AN1121: Headless Builds with Simplicity Studio v4

This document describes how to use Simplicity Studio's™ Python scripting capability to automate the building process from the command line.

There is also a reference section that describes the various functions that can be called from the Python script and the values for the function arguments. Examples are given for some basic scripts to perform builds in various scenarios.

KEY FEATURES OR KEY POINTS

- Required setup to perform headless builds
- Command line examples
- Listing of available Python scripts for supporting headless builds
- API guide for commands that can be issued from a Python script

Table of Contents

1. Requirements 3

2. Using the Script 4

2.1 Launch a Command Window 4

2.2 Using the <runScript.sh> or <runScript.bat> Script to Launch Simplicity Studio to Execute a Python Script 4

2.3 Available Example Python Scripts 4

2.3.1 <BuildExistingProject_v2.py> 4

2.3.2 <ImportAndGenerateIsc_v4Pub.py> 5

2.3.3 <ImportExistingProjectAndBuild.py> 5

2.3.4 <ImportExistingZippedProjectAndBuild.py> 6

2.3.5 <CopyProjectsAndBuild.py> 6

2.3.6 <CopySdkSettingsFromWorkspace.py> 7

3. API Definition of Available Commands 8

3.1 Arguments 9

3.2 Project-related Commands 10

3.2.1 getWorkspace() 10

3.2.2 getWorkspaceLocation() 10

3.2.3 getProjectsInWorkspace() 10

3.2.4 getOpenStudioProjectsInWorkspace() 10

3.2.5 getProject(String name) 10

3.2.6 getProjectForFile(String filePath) 11

3.2.7 getBuildTimeout() 11

3.2.8 isCompatibleWithPartOrGroup(Object project, Object part) 11

3.2.9 importExistingProject(String path, Map<String, Object> options, IProgressMonitor monitor) . .11

3.2.10 importSlsProject(String pathToProjectFile, Map<String, Object> options, IProgressMonitor monitor) 12

3.2.11 exportSlsProjFile(Object project, Map<String, Object> options, String pathToProjectFile) . .12

3.2.12 switchCurrentConfig(Object project, Object configuration, IProgressMonitor monitor) . . .13

3.2.13 buildProject(Object project, Map<String, Object> options, IProgressMonitor monitor) . . .13

3.2.14 cleanProject(Object project, IProgressMonitor monitor) 13

3.2.15 getProjectInfo(Object project) 13

3.2.16 getConfigInfo(Object project, String configName) 14

3.2.17 setConfigInfo(Object project, String configName, Map<String, Object> info, boolean force) .14

3.2.18 setBuildTimeout(long timeout) 14

3.2.19 deleteProject(Object project, boolean deleteContent, boolean force) 15

3.3 AppBuilder-related Commands 15

3.3.1 generateProject(Object project, Map<String, Object> options) 15

3.3.2 generateIsc(Object iscFiles, Map<String, Object> options) 16

3.3.3 generateIscFile(Object iscFile, Map<String, Object> options) 16

3.3.4 getCompatibleToolchains(Object iscFile) 16

3.3.5 isAfv6IscFile(Object iscFile) 17

3.3.6 validateSDK(Object sdk) 17

1. Requirements

- Simplicity Studio v4 Installed
- Either:
  - Cygwin, Git Bash, or another Windows-based GNU tool program or
  - Windows command prompt
- Python 2.7.x installed (version 2.7.x is required)
- Obtain example Python scripts

  If this application note was obtained from within Simplicity Studio, from the Application Note dialog box, highlight AN1121 and choose the "Open Folder" option and the example Python scripts will be in a subfolder called "PythonScripts".

  If the application note was downloaded from www.silabs.com, then a zip file of the Python scripts should be available for download from AN1121SW.zip.

2. Using the Script

2.1 Launch a Command Window

Either start up the Windows® OS bash program, either cygwin, git bash, or another Windows GNU based tool or else launch a Windows Command Prompt.

2.2 Using the <runScript.sh> or <runScript.bat> Script to Launch Simplicity Studio to Execute a Python Script

The <runScript.sh> and <runScript.bat> are script files that ship with Simplicity Studio by default. They can be used to start Simplicity Studio by passing arguments to the program. The Simplicity Studio default workspace is used (`C:\Users\[USERNAME]\SimplicityStudio\v4_workspace`) unless the -data workspace argument is specified. Other arguments point to the Python scripts and the script's arguments.

```
[pathTo_runScript] –data [pathToWorkspace] [pathToScript.py] [required script arguments] {optional arguments}
```

For example:

```
C:\SiliconLabs\SimplicityStudio\v4\developer\scripting\runScript.bat –data /c/Users/userA/SimplicityStudio/v4_workspace/MyProjectName ./BuildExistingProject_v2.py
```

Detailed examples of running each Python script are given after the description of the script.

2.3 Available Example Python Scripts

2.3.1 <BuildExistingProject_v2.py>

Use case: Headless build of an existing project in an existing workspace.

Required files: .project, .cproject and project-specific source files.

The script takes as an argument the name of an existing project folder in the workspace for the build. An optional argument specifies which build configuration to use - for example "GNU ARM v4.9.3 - Release". The project must already exist in the workspace.

Python script-required arguments

- The name of the project folder.

Python script-optional arguments

- -cfg is the build configuration for the build – if not given then the current active configuration of the project is used.
- --AutoEnable – If this option is specified then Simplicity Studio finds compatible values for the SDK and toolchain settings. The default is False.

Examples of using this script:

Using <runScript.sh>

```
/SiliconLabs/SimplicityStudio/v4/developer/scripting/runScript.sh -data /c/temp/test_v4_workspace /C/SWTools/scripts/BuildExistingProject_v2.py SLSTK3401A_emode_2
```

```
/SiliconLabs/SimplicityStudio/v4/developer/scripting/runScript.sh -data /c/temp/test_v4_workspace /C/SWTools/scripts/BuildExistingProject_v2.py soc-smartphone
```

```
/SiliconLabs/SimplicityStudio/v4/developer/scripting/runScript.sh -data /c/temp/test_v4_workspace /C/SWTools/scripts/BuildExistingProject_v2.py SLSTK3401A_emode_2 -cfg "GNU ARM v4.9.3 - Release"
```

```
/SiliconLabs/SimplicityStudio/v4/developer/scripting/runScript.sh -data /c/temp/test_v4_workspace /C/SWTools/scripts/BuildExistingProject_v2.py SLSTK3401A_emode_2 -cfg "GNU ARM v4.9.3 - Release" –AutoEnable
```

Using <runScript.bat>

```
\SiliconLabs\SimplicityStudio\v4\developer\scripting\runScript.bat -data \temp\test_v4_workspace \SWTools\scripts\BuildExistingProject_v2.py SLSTK3401A_emode_2 -cfg "GNU ARM v4.9.3 - Release"
```

2.3.2 <ImportAndGenerateIsc_v4Pub.py>

Use Case: This script is for Silicon Labs AppBuilder based workflows for Wireless products. AppBuilder generates the project settings and required SDK source file links from the project .isc file.

Required files: .isc file, customer specific source files.

This script takes as arguments the .isc file used for building Mesh, Flex or BLE projects and the path to the stack that generated the project and optionally the toolchain. It will import the .isc file into the specified workspace, call AppBuilder to generate the project and then it will build the project.

Script-required arguments

- The full path to the .isc file.

Script-optional arguments

- -tc is the toolchain type

Examples of using this script:

```
/SiliconLabs/SimplicityStudio/v4/developer/scripting/runScript.sh -data /c/temp/test_v4_workspace/ /c/SwTools/s
cripts/ImportAndGenerateIsc_v4Pub.py /c/temp/project_input_isc/Z3LightSoc/Z3LightSoc.isc /c/SiliconLabs/Simplic
ityStudio/v4/developer/stacks/znet/v5.8.1.0 -tc=iar

/SiliconLabs/SimplicityStudio/v4/developer/scripting/runScript.sh -data /c/temp/test_v4_workspace/ /c/SwTools/s
cripts/ImportAndGenerateIsc_v4Pub.py /c/temp/project_input_isc/Z3DimmableLightSoc_2/Z3DimmableLightSoc_2.isc /c
/SiliconLabs/SimplicityStudio/v4/developer/sdks/gecko_sdk_suite/v1.1 -tc=iar
```

2.3.3 <ImportExistingProjectAndBuild.py>

Use case: Build an existing project in a new workspace. This can be used for continuous integration builds where the workspace is based on a build number or some other identifier associated with the build.

Required files: .project, .cproject and project specific source files.

This script takes as arguments the path to the project folder (must contain .project and a .cproject files) to import. It will import the project into the specified workspace, and then it will build the project.

Script-required arguments

- The full path to the project folder.

Script-optional arguments

- -cfg is the build configuration for the build – if not given then the current active configuration of the project is used.
- --AutoEnable – If this option is specified then Simplicity Studio finds compatible values for the SDK and toolchain settings. The default is False.

Examples of using this script:

```
/SiliconLabs/SimplicityStudio/v4/developer/scripting/runScript.sh -data /c/temp/test_v4_workspace/ /C/SWTools/s
cripts/ImportExistingProjectAndBuild.py /c/temp/project_input_src/SLSTK3401A_emode_2
```

```
/SiliconLabs/SimplicityStudio/v4/developer/scripting/runScript.sh -data /c/temp/test_v4_workspace/ /C/SWTools/s
cripts/ImportExistingProjectAndBuild.py /c/temp/project_input_src/soc-smartphone -cfg "GNU ARM v4.9.3 - Default
"
```

2.3.4 <ImportExistingZippedProjectAndBuild.py>

Use case: Projects that are shared between engineers or build machines in a zip file as opposed to being pulled from a version control system.

Required files: A zip file containing the .project, .cproject and project specific source files.

This script takes as arguments the path to the zipped project folder (must contain .project and .cproject files) to import. It will unzip the project into the specified workspace, import the project into the workspace and then it will build the project.

Script-required arguments

- The path to the zipped project folder.

Script-optional arguments

- -cfg is the build configuration for the build – if not given then the current active configuration of the project is used.
- --AutoEnable – If this option is specified then Simplicity Studio finds compatible values for the SDK and toolchain settings. The default is False if the option isn't specified.

Examples of using this script:

Using <runScript.sh>

```
/SiliconLabs/SimplicityStudio/v4/developer/scripting/runScript.sh -data /c/temp/test_v4_workspace/ /C/SWTools/scripts/ImportExistingZippedProjectAndBuild.py /c/temp/project_input_zipped/railtest_efr32.zip
```

```
/SiliconLabs/SimplicityStudio/v4/developer/scripting/runScript.sh -data /c/temp/test_v4_workspace/ /C/SWTools/scripts/ImportExistingZippedProjectAndBuild.py /c/temp/project_input_zipped/railtest_efr32.zip -cfg "GNU ARM v4.9.3 - Default"
```

Using <runScript.bat>

```
\SiliconLabs\SimplicityStudio\v4\developer\scripting\runScript.bat -data \temp\test_v4_workspace \SWTools\scripts\ImportExistingZippedProjectAndBuild.py \temp\project_input_zipped\railtest_efr32.zip -cfg "GNU ARM v4.9.3 - Default"
```

2.3.5 <CopyProjectsAndBuild.py>

Use case: Building multiple projects in a new workspace. Projects can be stored in an existing workspace or any file structure that has all projects in subfolders to the specified path, but nested projects are not supported.

Required files: Each project in a subfolder that contains the .project, .cproject and project specific source files.

This script takes as an argument the path to a folder that has multiple project folders under it. It will copy all of the project folders into the specified workspace and then import each project into the workspace and then it will build the projects.

Script-required arguments

- The path to the folder that contains one or more project folders.

Script-optional arguments

- none

Examples of using this script:

Using <runScript.sh>

```
/SiliconLabs/SimplicityStudio/v4/developer/scripting/runScript.sh -data /c/temp/test_v4_workspace/ /C/SWTools/scripts/CopyProjectsAndBuild.py /c/temp/project_input_src
```

Using <runScript.bat>

```
\SiliconLabs\SimplicityStudio\v4\developer\scripting\runScript.bat -data \temp\test_v4_workspace \SWTools\scripts\CopyProjectsAndBuild.py \temp\project_input_src
```

2.3.6 <CopySdkSettingsFromWorkspace.py>

Use case: Headless builds will be performed on a new workspace and SDKs external to Simplicity Studio are used for the build.

Required files: There should be a '.metadata' folder with '.plugins' subfolder. Specific files that contain the SDK settings are then copied into the new workspace.

This script imports the SDK settings from the workspace passed as an argument to the script. This script is mainly useful if there are external SDKs that are used when building projects. External SDKs means SDKs that are installed outside of the Simplicity Studio installation directory. It can also be used if some projects are built in a workspace that disables some of the SDKs installed in the Simplicity Studio installation directory.

Script-required arguments

- The path to the workspace folder that contains the desired SDK settings.

Script-optional arguments

- None

Examples of using this script:

Using <runScript.sh>

```
/SiliconLabs/SimplicityStudio/v4/developer/scripting/runScript.sh -data /c/temp/test_v4_workspace/ /C/SWTools/scripts/CopySdkSettingsFromWorkspace.py /c/temp/project_input_src
```

Using <runScript.bat>

```
\SiliconLabs\SimplicityStudio\v4\developer\scripting\runScript.bat -data \temp\test_v4_workspace \SWTools\scripts\CopySdkSettingsFromWorkspace.py \temp/project_input_src
```

3. API Definition of Available Commands

The following section describes the available arguments and commands that can be used to modify the above Python scripts or to create a custom script.

3.1 Arguments

Parameter	Description

`String OPTION_BUILD_CONFIG = "buildConfiguration";`	This variable is used to change the project's active build configuration. Example configuration strings:
	`"GNU ARM v4.9.3 -Default"`
	`“IAR ARM – Release”`
	`“Keil 8051 v9.53 - Debug”`

Parameter	Description
`String OPTION_BUILD_TYPE = "buildType";`	This variable indicates the build type.
Possible values:	
`String BUILD_TYPE_FULL = "buildType_FULL";`	Full Build (clean and then build).
`String BUILD_TYPE_INCREMENTAL = "buildType_INCREMENTAL";`	Incremental Build
`String BUILD_TYPE_CLEAN = "buildType_CLEAN";`	Clean

Parameter	Description
`String INFO_BOARD_IDS = "boardIDs";`	Current board ID See `getProjectInfo()` See `getConfigInfo()`

Parameter	Description
`String INFO_PART_ID = "partID";`	Current part ID See `getProjectInfo()` See `getConfigInfo()`

Parameter	Description
`String INFO_SDK_ID = "sdkID";`	Current SDK ID See `getProjectInfo()` See `getConfigInfo()`

Parameter	Description
`String INFO_TOOLCHAIN_ID = "toolchainID";`	Current toolchain ID See `getProjectInfo()` See `getConfigInfo()`

Parameter	Description
`String INFO_CONFIGS = "configs";`	A list of labels of the configurations defined for a project. See `getProjectInfo()`

Parameter	Description
`String INFO_CURRENT_CONFIG = "currentConfig";`	The label of the current configuration for the project. See `getProjectInfo()`

Parameter	Description
`String OPTION_NO_AUTO_ENABLE="noAutoEnable";`	If set to True (disabling auto enable), the function won't try and find the board / part / SDK or toolchain that is referenced by the project. The values for each must be set in the current project / workspace settings. The default is False, but the default is not recommended unless that setting is failing. Note that most of the Python scripts use an AutoEnable option (so the opposite of what is passed to the Simplicity Studio function) and it defaults to False. It is easier to have it be an optional Python argument with a default value.

3.2 Project-related Commands

3.2.1 getWorkspace()

Description: Gets the current Eclipse workspace.

Prototype: `getWorkspace()`

Return Value: Returns an IWorkspace instance.

3.2.2 getWorkspaceLocation()

Description: Gets the filesystem path of the current Eclipse workspace.

Prototype: `getWorkspaceLocation()`

Return Value: Returns the path as a string.

3.2.3 getProjectsInWorkspace()

Description: Gets a list of all the names of the projects in the workspace (including closed or non-Studio projects).

Prototype: `getProjectsInWorkspace()`

Return Value: Returns a string array of project names.

3.2.4 getOpenStudioProjectsInWorkspace()

Description: Gets a list of all the names of the open SLS-compatible projects in the workspace.

Prototype: `getOpenStudioProjectsInWorkspace()`

Return Value: Returns a string array of project names.

3.2.5 getProject(String name)

Description: Gets the Eclipse IProject handle for an existing open project.

Prototype: `getProject(String name)`

Parameters:

1. *name* The name of the project.

Return Value: Returns handle to the specified project.

3.2.6 getProjectForFile(String filePath)

Description: Gets the project that owns this exact file or return null if none could be found.

Prototype: `getProjectForFile(String filePath)`

Parameters:

1. *filePath* The full path to the file.

Return Value: Returns handle to the parent project or null if the file isn't contained by a project.

3.2.7 getBuildTimeout()

Description: Gets the project build timeout (in milliseconds).

Prototype: `getBuildTimeout()`

Return Value: Returns timeout (in milliseconds).

3.2.8 isCompatibleWithPartOrGroup(Object project, Object part)

Description: Determines whether the given project (name or IProject) is compatible with the given part.

Prototype: `isCompatibleWithPartOrGroup(Object project, Object part)`

Parameters:

1. *project* The project name, IProjectHandle, or IProject.
2. *part* A part/group name, IPart[Group]Descriptor, or IPart[Group].

Return Value: Returns boolean true if compatible.

3.2.9 importExistingProject(String path, Map<String, Object> options, IProgressMonitor monitor)

Description: Imports a previously-existing project (with .project file) into the workspace.

Prototype: `importExistingProject(String path, Map<String, Object> options, IProgressMonitor monitor)`

Parameters:

1. *path* The path or URI to the project directory.
2. *options* A map of key to value options, or null; allowed keys:
   - OPTION_PROJECT_NAME
   - OPTION_NO_AUTO_ENABLE
3. *monitor* An optional progress monitor for progress and cancellation.

Return Value: Returns a handle to the imported project.

Error Handling: Throws Java CoreException if anything goes wrong.

3.2.10 importSlsProject(String pathToProjectFile, Map<String, Object> options, IProgressMonitor monitor)

Description: Imports a project from a .slsproj file.

Prototype: `importSlsProject(String pathToProjectFile, Map<String, Object> options, IProgressMonitor monitor)`

Parameters:

1. *pathToProjectFile* The full path to the .slsproj file.
2. *options* A map of key to value options, or null. Allowed keys are:
   - OPTION_PROJECT_NAME
   - OPTION_PROJECT_LOCATION
   - OPTION_BOARD_ID
   - OPTION_PART_ID
   - OPTION_SDK_ID
   - OPTION_TOOLCHAIN_ID
   - OPTION_CONTENT_ROOT
   - OPTION_NO_AUTO_ENABLE
3. *monitor* An optional progress monitor for progress and cancellation.

Return Value: A return handle to the opened project.

Error Handling: Throws Java CoreException if anything goes wrong.

3.2.11 exportSlsProjFile(Object project, Map<String, Object> options, String pathToProjectFile)

Description: Exports a project to a .slsproj file.

Prototype: `exportSlsProjFile(Object project, Map<String, Object> options, String pathToProjectFile)`

Parameters:

1. *project* The project name, IProjectHandle or IProject.
2. *options* A map of key to value options, or null; allowed keys:
   - OPTION_PROJECT_NAME
   - OPTION_LABEL string; to provide the label for the exported project
   - OPTION_DESCRIPTION string; to provide the description for the exported project
   - OPTION_IS_TEMPLATE boolean; set to True to configure the project for use in templates:

     Omit build configurations, promoting the build settings from OPTION_REFERENCE_CONFIGURATION to project level
   - OPTION_REFERENCE_CONFIGURATION string; the name of the configuration to use for driving OPTION_IS_TEMPLATE the default is the current/active configuration
   - OPTION_BOARD_COMPATIBILITY string; to override the purported board compatibility for the project
   - OPTION_PART_COMPATIBILITY string; to override the purported part compatibility for the project
   - OPTION_SDK_COMPATIBILITY string; to override the purported SDK compatibility for the project
   - OPTION_TOOLCHAIN_COMPATIBILITY string; to override the purported toolchain compatibility for the project
   - OPTION_CONTENT_ROOT string; to override the URI used in the content root
3. *pathToProjectFile* The path and filename of the .slsproj file to create.

Error Handling:

1. Throws Java IOException if the file cannot be written.
2. Throws Java CoreException if anything else goes wrong.

3.2.12 switchCurrentConfig(Object project, Object configuration, IProgressMonitor monitor)

Description: Switches the current configuration to the input configuration.

Prototype: `switchCurrentConfig(Object project, Object configuration, IProgressMonitor monitor)`

Parameters: param

1. *project* The project name, IProjectHandle, or IProject.
2. *configuration* The configuration, either its name or ID.
3. *monitor* An optional progress monitor for progress and cancellation.

Error Handling: Throws Java CoreException if anything goes wrong.

3.2.13 buildProject(Object project, Map<String, Object> options, IProgressMonitor monitor)

Description: Builds/Cleans the project based on the options map using OPTION_BUILD_TYPE.

Prototype: `buildProject(Object project, Map<String, Object> options, IProgressMonitor monitor)`

Parameters: Specify the configuration to build with OPTION_BUILD_CONFIG (this will change the active configuration).

1. *project* The project name, IProjectHandle, or IProject.
2. *options* A map of key to value options, or null allowed keys:
   - OPTION_BUILD_TYPE
   - OPTION_BUILD_CONFIG
   - OPTION_BUILDER_NAME
   - OPTION_NO_AUTO_ENABLE
3. *monitor* An optional progress monitor for progress and cancellation

Return Value: Returns buildResults with IMarkers and file changes.

Error Handling: Throws Java CoreException if anything goes wrong.

3.2.14 cleanProject(Object project, IProgressMonitor monitor)

Description: This is a convenience method and will call buildProject with BUILD_TYPE_CLEAN and build the active configuration.

Prototype: `cleanProject(Object project, IProgressMonitor monitor)`

Parameters:

1. *project* The project.
2. *monitor* An optional progress monitor for progress and cancellation.

Return Value: Returns buildResults with IMarkers and file changes.

Error Handling: Throws Java CoreException if anything goes wrong.

3.2.15 getProjectInfo(Object project)

Description: Gets details about the project.

Prototype: `getProjectInfo(Object project)`

Parameters:

1. *project* The project name, IProjectHandle, or IProject

Return Value: Returns map of keys to values for the following:

- INFO_PROJECT_LOCATION
- INFO_CONFIGS
- INFO_CURRENT_CONFIG

Error Handling: Throws Java CoreException for nonexistent, closed, or non-Studio project.

3.2.16 getConfigInfo(Object project, String configName)

Description: Gets details about the given configuration of the given project.

Prototype: `getConfigInfo(Object project, String configName)`

Parameters:

1. *project* The project name, IProjectHandle, or IProject.
2. *configName* The configuration name.

Return Value: Returns map of keys to values for the following:

1. INFO_BOARD_ID
2. INFO_PART_ID
3. INFO_SDK_ID
4. INFO_TOOLCHAIN_ID

Error Handling: Throws Java CoreException for nonexistent, closed, or non-Studio project.

3.2.17 setConfigInfo(Object project, String configName, Map<String, Object> info, boolean force)

Description: Sets details about the configuration. Any items in the map will be modified.

Prototype: `setConfigInfo(Object project, String configName, Map<String, Object> info, boolean force)`

Parameters:

1. *project* The project name, IProjectHandle, or IProject.
2. *configName* The configuration name.
3. *info* The map of INFO_... keys to values.
   - INFO_BOARD_ID
   - INFO_PART_ID
   - INFO_SDK_ID
   - INFO_TOOLCHAIN_ID
4. *force* If true, always apply changes even if nothing changed.

Error Handling: Throws Java CoreException for nonexistent, closed, or non-Studio project, or unable to change settings.

3.2.18 setBuildTimeout(long timeout)

Description: Set the project build timeout (in milliseconds).

Prototype: `setBuildTimeout(long timeout)`

Parameters:

1. *timeout* Timeout in milliseconds

Error Handling: Throws Java CoreException if anything goes wrong.

3.2.19 deleteProject(Object project, boolean deleteContent, boolean force)

Description: A convenience method to delete a project. It wraps the deletion in a workspace job so that it won't fail because a resource is in use.

Prototype: `deleteProject(Object project, boolean deleteContent, boolean force)`

Parameters:

1. *project* The project name, IProjectHandle, or IProject.
2. *deleteContent* Delete the disk content or just the project handle.
3. *force* A flag controlling whether resources that are not in sync with the local file system are tolerated.

Error Handling: Throws Java CoreException if:

- This resource could not be deleted.
- This resource or one of its descendants is out of sync with the local file system and force is false.

Remarks: Resource changes are disallowed during certain types of resource change event notifications.

- See IResourceChangeEvent for more details.

3.3 AppBuilder-related Commands

3.3.1 generateProject(Object project, Map<String, Object> options)

Description: Generates into this project using the first seen isc file or the isc file specified in the options map. This will work for both Afv2 and Afv6 depending on the isc file that is generated.

Prototype: `(Object project, Map<String, Object> options)`

Parameters:

1. *project* The project name, IProjectHandle or IProject.
2. *options* A map of key to value options, or null; allowed keys:
   - Afv2 & Afv6 Keys:
     - OPTION_ISC_FILE_NAME: string; to specify which isc file to use - the first one found will be chosen
     - OPTION_ISC_FILE_PATH: string; to specify the path to the isc file that should be used - can be project relative or absolute
     - OPTION_GENERATE_PROTECTED_FILES: boolean; to specify whether any protected should overwritten
     - OPTION_VERBOSE: boolean; specify if the generation process should be verbose
   - Afv6 Only Keys:
     - OPTION_GENERATE_ARCHITECTURES: list of strings, Space/Comma separated list of String (architecture IDs)
       - Specify the list of generated architectures. If null, the default architecture will be used
     - OPTION_ALLOWED_ARCHITECTURES: list of strings, Space/Comma separated list of String (architecture IDs)
       - Specify the list of allowed architectures. If null, any framework valid architectures can be used
     - OPTION_ALLOW_INTERNAL_STACK: boolean; specify whether an internal stack can be used to generate

Error Handling: Throws Java CoreException if there is no isc file in this project or something goes wrong.

3.3.2 generateIsc(Object iscFiles, Map<String, Object> options)

Description: Runs the generation process on the input isc file. This will correctly operate on Afv2 or Afv6 depending on the input isc file.

Prototype: `generateIsc(Object iscFiles, Map<String, Object> options)`

Parameters:

1. *iscFile* A string file path to the generated isc file
2. *options* A map of key to value options, or null; allowed keys:
   - Afv2 & Afv6 Keys:
     - OPTION_VERBOSE: boolean; specify if the generation process should be verbose
   - Afv6 Only Keys:
     - OPTION_GENERATE_ARCHITECTURES: list of strings, Space/Comma separated list of String (architecture IDs)
       - Specify the list of generated architectures. If null, the default architecture will be used
     - OPTION_ALLOWED_ARCHITECTURES: list of strings, Space/Comma separated list of String (architecture IDs)
       - Specify the list of allowed architectures. If null, any framework valid architectures can be used
     - OPTION_ALLOW_INTERNAL_STACK: boolean; specify whether an internal stack can be used to generate

Error Handling: Throws Java CoreException if something goes wrong.

3.3.3 generateIscFile(Object iscFile, Map<String, Object> options)

Description: Runs the generation process on the input isc file. This will correctly operate on Afv2 or Afv6 depending on the input isc file. This is a utility method, fully equivalent to `generateIsc(Object, Map)`.

Prototype: `generateIscFile(Object iscFile, Map<String, Object> options)`

Parameters:

1. *iscFile* A string file path to the generated isc file.
2. *options* A map of key to value options, or null; allowed keys:
   - Afv2 & Afv6
     - OPTION_GENERATE_PROTECTED_FILES: boolean; to specify whether any protected files should overwritten
     - OPTION_VERBOSE: boolean; specify if the generation process should be verbose
   - Afv6 Only
     - OPTION_GENERATE_ARCHITECTURES: list of strings, Space/Comma separated list of String (architecture IDs)
       - Specify the list of generated architectures. If null, the default architecture will be used
     - OPTION_ALLOWED_ARCHITECTURES: list of strings, Space/Comma separated list of String (architecture IDs)
       - Specify the list of allowed architectures. If null, any framework valid architectures can be used
     - OPTION_ALLOW_INTERNAL_STACK: boolean; specify whether an internal stack can be used to generate

Error Handling: Throws Java CoreException if something goes wrong.

3.3.4 getCompatibleToolchains(Object iscFile)

Description: Returns the toolchain IDs that are compatible with the framework that the input isc file will use.

Prototype: `getCompatibleToolchains(Object iscFile)`

Parameters:

1. *iscFile* A string or file path to a *.isc file.

Return Value: Returns a list of compatible toolchain IDs.

Error Handling: Throws CoreException if there is no isc file in this project or something goes wrong.

3.3.5 isAfv6IscFile(Object iscFile)

Description: Returns whether the input file or string filepath is AFV6 or AFV2.

Prototype: `isAfv6IscFile(Object iscFile)`

Parameters:

1. *iscFile* A string or file path to a *.isc file.

Return Value: Returns true if the file is AFV6, false otherwise.

Error Handling: Throws Java CoreException if something goes wrong.

3.3.6 validateSDK(Object sdk)

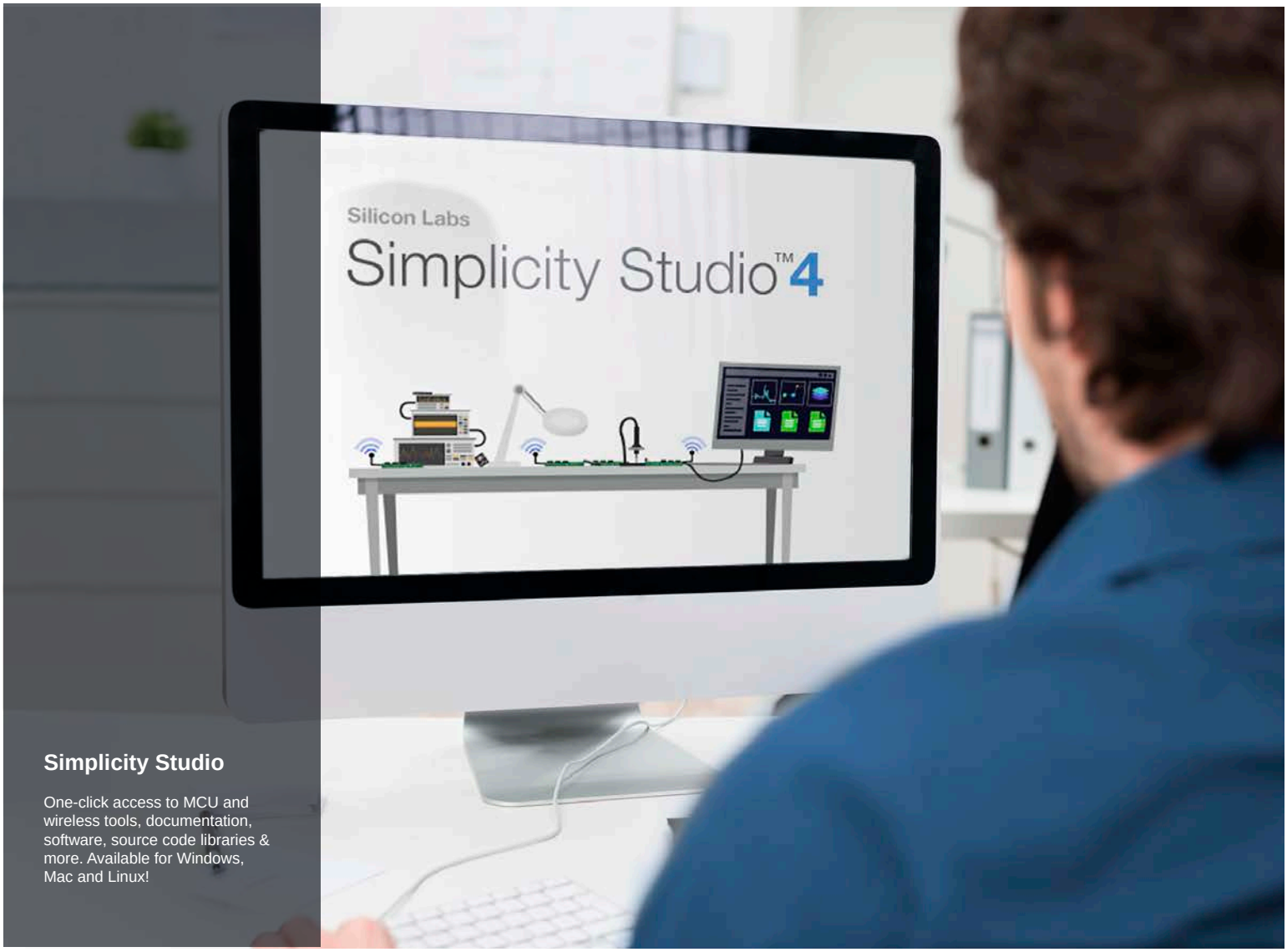
Description: Validates that all frameworks within this SDK load correctly.

Prototype: `validateSDK(Object sdk)`

Parameters:

1. *SDK* The SDK to validate

Error Handling: Throws Java CoreException if there is a loading error



Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

Disclaimer
Silicon Labs intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Labs products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Labs reserves the right to make changes without further notice and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Silicon Labs shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any Life Support System without the specific written consent of Silicon Labs. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Labs products are not designed or authorized for military applications. Silicon Labs products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons.

Trademark Information
Silicon Laboratories Inc.® , Silicon Laboratories®, Silicon Labs®, SiLabs® and the Silicon Labs logo®, Bluegiga®, Bluegiga Logo®, Clockbuilder®, CMEMS®, DSPLL®, EFM®, EFM32®, EFR, Ember®, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Ember®, EZLink®, EZRadio®, EZRadioPRO®, Gecko®, ISOmodem®, Micrium, Precision32®, ProSLIC®, Simplicity Studio®, SiPHY®, Telegesis, the Telegesis Logo®, USBXpress®, Zentri and others are trademarks or registered trademarks of Silicon Labs. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. All other products or brand names mentioned herein are trademarks of their respective holders.

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com