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

Preparing for a staging release:

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

Make a new branch for the release:

* git checkout –b 20130218A0Release0\_1\_6 (if that doesn’t work, do git branch, git checkout).
* git push --set-upstream origin 20130304A0Release0\_1\_6

Change the define in CLDefinitions from development cliff servers to the cloudstaging servers. (CloudApiPublic\Static\CLDefinitions: define STAGING\_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 BadgeCom, 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. For BadgeCom, edit the Resources\BadgeCom.rc Version field. Also, for BadgeCom in project properties edit the post build event: change the tlbimp command’s parameters asmversion and product version to match (the non-rem command which does not do delay-signing).

Set the project for signing (CloudApiPublic, CloudSetupSdkSyncSampleSupport, CloudSetupSdkSyncSample).

Test the application under Visual Studio to make sure it is working.

Close Visual Studio.

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.

Delete BadgeCOM.dll from 3rdParty\bin\release, and delete BadgeCOM.dll from C:\Cloud\CloudSDK-Windows\CloudSdkSyncSample\bin\Release\amd64.

In Debug solution configuration, clean solution. Check that it succeeded.

In ReleaseSampleAppOnly solution configuration, clean solution. Check that it succeeded.

In Release64 solution configuration, clean solution. Check that it succeeded.

Stay in Release64 solution configuration

Build BadgeCOM project, check for build success

Switch to ReleaseSampleAppOnly solution configuration

Build BadgeCOM project, check for build success

Open a Visual Studio 2012 Developer Command Prompt “as Administrator”.

Change directory to ~\3rdParty\bin\Release in command prompt

Run these commands in command prompt (use copy/paste to copy copy the lines directly from below. If copying commands from this document, check for a “blip” character at the start of the command in the command window. Delete it if it appears.) :

* (Note: Change the versions in the next line)
* tlbimp.exe BadgeCOM.dll /delaysign /publickey:C:\CertBackup\CloudSigning\CloudPlatformCodeSigning.pub /out:BadgeCOMLib.dll /company:Cloud.com /copyright:"Copyright (C ) Cloud.com. All rights reserved." /asmversion:0.1.6.1 /productversion:0.1.6.1
* sn.exe -R BadgeCOMLib.dll C:\CertBackup\CloudSigning\CloudPlatformCodeSigning.pfx
  + requires password to certificate
* copy /Y BadgeCOMLib.dll ..\..\..\CloudSdkSyncSample\bin\release\BadgeCOMLib.dll
* copy /Y BadgeCOMLib.dll ..\..\..\CloudSdkSyncSample\bin\debug\BadgeCOMLib.dll
* Make sure for the last two commands that one was “release” and the other was “debug”.

Open CloudApiPublic and change BadgeComLib Embed Interop Types property to false.

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)

* Make sure that specific versions are selected, and that the references specify the proper versions.
* 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

Obfuscate CloudApiPublic binary in CloudApiPublic\bin\Release and copy from CloudApiPublic\bin\Release\Obfuscated to CloudApiPublic\bin\Release, CloudSdkSyncSample\bin\Release, and CloudSdkSyncSample\bin\Debug

Build CloudSetupSdkSyncSampleSupport project, 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.

Switch to Debug solution configuration

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

Switch to ReleaseSampleAppOnly solution configuration

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

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

Build CloudSetupSdkSyncSample setup project, check for build success

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

Resource hack the copied CloudSdkSetup.exe to change the OriginalFilename field to an empty string in Version Info -> 1 -> 1033 and compile the script; then change the Icon Group -> 100 -> 0 by replacing resource with ~\Artwork\cloudForInstallShield.ico

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

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. This should be on a network drive which is routinely backed up.

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 master 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

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

* To do this: “Rebuild” the project “DeleteLicenseFilesOnly”.

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 Release0.1.5
* 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