Transition of the ControlPoint Build and Release.

This document describes the set of build scripts used to build the ControlPoint kits and how to run a build from the command line.

**How to build ControlPoint kits**

1. ControlPoint Build Systems:

The ControlPoint build scripts are located on 3 systems, (CPBuild, CPBuild2010, and CPBuild2013). When doing a build, login to the local account "Administrator" with password "demo". It is the same for all 3 systems.

1. ControlPoint Build Scripts:

Currently, ControlPoint kits are built using a number of top level scripts located in the c:\ControlPoint\_Build folder tree of each build system. All kits built are placed in a subfolder off the "\\axstore\development\builds\controlpoint" directory. The following is a list of current build script files and the kits that they build:

**BuildBStone.bat** - builds the 2010 and 2013 On-Prem kit from the main trunk branch (ControPointDev) and all built kits are placed in the subfolder "BackStone\_nnnn" where nnnn is the SharePoint version number it's for.

**BuildBStoneNative.bat** - builds a 2010 or 2013 kit from the trunk branch (ConrolPointDev) and place the kit in the "BlackStone\_nnnn\_Native" subfolder. Same convention as above.

**BuildCPOnline.bat** - builds a "Web Services" flavor of the ControlPoint Online kit for 2013 only. Can be used for either BlackStone or Houdini releases. . For BlackStone, the target kit location is “BlackStone\_2013-Online”. For Houdini the kit will be placed in the "Houdini\_2013-Online" subfolder

**BuildShell-Native\_MR.bat** - builds a ControlPoint 2010 5.0.1 Native kit from the (CPShell-Native\_MR) branch and places the kit in the "CP5.0.1-MR\_2010\_Native" subfolder.

\*\* All build scripts are checked in to TFS under the “ControlPoint\_Build” branch. This document included. \*\*

1. Build Script command syntax:

With the exception of the BuildCPOnline script, all of the above scripts have the following syntax:

c:\ControlPoint\_Build> <buildscript> <kit\_name> <spVersionNUmber>

buildscript - name of one of the above build script.

kit\_name - the filename of the resulting kit you would like to be created in the target subfolder if the build was successful. (08-08-2013, Shell\_08-07-2013)

\*\* The Native build script will add a “\_N” suffix to the kit name and the Online build will have “\_Online ”appended to the kit name.

spVersionNumber - one of the following: (2007, 2010, or 2013)

Example: BuildBStone 12-21-2012 2013

The above line will build a Blackstone 2013 kit off the (ControlPointDev) branch and places the kit in \\axstore\development\builds\controlpoint\BlackStone\_2013 with the folder name of "ControlPoint\_12-21-2012".

\*\* The third param needs to correspond with SharePoint version and the system on which you run the build. \*\*\*

The BuildCPOnline script can be used to build either a Houdini or BlackStone kit. The syntax is:

c:\ControlPoint\_Build> BuildCPOnline <kit\_name> <kit type>

kit\_name – (The same as the above scripts.)

kit type - (Houdini) or (Copperfield)

1. Post build steps:

After a kit has been successfully built, I will usually update the InternalReleaseReadme.xlsx file. To do that, just take the "Master" InternalReleaseReadme.xlsx file in the [\\axstore\development\builds\controlpoint\Blackstone\_2010](file:///\\axstore\development\builds\controlpoint\Blackstone_2010) folder and edited it as follow.

Open the "Master" InternalReleaseReadme.xlsx file for the kit that you are announcing. Then from "Source Control Explorer" in VS, right click on the source branch that the kit was built from and select "View History". In the "History" tab, select all changesets from the date of the last build to current and do a <ctrl/c> to copy them into the clipboard.

In the Excel application, select the "Slated for next build" worksheet and right click with the cursor on the first row. Select the "Paste Special..." item and click "OK" on all dialog boxes that come up. Select all rows on that worksheet and do a <ctl/x> to cut the rows. Change the worksheet to the "Current build" and right click with the cursor on row 2.

Select the "Insert Cut Cells" item. This will move the new entries to this worksheet. Finally, Save and Close the spreadsheet. Copy the updated xlsx file into the newly created kit for that branch on \\Axstore.

1. Announcing the kits:

To announce the kit, open an existing email announcement from the previous kit of the same type and enter "Reply All". Change the Subject text and the kit links and then "Send" it. You are all done.

1. Possible Build Errors:

The following are some possible errors that I’ve encountered and their explanation.

* From time to time, you may get "too many symbols ..." error in the DOS session used to run the manual build. This is because the maximum number of variables defined in one DOS session has been exceeded. Probably from the days of builds in that session. I will just close the session and start a new DOS Prompt session.
* On the CPBuild2010 system, try not to install the Shell kit. Doing so will placed our xc...dlls into the GAC. As a result, when a batch build gets run VS will see those dlls being vailable; hence not to create them. So, the kit generated will be missing the xc...dlls. (Case, a bad kit.) If you did not an install, just remove or uninstall it before the evening so the nightly build does not fail.
* Make sure the "Dev folders" (c:\SharePointDev and HIVE\Axceler) are not locked by a DOS session or a "File Explorer", otherwise the build might fail.
* Because the build dynamically shares the "Dev folders" across builds for all TFS branchs, do not try to check-in source files using any of the build system.

1. Shell Kits:

The AxcelerFeatures.wsp file is created differently from the normal way. The following highlights the major differences. This was done to support deployment into an existing WAP and without using a virtual directory. (All changes are inside the xcDiscoveryFeature project.)

* Because the build dynamically shares the "Dev folders" across builds for all TFS branchs, do not try to check-in source files using any of the build system.
* The manifest.xml file is not created by the WSPBuild.exe utility. This file is located in \WSPBuilder\AxcelerFeatures folder.
* The placement of the files in the \WSPBuilder\12 HIVE is done using the MakeWSP\_Claims.cmd script and the "CabArc.exe" utility is used to create the .WSP file instead of the WSPBuild.exe utility.

The manual installation (also called the MSOCAF install) piece of the Shell Kit is packaged in the "MSOCAF\_Kit" folder of the Shell kit at the same level as the \Disk1 folder. It has the MSOCAF required folder structure for going through the certification and it has all the files to do a complete manual installation of the ControlPoint for DedicatedOnline as well as an OnPremises (Classic) environment. There is a \MSOCAF folder in TFS which hold all auxcillary files needed for the certification. (\*\* The BUILDSHELL… script will create a complete kit that has both the InstallShield and MSOCAF kits. \*\*)

# Build Scripts:

The build scripts are divided into the following categories:

1. Execute scripts.
2. Release specific scripts.
3. Generic Utiltity scripts for all 3 sharepoint environments. (2007. 2010, and 2013).
4. SharePoint specific Utility Scripts

**Execute scripts:**

The “Execute” scripts are the controller that uses scripts from the two sets of Utility scripts to build the ControlPoint Kit. There are 2 “Execute” scripts:

* **ExecuteBuild.bat** : Builds the ControlPoint On-Prem and Native kits.
* **ExecuteHostedBuild .bat**: Builds the ControlPoint Online kits.

This is what the “ExecuteBuild.bat” does by using the appropriate .bat file.

1. Setup SP environment specific variables. (**SetEnvironmentVariables.bat**)
2. Performs an IISReset.
3. Clean the project work directories. (**RefreshProjectDirectories.bat**)
4. Get sources from TFS. (**GetLatestTFSbranch.bat**)
5. Prepares the SP2010 web.config file and the WspBuilderHelp folder. (**Prepare2010FolderForCompile.bat**)
6. Set all source files to writable. (**MakeWritable.bat**)
7. Setup the correct Telerik files for Native/Shell builds.
8. Enable or disable obfuscation.
9. Set the kit version ident. (**Deployment.exe**)
10. Copy the appropriate version of the documentation. (**CopyManuals.bat**)
11. Compile the ControlPoint VS solutions. (**CompileSolutionsVSnnnn.bat**)
12. Build the Online help .wsp solution. (**CopyHelp.bat**)
13. Create the ControlPoint kit with Install Shield. (**CompileInstallShieldProjects.bat**)
14. Digitally sign all .exe and .dll files. (**SignManualInstallImages.bat**)
15. Copy kit files into drop location on [\\Axstore](file:///\\Axstore). (**CopyBuildProducts.bat**)
16. Create the MSOCafkit for Native builds. (**CreateMSOCafKit.bat**)

The “ExecuteHostedBuild.bat” performs the following in building the Online kit.

1. Setup SP environment specific variables. (**SetEnvironmentVariables.bat**)
2. Performs an IISReset.
3. Clean the project work directories.
4. Get sources from TFS. (**GetLatestTFSbranch.bat**)
5. Prepares the SP2010 web.config file and the WspBuilderHelp folder. (**Prepare2010FolderForCompile.bat**)
6. Set all source files to writable. (**MakeWritable.bat**)
7. Setup the correct Telerik files for Native/Shell builds.
8. Enable or disable obfuscation.
9. Set the kit version ident. (**Deployment.exe**)
10. Compile the “xcSandBoxSolution” using the CPDevHosted2010.sln file.
11. Compile the CP online client files using ControlPointBuildxxxxOnline.sln file.
12. Copy the CP online documentation from “\\axsource\sourcsafe.
13. Create the ControlPoint kit with Install Shield. (**CompileInstallShieldProjects.bat**)
14. Digitally sign all .exe and .dll files. (**SignManualInstallImages.bat**)
15. Copy kit files into drop location on [\\Axstore](file:///\\Axstore). (**CopyHostedBuildProducts.bat**)

Both of these build scripts take 8 command line input. Out of them only 4 are being used for controlling what kit is to be build. They are p1, p5, p6, and p8.

P1 – the name of the TFS branch to build from. If “TFS”, the ControlpointDev is used. Also there must be a TFS Workspace with the name <branchname><spver> defined on each of the build machine. (i.e. BlackStoneOnline2013).

P5 – name of the subfolder in the [\\axstore](file:///\\axstore)... Location to which the kit is to be placed.

P6 – the Sharepoint version of the kit to be build.

P8 – the VisualStudio version on the build system to use for compile.

**Release specific scripts:**

The release specific scripts are used to make manual builds easier by allow user to enter a simpler set of inputs. These scripts have a prefix of “Build…bat”. The following are currently being used.

* BuildBStone.bat : build the BlackStone release kits. (off the trunk)
* BuildCPOnline.bat : build the CPonline kit for BlackStone
* BuildBStoneNative.bat : build the BlackStone Native kit.
* BuildShell-Native\_MR.bat: build the Native kit from the “Shell-Native\_MR” branch.

There are other such scripts in TFS and on the build systems which are obsoleted and can be used as a template for creating new ones. The input to these scripts are specify in the “How to build ControlPoint kits” section in this document. Within these script, the following global variables are setup to be used by the by the GetLatestTfsBranch.bat.

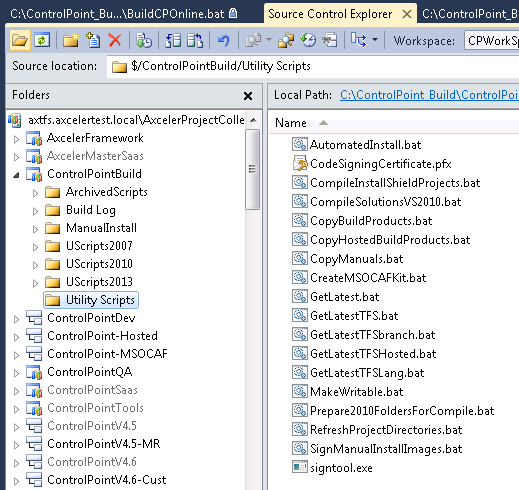
* cpBranch = name of the branch/workspace definition to use. (get from trunk if blank)
* GetFromLabel = name of the label within the branch get the sources from.
* Docver = version location of doc files from [\\axsource\sourcesafe\](file:///\\axsource\sourcesafe\).... Folder to copy. (i.e. “5.1” represents folder location from” [\\AxSource\SourceSafe\ControlPointOnline5.1](file:///\\AxSource\SourceSafe\ControlPointOnline5.1)”.

Both the “Execution” and “Build…” scripts are location in the c:\Controlpoint\_Build folder of each build system and are kept in the top level of the “ControlPointBuild” project branch in TFS. In addition, there are 3 python scripts file in this directory used to drive the nightly buiids. The following are their filenames:

* **Automate\_build\_2007.py**
* **Automate\_build\_2010.py**
* **Automate\_build\_2013.py**

**Generic Utility Scripts:**

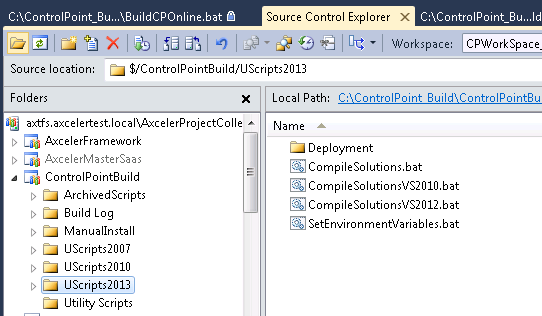
The following are a list of the Generic Utility scripts in the TFS location “$/ControlPointBuild/Utility Scripts”.



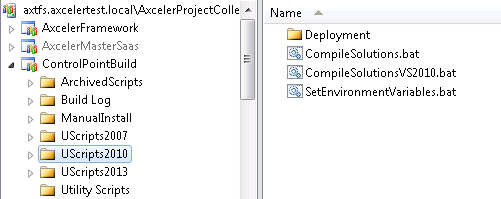
SharePoint Specific Utility Scripts:

Here are the SharePoint Specific scripts for each of the SP Versions.

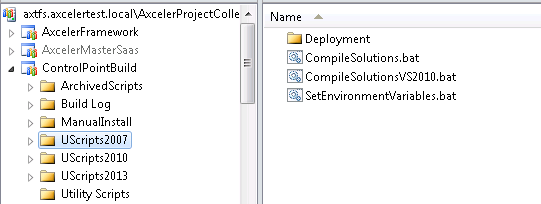
(SharePoint 2013)



(SharePoint 2010)



(SharePoint 2007)



# Nightly Builds:

The nightly build runs automatically between 12:00 midnight and 2:00 am. Currently the .py python scripts are started on build systems CPBuild2010 and CPBuild2013. Within the .py scripts, it calls the “BuildBStone.bat”, “BuildBStoneNative.bat”, and “BuildCPOnline.bat” scripts to build 2013 (On-Prem and Online) and 2010 (On-Prem and Native) kits. The results are placed in the [\\Axstore\development\ControlPoint\BlackStone\_xxxx](file:///\\Axstore\development\ControlPoint\BlackStone_xxxx) folders. The name given to the kit folder have the string “Nightly” in them.

For all On-Prem kits, the Change Manager Product kit is also included. The Change Manager kit is copied for a designated location in the following network drive of the Cybage team.

“[\\lee.northsydney.winapp.com.au\ChangeManagerCurrentRelease\](file:///\\lee.northsydney.winapp.com.au\ChangeManagerCurrentRelease\)“

The Cybage team will place their latest release kit into these folder so it will be automatically included in our kits no matter it is a nightly or release build. The installer can accommodate any version of the Change Manager kit. (\*Copying from the above location can take a long time. We may want to have a cache copy locally.)

# Automated install and test:

The nightly builds when successful will be automatically installed on the build machines, CPBuild2010 and CPBuild2013. In the c:\CP\_AutomatedInstall folder of each build machine,