NOTE: “Build” does not mean “Rebuild”. Don’t ever “Rebuild”.

Preparing for a staging release:

Inspect the Sample-Live-Sync directory hierarchy and remove any \*.pdb and \*.zip files, and any other non-public files.

Exit Visual Studio in the Cloud\_SDK solution.

Check out the master branch:

* Make sure that all current work has been committed and staged to master.
* Make sure you are current on the master branch.

Pull master.

Make a new branch for the release:

* git branch 20130218A0Release0\_1\_6
* git checkout 20130218A0Release0\_1\_6
* git push -u origin 20130218A0Release0\_1\_6
* To delete a branch:
  + Git branch - -d <branch>
  + Git push origin - -delete <branch

Start Visual Studio in the Cloud\_SDK solution.

Change to the ReleaseSampleAppOnly configuration.

Change the define in CLDefinitions from development cliff servers to the cloud.com servers. (CloudApiPublic\Static\CLDefinitions: define PRODUCTION\_BACKEND.

Increment the version build number of any 3rd party DLL built from source if it was changed, but this should have been done during normal development by the time it got to master.

Change the Sample-Live-Sync, CloudApiPublic and CloudSetupSdkSyncSample, and CloudSetupSdkSyncSampleSupport Assembly and File Versions to the current release. E.g., 0.1.2.0. For CloudSetupSdkSyncSample, click Project Assistant, and then Application Information. Set the application version.

Set the project for signing (CloudApiPublic, CloudSetupSdkSyncSampleSupport, CloudSetupSdkSyncSample). C:\CertBackup\CloudSigning

For signing CloudSetupSdkSyncSample - > open the project tree in VS Solution explorer -> select the Prepare for release node and double click “releases” Then in the Releases window Select the Single Image Node.

This will bring up a new screen with a tab for “Signing” Digital Certificate File should point to the same ptx file as is used in the other signed projects – Cert Password is stored in desk, and Sign Output Files to “Setup.exe and Windows Installer Package

Close Visual Studio.

Review the changes before ther push to validate whats being added

Push the release branch to remote. DO NOT CHECK-IN SIGNING FILES\* TO GITHUB!!!!!! Including .pfx files copied to the projects where you changed the settings. These should be ignored by gitignore, but remove them anyway.

\* By SIGNING FILES I do not mean the changes to the projects to turn on signing, but instead the certificates or keys

Staging release procedure:

Start Visual Studio in the CloudSDK solution.

Run REBUILD (not a typo) on the project ZGacUninstall. That will remove all of our old DLLs from the GAC.

In Debug solution configuration, choose the clean solution selection from the build menu. Check that it succeeded.

In ReleaseSampleAppOnly solution configuration, choose the clean solution selection from the build menu. Check that it succeeded.

Stay in ReleaseSampleAppOnly solution configuration.

Open the CloudApiPublic properties pane and select “Obfuscate this assembly”, and set the “Password of strong name key file”.

Check the CloudApiPublic and Sample-Live-Sync references(CloudApiPublic should appear with no warning symbol for the BadgeCOMLib reference, but Sample-Live-Sync should actually have a warning symbol on the Cloud reference since its correct version requirement should be for the new version of CloudApiPublic which hasn’t been built yet)

* Unload the CloudApiPublic project, edit the XML project file and change the BadgeComLib version to the proper version. e.g.: 0.1.8.0. and remove any indication that the specific version is not set to false (may just delete the line)
* Reload the CloudApiPublic project.
* In the CloudApiPublic project, make sure that specific versions are selected for BadgeComLib and all of the .CSDK DLLs, and that the references specify the proper versions.
* Unload the Sample-Live-Sync project, edit the XML project file and change the Cloud.dll reference version to the proper version. e.g.: 0.1.8.0. Make it like “<Reference Include="Cloud, Version=0.2.8.0, Culture=neutral, PublicKeyToken=840eb90fa4503c3c, processorArchitecture=MSIL" />”
* Remove the SpecificVersion line for the Cloud.dll reference.
* Reload the Sample-Live-Sync project.
* Set the Sample-Live-Sync project as the Startup Project.
* In the Sample-Live-Sync project, make sure that specific versions are selected for Cloud.dll and all of the .CSDK DLLs, and that the references specify the proper versions. NOTE: The Cloud.dll reference will have a yellow exclamation mark because it has not been built yet.
* If they do not specify the proper versions (they probably won’t), unload the project and edit the version number manually.
  + If it asks you to remove the project from the InstallShield .isl file, always answer “NO”.
  + If you unloaded the project, reload the project after saving the changes. Answer “YES” to close the XML project window.
  + Unloading Sample-Live-Sync and reloading the project will switch the startup project solution. Make the Sample-Live-Sync the startup project again.

Build CloudApiPublic project, check for build success. Also check that obfuscation ran by inspecting the output window.

Switch to Debug solution configuration

Build Sample-Live-Sync project, check for build success

Switch to ReleaseSampleAppOnly solution configuration

Check the Sample-Live-Sync project references and make sure that the Cloud.dll reference is “specific version”, and that it is the correct version.

Build Sample-Live-Sync project, check for build success

Make sure that no Explorer or command window is open to any folder below c:\Cloud\CloudSDK-Windows\CloudSetupSdkSyncSample.

Build CloudSetupSdkSyncSample project and check for build success.

Check html\index.htm. Go through the classes and namespaces to make sure that no new internal classes or namespaces appear.

Copy CloudSdkSetup.exe from the setup project output to a convenient location like C:\

Resource hack the copied CloudSdkSetup.exe:

* Select the Icon Group -> 100 -> 0. Right-click and select Replace Resource. Click “Open file with new icon” and replace the resource with ~\Artwork\cloudForInstallShield.ico.
* Select Version Info -> 1 -> 0. Change the OriginalFilename field to an empty string in and compile the script.
* Save changes as CloudSdkSetup.exe in the same “copied to” location

In the previous VS Developer Command Prompt windows, change directory to where you placed the “copied to and modified” CloudSdkSetup.exe.

Run these commands in command prompt (replace <password> with the certificate password):

* signtool remove /c CloudSdkSetup.exe
* signtool sign /f C:\CertBackup\CloudSigning\CloudPlatformCodeSigning.pfx /p <password> CloudSdkSetup.exe
* cls to clear the screen.

Zip CloudSdkSetup.exe into a zip file with the naming convention CloudSDK-v0.1.2>.zip where the “0.1.2” is the “version.release.build”. The build should be incremented at each release. The others are incremented by management decision.

The zip file is the completed release.

* Copy the file to c:\Source\Projects\ArchivedCloudSdkReleases on the build machine.
* Then use search in win-client (or an automated tool) to gather the current .pdb files and move them into the ArchivedCloudSdkReleases as well in a “PDBs” directory.
* Also copy the release folder to a local backup drive on the build machine (\\10.3.0.28\Builds).
* Also copy the release folder to a network drive which is routinely backed up.

Delete the .pfx files copied to the CloudApiPublic and CloudSetupSdkSyuncSampleSupport projects where you changed the settings.

* To do this: “Rebuild” (OK in this case) the project “ZDeleteLicenseFilesOnly”.

Close the Visual Studio win-client solution.

Push the release branch to remote. DO NOT CHECK-IN SIGNING FILES\* TO GITHUB!!!!!! Including .pfx files copied to the projects where you changed the settings. These should be ignored by gitignore, but remove them anyway.

\* By SIGNING FILES I do not mean the changes to the projects to turn on signing, but instead the certificates or keys

Tag the last commit to the yyyymmddRelease0.2.8 branch with this release number (e.g., Release0.1.2).

* Delete a local and remote tag named 12345:
  + git tag -d 12345
  + git push origin :refs/tags/12345
* Create a local tag named 12345 and push it to remote:
  + git tag 12345
  + git push –tags

When the release has been tested, merge the release branch back into master, but don’t merge the signing changes or the switch to the cloudstaging.us URL.

* git checkout master
* git merge --no-commit --no-ff <BranchName>.
* Use GIT Extensions to selectively merge modules. The CloudSetupSdkSyncSample.isl file must be manually merged to remove the signing changes, but leave the version number change.
* git push

Test the following before release:

* COL private SQL CE ClientApiTest.exe program.
* Zach’s Windows client SDK functional test.
* Installation and operation on all supported Windows platforms.