Build components

The ReadyRoll *build components* depend on the migration scripts created by the *Visual Studio extension*.

The ReadyRoll *build components* work with any continuous integration server (eg, VSTS/TFS, TeamCity, Jenkins, Bamboo, etc), and let you:

- build the database alongside an application as part of a continuous integration process
- validate the migration scripts
- run tSQLt tests, integration tests, etc
- create a package used by the *release components*

Requirements

- Any continuous integration server that can run MSBuild
- For build-time migration script verification, then on the agents:
  1. Microsoft ODBC Driver 13.1 for SQL Server (x86/x64)
  2. Microsoft Command Line Utilities 14.0 for SQL Server (x86/x64)
  3. After installing the above, the machine (or agent process) must be restarted: the above installer adds the utility to the PATH system environment variable, enabling ReadyRoll to find and execute the SqlCmd utility.

Installation

The *build components* need to be installed on your build agents.

There are a few of options available to you. Options A and B are easier to set up and maintain, however if consistency between Visual Studio and Continuous Integration builds is a priority, Option C is recommended.

Option A - In VSTS/TFS, use the ReadyRoll build definition template

If you are building a ReadyRoll project with the build definition template provided by the ReadyRoll extension, the ReadyRoll build components will be installed to your build agent automatically using two build tasks: *NuGet Tool Installer* and *ReadyRoll Download Build Components.*

If you have a custom build definition, just add these two tasks before the *Visual Studio Build* task.

Some versions of TFS may not support these tasks. If you cannot find them in your version of TFS, consider options B or C.

More detailed instructions on how to set up a ReadyRoll build in VSTS/TFS can be found here.

Or, Option B – Install the ReadyRoll Visual Studio extension on each build agent:

If you have admin and remote desktop access to your build agents, you may find this to be the simpler of the two options.

Install the ReadyRoll *build components* on each build agent by downloading and installing the ReadyRoll Visual Studio extension (eg, via remote desktop).

Or, Option C – Use the ReadyRoll.MSBuild NuGet package:

If you don't have admin access to your build agents, or if you would like to control the versioning of your build-time dependencies, ReadyRoll also allows you to install the *build components* using the ReadyRoll.MSBuild NuGet package.

The below steps differ to the standard package installation/upgrade process due to the lack of support within the NuGet client for database projects. This necessitates the addition of an (otherwise empty) class library project within your solution, which acts as a placeholder for the NuGet package reference.

1. In Visual Studio, open the solution containing database project(s) and add a new *Class Library (.NET Framework)* project to the solution, e.g. *PlaceholderLibrary*
2. Open the *Package Manager Console* by clicking **View** then **Other Windows** then **Package Manager Console**.
3. From the *Default Project* drop-down select the placeholder class library project.
4. In the **Package Manager Console**, enter:

```
Install-Package ReadyRoll.MSBuild
```

5. In the **Solution Explorer**, right-click the ReadyRoll database project, then click **Unload Project**.
6. In the **Solution Explorer**, right-click the ReadyRoll database project, then click **Edit <ProjectName>.sqlproj**
7. In the *.sqlproj file*, add the following snippet under the root node (*<Project>*):

```
<PropertyGroup>
  <ReadyRollNuGetBaseFolder>$(MSBuildThisFileDirectory)..\packages</ReadyRollNuGetBaseFolder>
  <ReadyRollNuGetIsRestored Condition="$([System.IO.Directory]::GetDirectories
($(ReadyRollNuGetBaseFolder), 'ReadyRoll.MSBuild.*').Length) != 0">True</ReadyRollNuGetIsRestored>
  <ReadyRollTargetsPath Condition="$(ReadyRollNuGetIsRestored) == 'True'">$([System.IO.Directory]::
GetDirectories($(ReadyRollNuGetBaseFolder), 'ReadyRoll.MSBuild.*')[0])\tools\ReadyRoll.Data.Schema.SSDT.
targets</ReadyRollTargetsPath>
</PropertyGroup>
```

> If your project was created with ReadyRoll 1.14.17 or earlier, the *.sqlproj* file will contain another instance of the *<ReadyRollTargetsPath>* element. Ensure that the element is only specified once by replacing it with the above *<ReadyRollTargetsPath>* definition.

> The *<ReadyRollNuGetBaseFolder>* element in the above snippet assumes that your solution file (.sln) is in the parent folder of the ReadyRoll database project *.sqlproj* file. If the solution file is not in the parent folder, then you will need to modify the relative path in the value as appropriate (that is the part that contains the sub-string "*..\packages*").

8. Save the file.
9. In the **Solution Explorer**, right-click the ReadyRoll database project, then click **Reload Project**.
10. In the dialog box, click **Yes**.
11. To confirm that the *build components* have been successfully installed, **Rebuild Solution**, then in the **Output** window, find the line containing **"Using ReadyRoll toolpath"** and check that the path to the *.targets* file is within your solution's *packages* folder.
12. Commit the changes to source control.

> You will either need to do a *NuGet restore* during build, or also commit the *packages* folder to source control, just like with any other NuGet reference.

You have now completed the installation of the *ReadyRoll.MSBuild* NuGet package.

Upgrading the ReadyRoll.MSBuild NuGet package

1. Close and re-open Visual Studio. *This is needed in order to release file locks on ReadyRoll build-time dependencies.*
2. Open the *Package Manager Console* by clicking **View** then **Other Windows** then **Package Manager Console**.
3. From the *Default Project* drop-down select the placeholder class library project.
4. In the **Package Manager Console**, enter:

```
Update-Package ReadyRoll.MSBuild
```

5. Close and re-open the solution. *This is needed in order for the upgrade to take effect within the database projects.*
6. To confirm that the build components have been successfully upgraded, **Rebuild Solution**, then in the **Output** window, check that the ReadyRoll version matches that of the upgraded NuGet package.

> To locate the version number, in the output text look for the file path that follows the string **"Using ReadyRoll toolpath".** If the version number does not match the upgraded NuGet package, check that the *ReadyRollTargetsPath* element is defined exactly as specfied above in the project file, and that there are no other versions of **ReadyRoll.MSBuild** present in the *packages* folder.

> **Getting started**
>
> For details on how to get started:
>
> - If you are using VSTS/TFS, see Create VSTS/TFS build
> - For anything else, see Continuous integration