# Walkthrough #1

In this walkthrough, we're going to build a command configuration file that will do the following:

1. Update a web app with the latest versions from an SVN repo.

2. Rebuild the updated web app.

3. Publish the web app.

4. Zip the web app.

Prerequisites:

1. CommandCenter repo (<https://github.com/Jabestrada/CommandCenter>) downloaded
2. CommandCenter.sln opened in Visual Studio (tested in VS 2019 only) and startup project set to CommandCenter.UI.WinForms
3. Svn.exe
4. 7Zip.exe
5. Local SVN repo copy

## Step 1

Create XML file "PackageApp.config" with the following initial contents:

<?xml version="1.0" encoding="utf-8" ?>

<commandCenter>

<commands>

</commands>

</commandCenter>

## Step 2

Within the <commands> node in PackageApp.config, add the highlighted entry to represent the svn update command; update the svnExe and directoryToUpdate values accordingly:

<?xml version="1.0" encoding="utf-8" ?>

<commandCenter>

<commands>

<command>

<typeName>CommandCenter.Commands, CommandCenter.Commands.Svn.SvnUpdateCommand</typeName>

<shortDescription>Update staging folder with latest source code versions</shortDescription>

<ctorArgs>

<ctorArg name='svnExe' value='C:\Program Files\TortoiseSVN\bin\svn.exe' />

<ctorArg name='directoryToUpdate' value='C:\FullLocalPathToRepo' />

</ctorArgs>

</command>

</commands>

</commandCenter>

N.B. ctorArg name attribute values can be anything as long as they are unique within their <ctorArgs> parent node. However, they have to be in the correct listing order since these are passed as constructor parameters to the Command during object instantiation.

## Step 3

Add the MsCleanAndRebuildCommand after the SvnUpdateCommand to represent the step to build the web app:

<?xml version="1.0" encoding="utf-8" ?>

<commandCenter>

<commands>

<command>

<typeName>CommandCenter.Commands, CommandCenter.Commands.Svn.SvnUpdateCommand</typeName>

<shortDescription>Update staging folder with latest source code versions</shortDescription>

<ctorArgs>

<ctorArg name='svnExe' value='C:\Program Files\TortoiseSVN\bin\svn.exe' />

<ctorArg name='directoryToUpdate' value='C:\FullLocalPathToRepo' />

</ctorArgs>

</command>

<command>

<typeName>CommandCenter.Commands, CommandCenter.Commands.MsBuild.MsCleanRebuildCommand</typeName>

<shortDescription>Clean and rebuild solution</shortDescription>

<ctorArgs>

<ctorArg name='vsExe' value='C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\MSBuild\Current\Bin\MsBuild.exe' />

<ctorArg name='solutionFile' value='C:\ FullLocalPathToRepo\MySolution.sln' />

<ctorArg name='configuration' value='Release' />

</ctorArgs>

</command>

</commands>

</commandCenter>

As before, adjust vsExe and solutionFile values accordingly. At this point, one would notice that ‘C:\FullLocalPathToRepo’ has been referenced twice. To prevent typo errors and to simplify the configuration file, one can define a token once, and then use the token in lieu of the value that it represents. The modified .config file would look like this:

<?xml version="1.0" encoding="utf-8" ?>

<commandCenter>

<tokens>

<token key=”[SOURCE\_DIR]” value=” C:\FullLocalPathToRepo” />

</tokens>

<commands>

<command>

<typeName>CommandCenter.Commands, CommandCenter.Commands.Svn.SvnUpdateCommand</typeName>

<shortDescription>Update staging folder with latest source code versions</shortDescription>

<ctorArgs>

<ctorArg name='svnExe' value='C:\Program Files\TortoiseSVN\bin\svn.exe' />

<ctorArg name='directoryToUpdate' value='[SOURCE\_DIR]’ />

</ctorArgs>

</command>

<command>

<typeName>CommandCenter.Commands, CommandCenter.Commands.MsBuild.MsCleanRebuildCommand</typeName>

<shortDescription>Clean and rebuild solution</shortDescription>

<ctorArgs>

<ctorArg name='vsExe' value='C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\MSBuild\Current\Bin\MsBuild.exe' />

<ctorArg name='solutionFile' value='[SOURCE\_DIR]\MySolution.sln' />

<ctorArg name='configuration' value='Release' />

</ctorArgs>

</command>

</commands>

</commandCenter>

There are no strict conventions for defining tokens but one would have to ensure that tokens are uncommon enough so that no unintended token substitution happens.

## Step 4

Append MsPublishWebAppCommand to publish the web app.

<?xml version="1.0" encoding="utf-8" ?>

<commandCenter>

<tokens>

<token key=”[SOURCE\_DIR]” value=” C:\FullLocalPathToRepo” />

</tokens>

<commands>

<command>

<typeName>CommandCenter.Commands, CommandCenter.Commands.Svn.SvnUpdateCommand</typeName>

<shortDescription>Update staging folder with latest source code versions</shortDescription>

<ctorArgs>

<ctorArg name='svnExe' value='C:\Program Files\TortoiseSVN\bin\svn.exe' />

<ctorArg name='directoryToUpdate' value='[SOURCE\_DIR]’ />

</ctorArgs>

</command>

<command>

<typeName>CommandCenter.Commands, CommandCenter.Commands.MsBuild.MsCleanRebuildCommand</typeName>

<shortDescription>Clean and rebuild solution</shortDescription>

<ctorArgs>

<ctorArg name='vsExe' value='C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\MSBuild\Current\Bin\MsBuild.exe' />

<ctorArg name='solutionFile' value='[SOURCE\_DIR]\MySolution.sln' />

<ctorArg name='configuration' value='Release' />

</ctorArgs>

</command>

<command>

<typeName>CommandCenter.Commands, CommandCenter.Commands.MsBuild.MsPublishWebAppCommand</typeName>

<shortDescription>Publish web app</shortDescription>

<ctorArgs>

<ctorArg name='exe' value='C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\MSBuild\Current\Bin\MsBuild.exe' />

<ctorArg name='project' value='[SOURCE\_DIR]\MyProject.csproj' />

<ctorArg name='configuration' value='Release' />

<ctorArg name='publishProfile' value='FolderProfile.pubxml' />

</ctorArgs>

</command>

</commands>

</commandCenter>

At this point, one would notice that the full path of MsBuild.exe has been referenced twice. As before, we could simplify the configuration file by defining a token, resulting to the config file below:

<?xml version="1.0" encoding="utf-8" ?>

<commandCenter>

<tokens>

<token key=”[SOURCE\_DIR]” value=” C:\FullLocalPathToRepo” />

<token key=”[MSBUILD]” value=”C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\MSBuild\Current\Bin\MsBuild.exe” />

</tokens>

<commands>

<command>

<typeName>CommandCenter.Commands, CommandCenter.Commands.Svn.SvnUpdateCommand</typeName>

<shortDescription>Update staging folder with latest source code versions</shortDescription>

<ctorArgs>

<ctorArg name='svnExe' value='C:\Program Files\TortoiseSVN\bin\svn.exe' />

<ctorArg name='directoryToUpdate' value='[SOURCE\_DIR]’ />

</ctorArgs>

</command>

<command>

<typeName>CommandCenter.Commands, CommandCenter.Commands.MsBuild.MsCleanRebuildCommand</typeName>

<shortDescription>Clean and rebuild solution</shortDescription>

<ctorArgs>

<ctorArg name='vsExe' value='[MSBUILD]’ />

<ctorArg name='solutionFile' value='[SOURCE\_DIR]\MySolution.sln' />

<ctorArg name='configuration' value='Release' />

</ctorArgs>

</command>

<command>

<typeName>CommandCenter.Commands, CommandCenter.Commands.MsBuild.MsPublishWebAppCommand</typeName>

<shortDescription>Publish web app</shortDescription>

<ctorArgs>

<ctorArg name='exe' value='[MSBUILD]’/>

<ctorArg name='project' value='[SOURCE\_DIR]\MyProject.csproj' />

<ctorArg name='configuration' value='Release' />

<ctorArg name='publishProfile' value='FolderProfile.pubxml' />

</ctorArgs>

</command>

</commands>

</commandCenter>

## Step 5

Add Zip7CompressCommand to zip the published web app.

<?xml version="1.0" encoding="utf-8" ?>

<commandCenter>

<tokens>

<token key=”[SOURCE\_DIR]” value=” C:\FullLocalPathToRepo” />

<token key=”[MSBUILD]” value=”C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\MSBuild\Current\Bin\MsBuild.exe” />

</tokens>

<commands>

<command>

<typeName>CommandCenter.Commands, CommandCenter.Commands.Svn.SvnUpdateCommand</typeName>

<shortDescription>Update staging folder with latest source code versions</shortDescription>

<ctorArgs>

<ctorArg name='svnExe' value='C:\Program Files\TortoiseSVN\bin\svn.exe' />

<ctorArg name='directoryToUpdate' value='[SOURCE\_DIR]’ />

</ctorArgs>

</command>

<command>

<typeName>CommandCenter.Commands, CommandCenter.Commands.MsBuild.MsCleanRebuildCommand</typeName>

<shortDescription>Clean and rebuild solution</shortDescription>

<ctorArgs>

<ctorArg name='vsExe' value='[MSBUILD]’ />

<ctorArg name='solutionFile' value='[SOURCE\_DIR]\MySolution.sln' />

<ctorArg name='configuration' value='Release' />

</ctorArgs>

</command>

<command>

<typeName>CommandCenter.Commands, CommandCenter.Commands.MsBuild.MsPublishWebAppCommand</typeName>

<shortDescription>Publish web app</shortDescription>

<ctorArgs>

<ctorArg name='exe' value='[MSBUILD]’/>

<ctorArg name='project' value='[SOURCE\_DIR]\MyProject.csproj' />

<ctorArg name='configuration' value='Release' />

<ctorArg name='publishProfile' value='FolderProfile.pubxml' />

</ctorArgs>

</command>

<command>

<typeName>CommandCenter.Commands, CommandCenter.Commands.FileZip.Zip7CompressCommand</typeName>

<shortDescription>Zip publish folder</shortDescription>

<ctorArgs>

<ctorArg name='exe' value='C:\Program Files\7-Zip\7z.exe' />

<ctorArg name='targetZip' value='C:\MyPublishFolder\MyPackagedApp.7z' />

<ctorArg name='source1' value='[SOURCE\_DIR]’ />

</ctorArgs>

</command>

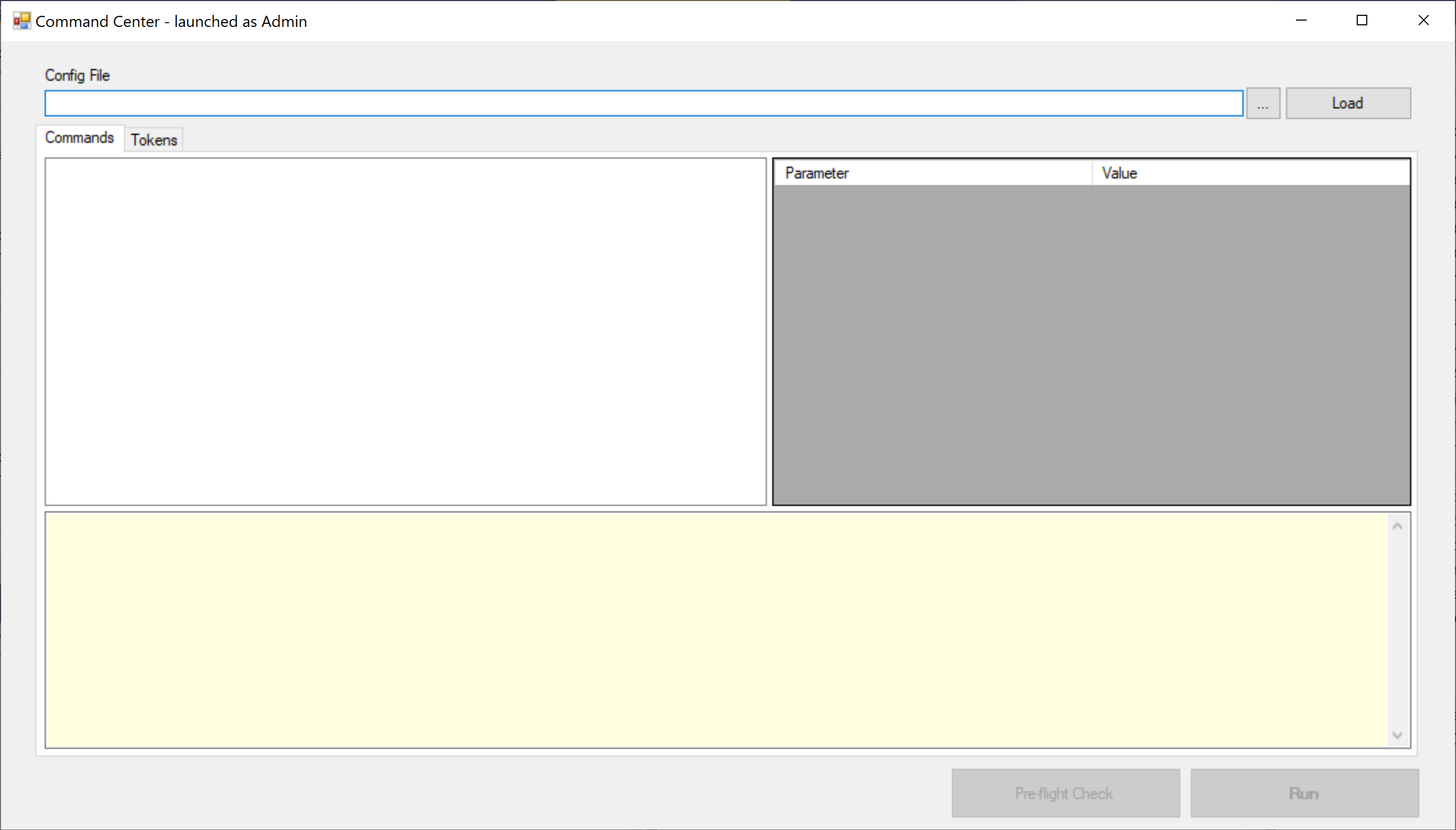
</commands>

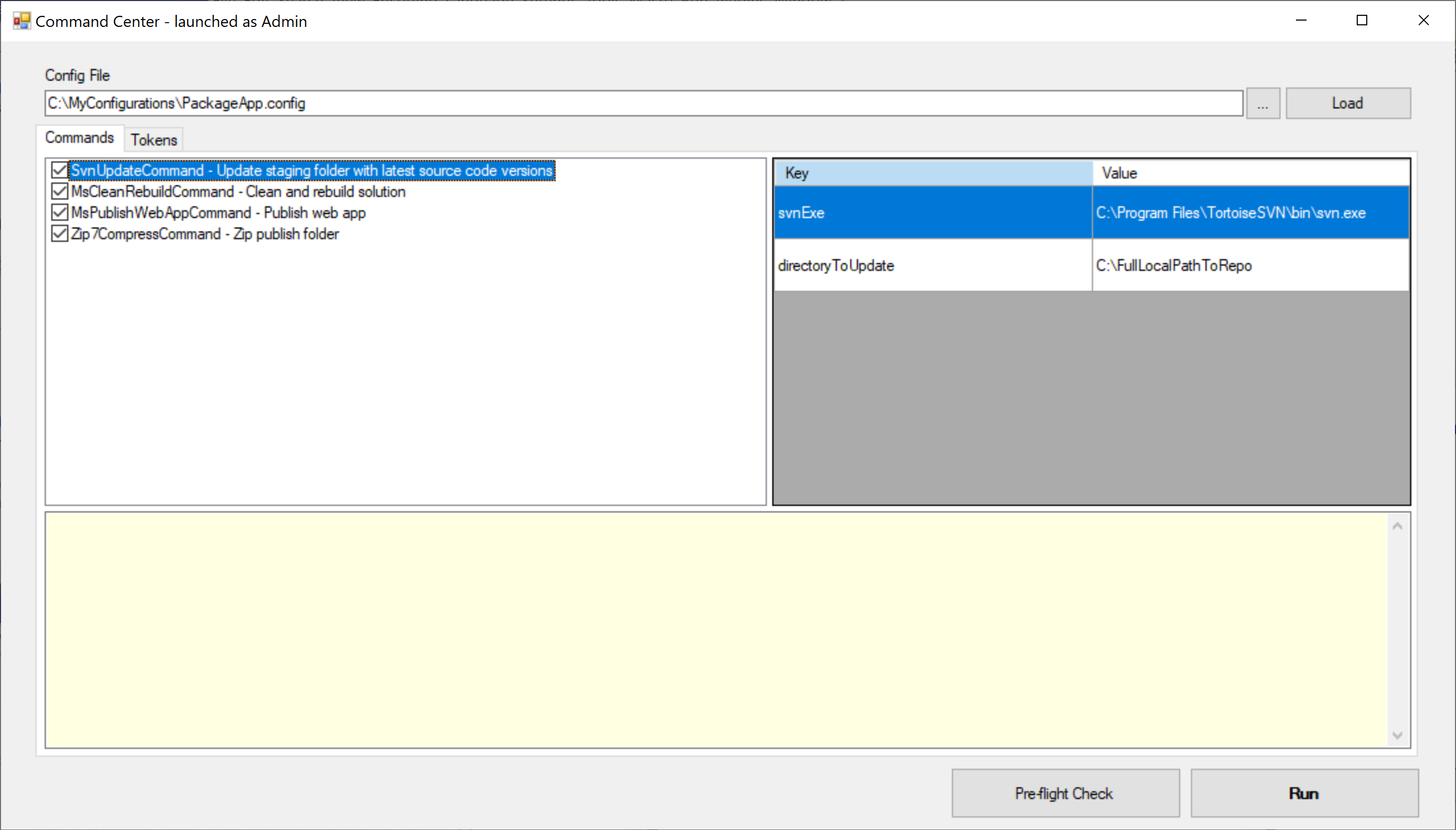
</commandCenter>

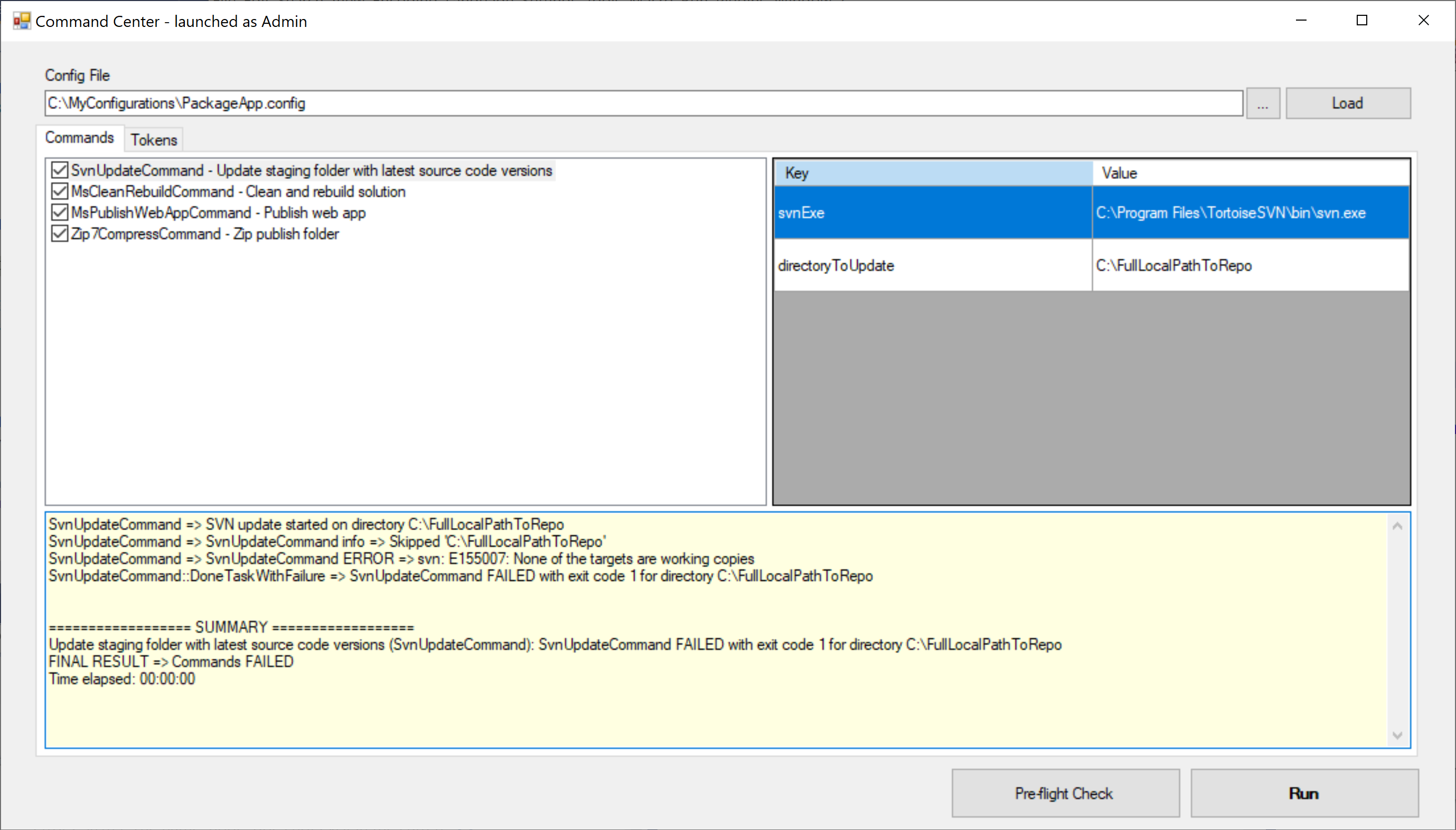
## Step 6

At this point, we are ready to test our new configuration file.

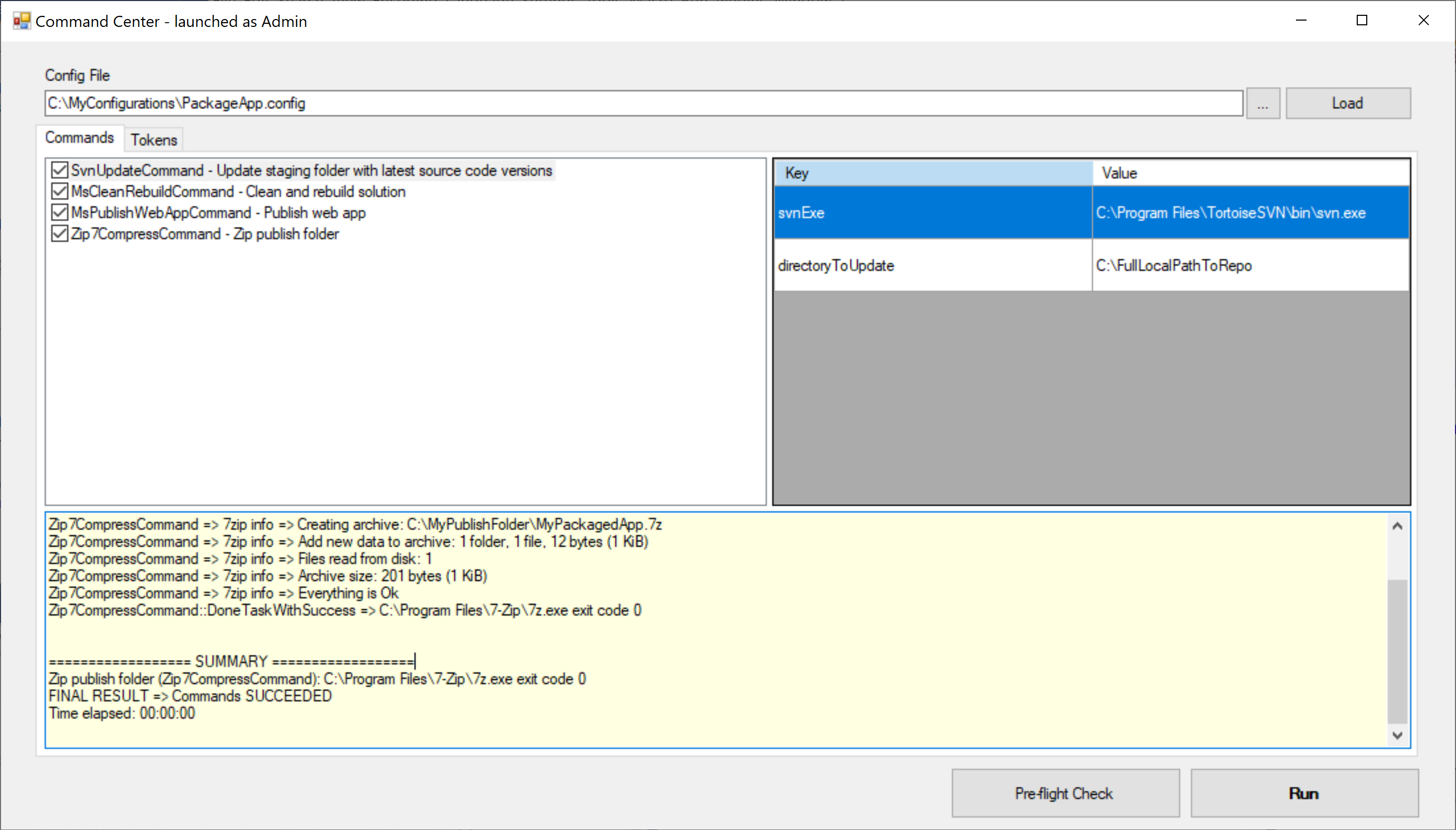
1. Run CommandCenter solution in Visual Studio, ensuring that CommandCenter.UI.WinForms is the startup project:

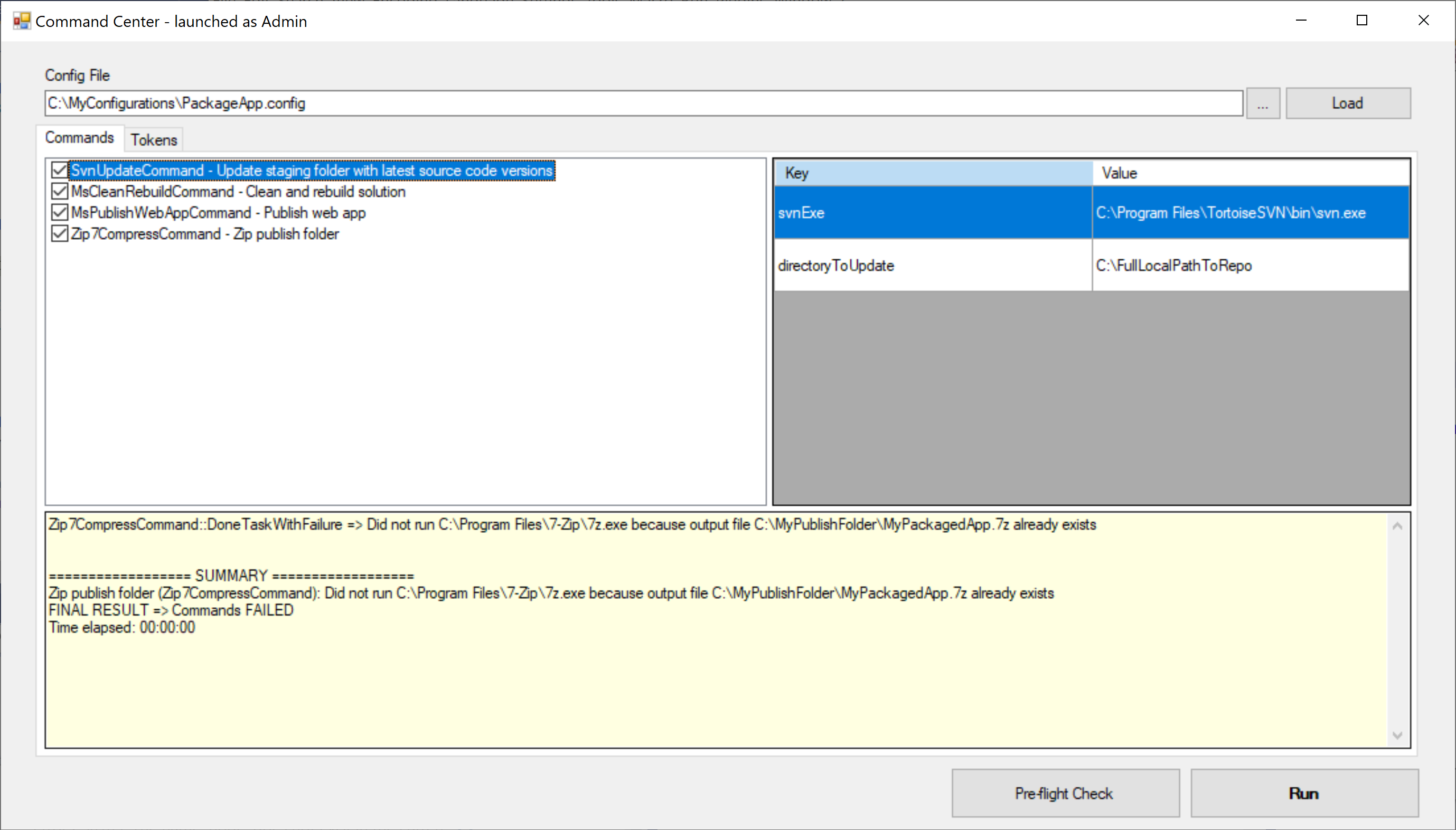


1. Click on **Browser for config file** button (next to **Load** button) and select PackageApp.config from the file system:  
     
   
2. Click on **Run** button to execute the defined commands. The status reports from the executed commands will display in the status window:



Below is a sample output where the command chain succeeded:



However, re-running the same command will fail on Zip7CompressCommand because it’s been designed to do so if the target zip file already exists:   
  


We can fix this failure by inserting a DeleteDirectoryContentsCommand before the MsPublishWebAppCommand (see next Step)

## Step 7

Insert a DirectoryDeleteContentsOnlyCommand right before the MsPublishWebAppCommand to ensure that the published files and target zip file do not exist before the publishing and file compression begins:

<?xml version="1.0" encoding="utf-8" ?>

<commandCenter>

<tokens>

<token key=”[SOURCE\_DIR]” value=” C:\FullLocalPathToRepo” />

<token key=”[MSBUILD]” value=”C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\MSBuild\Current\Bin\MsBuild.exe” />

</tokens>

<commands>

<command>

<typeName>CommandCenter.Commands, CommandCenter.Commands.Svn.SvnUpdateCommand</typeName>

<shortDescription>Update staging folder with latest source code versions</shortDescription>

<ctorArgs>

<ctorArg name='svnExe' value='C:\Program Files\TortoiseSVN\bin\svn.exe' />

<ctorArg name='directoryToUpdate' value='[SOURCE\_DIR]’ />

</ctorArgs>

</command>

<command>

<typeName>CommandCenter.Commands, CommandCenter.Commands.MsBuild.MsCleanRebuildCommand</typeName>

<shortDescription>Clean and rebuild solution</shortDescription>

<ctorArgs>

<ctorArg name='vsExe' value='[MSBUILD]’ />

<ctorArg name='solutionFile' value='[SOURCE\_DIR]\MySolution.sln' />

<ctorArg name='configuration' value='Release' />

</ctorArgs>

</command>

<command>

<typeName>CommandCenter.Commands, CommandCenter.Commands.FileSystem.DirectoryDeleteContentsOnlyCommand</typeName>

<shortDescription>Delete contents of publish folder</shortDescription>

<ctorArgs>

<ctorArg name='backupDir' value='C:\MyBackupFolder’ />

<ctorArg name='dir1' value='C:\MyPublishFolder’ />

</ctorArgs>

</command>

<command>

<typeName>CommandCenter.Commands, CommandCenter.Commands.MsBuild.MsPublishWebAppCommand</typeName>

<shortDescription>Publish web app</shortDescription>

<ctorArgs>

<ctorArg name='exe' value='[MSBUILD]’/>

<ctorArg name='project' value='[SOURCE\_DIR]\MyProject.csproj' />

<ctorArg name='configuration' value='Release' />

<ctorArg name='publishProfile' value='FolderProfile.pubxml' />

</ctorArgs>

</command>

<command>

<typeName>CommandCenter.Commands, CommandCenter.Commands.FileZip.Zip7CompressCommand</typeName>

<shortDescription>Zip publish folder</shortDescription>

<ctorArgs>

<ctorArg name='exe' value='C:\Program Files\7-Zip\7z.exe' />

<ctorArg name='targetZip' value='C:\MyPublishFolder\MyPackagedApp.7z' />

<ctorArg name='source1' value='[SOURCE\_DIR]’ />

</ctorArgs>

</command>

</commands>

</commandCenter>

Re-loading the revised PackageApp.config should now look as follows, and should succeed when re-run:

