, please use the most recent framework installed on the machine instead: "**C:\Windows\Microsoft.NET\Framework64\<use the most recent>**". In version **1.0.x**, the .NET analyzer will not attempt to extract the .NET framework assemblies during the analysis: this is why a specific package for external assemblies must be created in the CAST Delivery Manager Tool. If this package is not created and delivered, **the .NET analyzer will skip all projects** having a dependency to .NET assemblies, leading to incomplete analysis results. In version **1.1**, this package containing the .NET framework assemblies is no longer required since the .NET framework is provided in the extension and will be used instead. However because it is not necessary to create a package, **missing library/assembly alerts** for these items will be generated during the packaging action in the CAST Delivery Manager Tool. The alerts can be safely ignored if the alert references an item that CAST provides in the extension. See the section **Dependent frameworks and third-party packages provided in the extension** documentation for more information.
4	Custom assemblies	If required (typically when there are unresolved alerts)		Source code root folder (or "bin" sub-folder)	Use the "**Automated extraction of .NET assemblies on your file system**" option in the CAST Delivery Manager Tool: *Click to enlarge* New Package **Select a source code location** This wizard allows you to select the location of your source code or library files MS SQL Server Sybase Select a Source Code Management system (SCM) SVN - Subversion TFS - Team Foundation Server Select a vendor specific repository Oracle Forms and Reports Mainframe Business Objects SAP Generic extraction Others Files on your file system Automated extraction of required jar files Automated extraction of required .NET assemblies Definition of selection Choose this item to select the corresponding extractor provided by CAST. < Back · Next > · Finish · Cancel **When to create this package** • When the custom assemblies are stored in the correct location as specified in the project definition, then package #1 will retrieve them. In this case no alert will be raised by the CAST Delivery Manager Tool about missing assemblies, so in this case, **there is no need** to create package #4. • When the custom assemblies are not stored in the correct location as specified in the project definition, then package #1 will not be able to retrieve them. "Missing assemblies" alerts will be raised for package #1, so the creation of package #4 for custom assemblies located either in the source code root folder or in the "bin" sub-folder **is required** to clear these alerts.

Information extracted during the packaging

How do I fine-tune my Version ?

See How do I fine-tune my Version for more information.

Note that if you are using **1.1** of the .NET Analyzer extension, CAST provides some **specific frameworks and third party packages with the extension,** which will automatically be "used" during an analysis - as such, no specific DMT package is required for these. However because it is not necessary to package the frameworks and third party packages used by your source code, **missing library/assembly alerts** for these items will be generated during the packaging action in the CAST Delivery Manager Tool. The alerts can be safely ignored if the alert references an item that CAST provides in the extension.

See the section **Dependent frameworks and third-party packages provided in the extension** documentation for more information.

How do I deliver the Version for analysis?

See How do I deliver the Version for analysis for more information.

Delivery acceptance

See Validate and Accept the Delivery for more information.

Technical information

Information extracted during delivery

C# and VB.NET

- Source files and their associated **BuildAction** (e.g. Compile, Content, EmbeddedResource, None, etc)
- Assembly name (can be different from the project name)
- .NET framework version
- Compilation constants, including their values for VB.NET

C# specific

- Default namespace (the default namespace to be used when creating new files; will be used when processing .xsd files)
- Option "Allow unsafe code"

VB.NET specific

- Root namespace (the default namespace to be used instead of the global namespace)
- Option explicit (On / Off)
- Option strict (On / Off)
- Option infer (On / Off)
- Imported namespaces (namespaces that are automatically imported by each file in the project)