Connector

This is the central tool which allows the user to define the interfaces to interact with the other system.

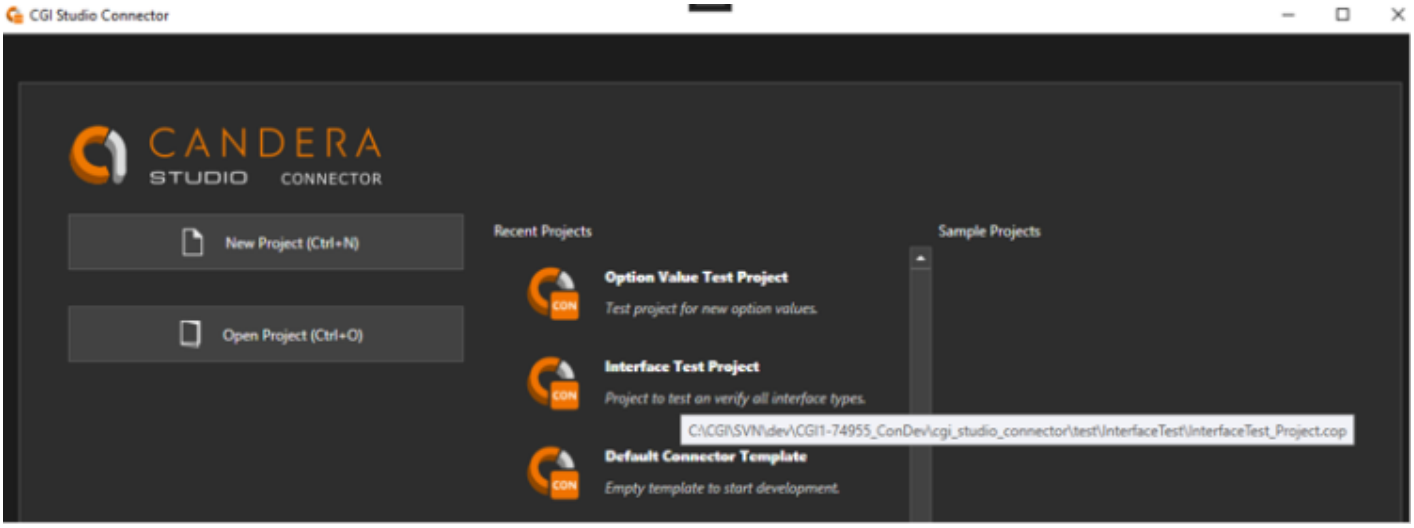
- Welcome Screen
- New Project
- Modules
- Interfaces
- Edit Menu
- Build Menu
- Project Settings
- Builds

Welcome Screen

The welcome screen allows the user to create a new (Ctrl + N) or open an existing (Ctrl + O) CGI Connector Project.

Additionally, it also shows a list of the recently used projects to provide quick access. The exact location of the project can be seen in the tooltip of the individual projects.

It will also display a list of sample projects that can be used as references.



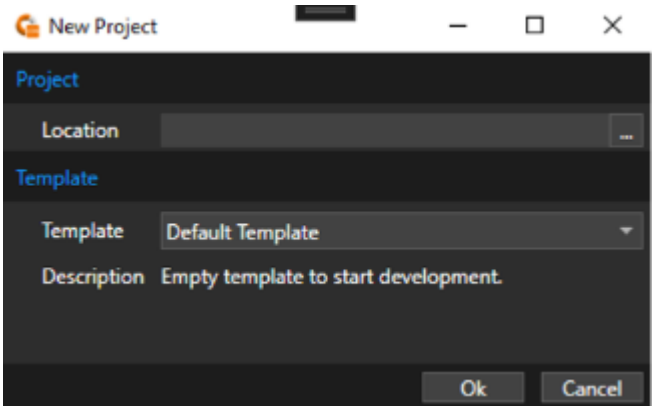
New Project

A new project can be created from the Welcome Screen or via the menu bar (File > New Project) or via the shortcut “Ctrl + N”. This will open additional windows to help setting up the new project.

Template-Selection

In the Template-Selection screen the user needs to first select the location where the new project shall be stored and select which template shall be used as a starting point for the project.

A template usually contains premade application code, build scripts, tooling related files and predefined project settings.



Setup

On the next screen the user can configure the most important project settings.

General

Property Name	Description
Name	Name of the project.
Description	Description of the project.

Tools

Property Name	Description
Generic	Enables generic support. This will provide a plain C/C++ interface.
MATLAB	Enables support for MATLAB. This will provide additional options/setting related to MathWorks MATLAB support

CGI Studio

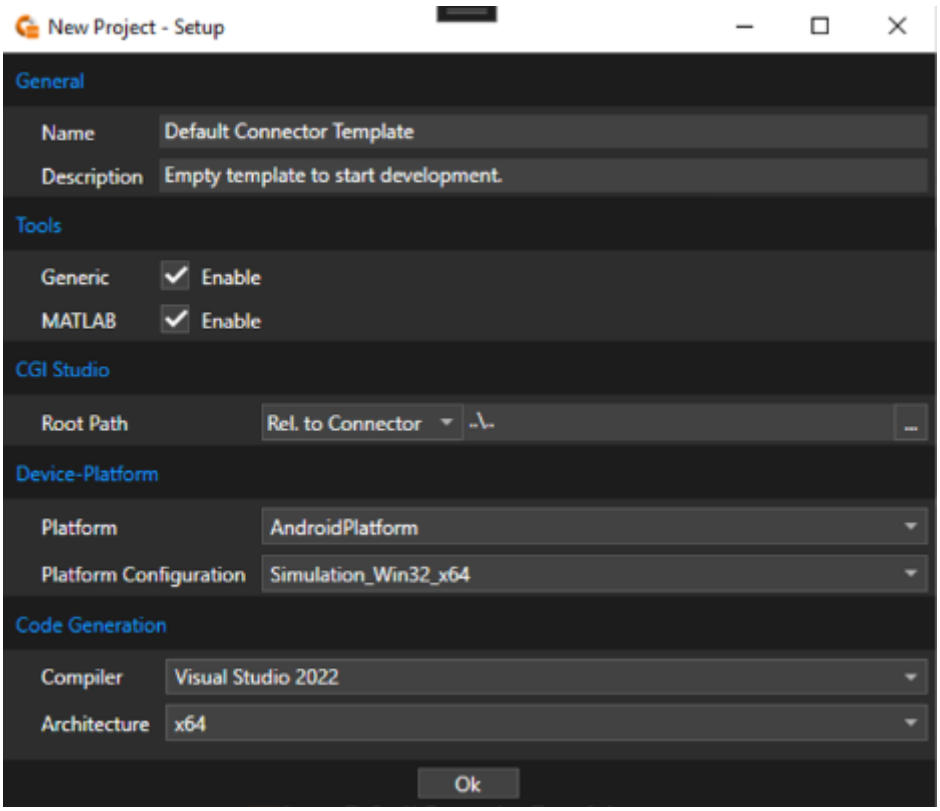
Property Name	Description
Root Path	Path to CGI Studio package. The path can be either absolute or relative to CGI Connector/Project paths.

Device-Platform

Property Name	Description
Platform	Name of the CGI Studio Platform (as found in CMake)
Platform Configuration	Name of the CGI Studio Platform configuration (as found in CMake)

Code Generation

Property Name	Description
Compiler	Compiler used to build code.
Architecture	Architecture used to build the code.



> These, and more, settings can later be changed. For more information, please refer to the "Project Settings" chapter.

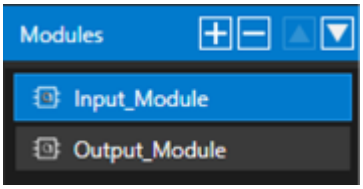
Custom Templates

It is possible to add custom templates to the CGI Connector. This can be useful to apply customizations from a project to other projects.
To create a template, pack all needed content into a ZIP file. Make sure that it contains the CGI Connector-Project file (.cop) in the root. Then add an entry to the “Templates.xml” in “cgi_studio_connector/bin” where the name, description and file location needs to be specified. Note that the file location is relative to the CGI Connector binaries.

Modules

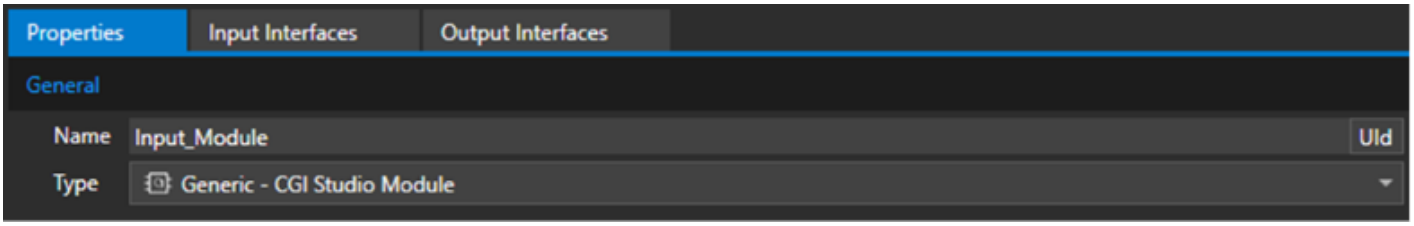
Modules are used to group a set of interfaces which usually belong logically together.

It is possible to add, remove and reorder them using the corresponding buttons at the top of the module list.



Properties

In general, these offer a few basic options. However, depending on the targeted tool, different tool-specific options might be available. For more information about tool specific options and limitation refer to the specific target tool chapter.

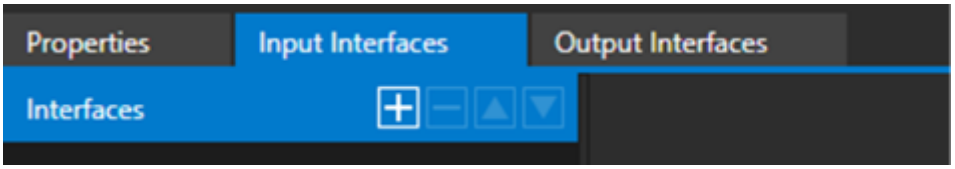


Property Name	Description
Name	Name of the module. Only characters from ‘a’-‘z’, ‘A’-Z’, ‘0’-‘9’, ‘ ‘ , ‘-‘ and ‘_’ are allowed, though the name must start with a character.
Uld	The Uld is used to identify the module. It can be seen as the tooltip of the ‘Uld’ button. When the button is clicked a new Uld will be generated.
Type	Defines the type of module. The available options depend on the specific target tool.

Interfaces

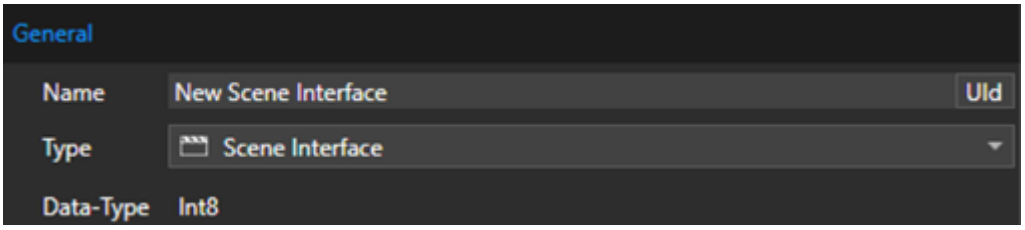
Interfaces define the inputs and outputs of the CGI Application. Depending on the module type either one or both types of interfaces are available.

By selecting the corresponding tab, interfaces can be added, removed or rearranged using the buttons at the top of the interface list.



General Properties

The following general properties are available for all interfaces.



Property Name	Description
Name	Name of the interface. Only characters from ‘a’-‘z’, ‘A’-Z’, ‘0’-‘9’, ‘ ‘, ‘-’ and ‘_’ are allowed, though the name must start with a character.
Uld	The Uld is used to identify the interface. It can be seen as the tooltip of the ‘Uld’ button. When the button is clicked a new Uld will be generated.
Type	Defines the type of interface. The available options depend on the specific target tool.
Data-Type	The used basic data type of the interface to transfer data. This value is automatically calculated depending on the used interface type and set properties. For arrays the length is indicated in brackets.

Types

The following general interface types are available.

Input:

Name
Scene
Scene-Table
Data-Input
Function Call (Action)
Message (Action)

Output:

Name
Data-Output
Generic Event
Function Call (Event)
Message (Event)

> Depending on the used module type, some interface types might not be available or special interface types might be supported. Please refer to the target tool specific section for more details.

Scene Interface

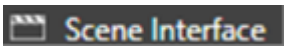
The Scene interface allows to control the state of a scene or group of scenes. Depending on the set value the scene/scenes will change to a different state. By default, the values have the following meaning:

Property Name	Description
0	Destroyed (No data loaded)
1	Loaded (Scene data is loaded to system memory)
2	Uploaded (Scene data is uploaded to VRAM)
3	Active (Scene and related behaviors are updated)

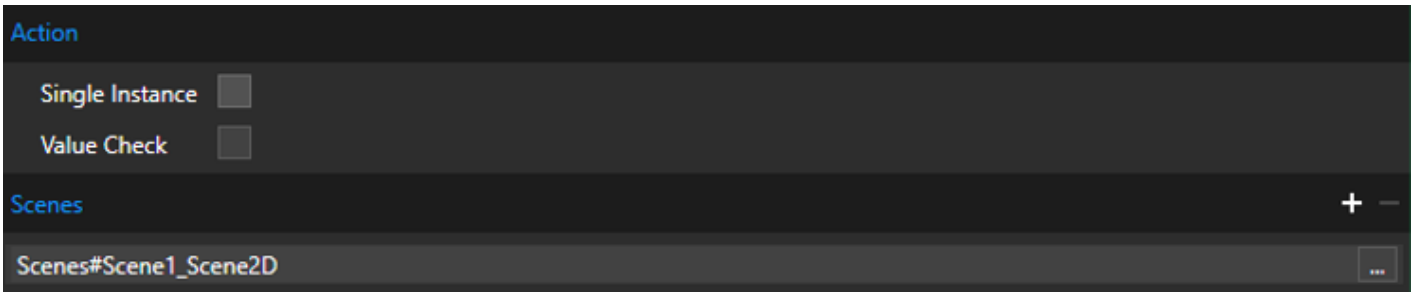
4	Rendering (Scene is drawing)

> The actual meaning of each value is controlled by the application specific implementation.

Icon:



Properties:



Property Name	Description
Single Instance	When selected, only a single instance of this action will be queue. If the action is triggered again before the previous instance is executed it will overwrite the previous action in the queue instead.
Value Check	If enabled, the sent value will be compared to the previous value and processing will be skipped if it has not changed. This can improve performance if the sent value is often the same/different.
Scenes	Scene controlled by the interface. A scene is defined by the following expression: [FOLDERNAME#….#]SCENENAME Clicking on the '…' button next to the field opens a dialog window which lists possible scenes from the referenced Scene Composer solution. Multiple scenes can be specified by adding or removing additional lines with the '+' and '-' buttons.

Scene Table Interface

The Scene-Table signal allows to specify a collection of scenes and assigns them a certain Id. The interface value controls this Id and therefore selects the next entry to be activated. If the new entry contains a scene which was not 'active' before, it will be 'activated'. Similar if the new entry is missing a scene which was 'active' before it will be 'deactivated'. If a scene was part of the previous set of scenes and is still part of the new set, no action will be performed.

Additionally, "Hints" can be specified which can affect how the change between different sets happens. These are primarily used to control the transition animation.

>

> The actual behavior is controlled by the application specific implementation.

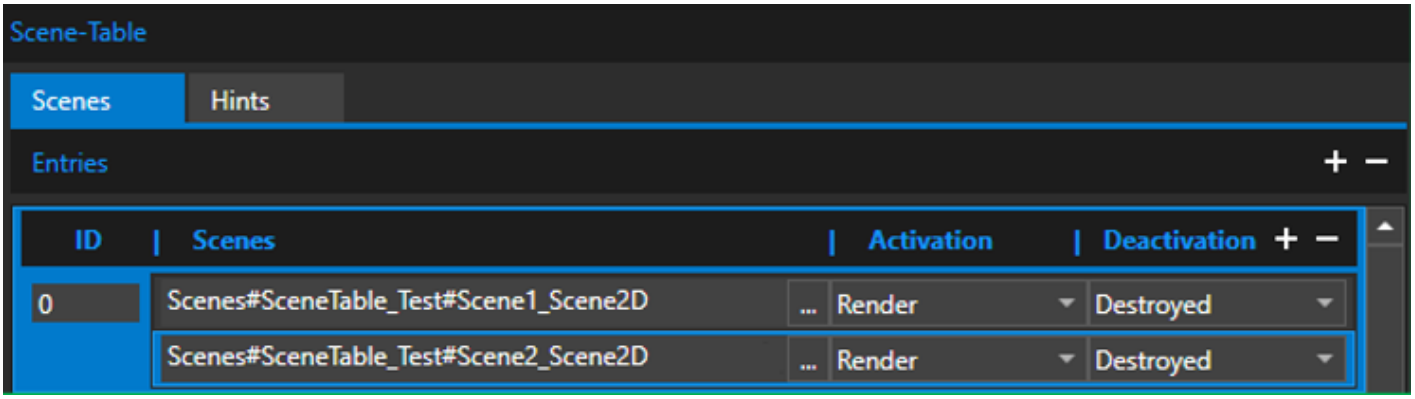
Icon:



Properties:

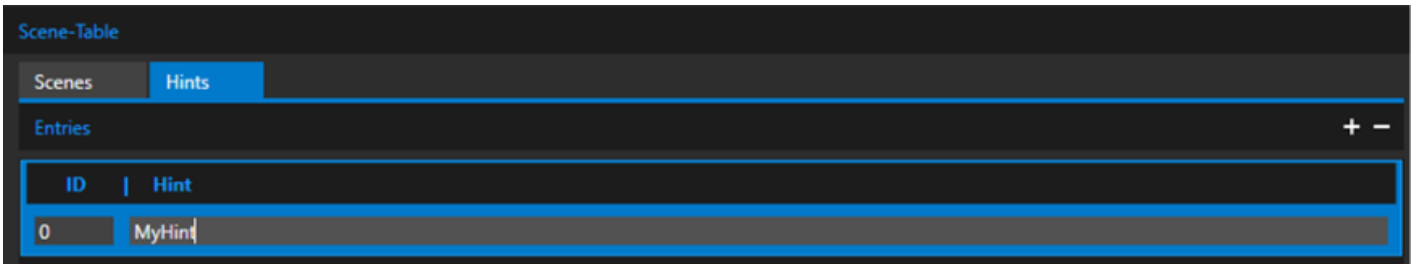


Property Name	Description
Single Instance	When selected, only a single instance of this action will be queue. If the action is triggered again before the previous instance is executed it will overwrite the previous action in the queue instead.
Value Check	If enabled, the sent value will be compared to the previous value and processing will be skipped if it has not changed. This can improve performance if the sent value is often the same/different.



Property Name	Description
(Scenes) Entries	Collection of Scenes represented by an Id. Multiple collections can be specified by adding or removing additional lines with the '+' and '-' buttons.
(Scenes) Entries ID	The Id representing this set of Scenes.
(Scenes) Entries Scenes	Scenes which belong to this entry. A scene is defined by the following expression: [FOLDERNAME#….#]SCENENAME Clicking on the '…' button next to the field opens a dialog window which lists possible scenes from the referenced Scene Composer solution. Multiple scenes can be specified by adding or removing additional lines with the '+' and '-' buttons.

(Scenes) Entries Activation	State of the scene to be applied when scene is activated.
(Scenes) Entries Deactivation	State of the scene to be applied when scene is deactivated.

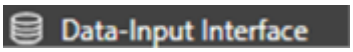


Property Name	Description
(Hints) Entries	Collection of Hints represented by an Id. Multiple collections can be specified by adding or removing additional lines with the '+' and '-' buttons.
(Hints) Entries ID	The Id representing this Hint.
(Hints) Entries Hint	Hint which belongs to this entry.

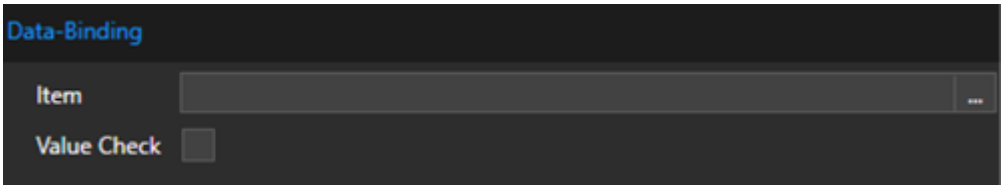
Data-Input Interface

The Data-Input interface is used to send data values from external to the CGI Studio application. Only data-items provided by Courier Data-Binding can be set as destination. For more information about Data-Binding please refer to the Courier documentation.

Icon:



Properties:



Property Name	Description
Item	Data-Item used as a destination for the data value transferred via this signal. Clicking on the '...' button next to the field opens a dialog window which lists all possible data items defined for this application.
Value Check	If enabled, the sent value will be compared to the previous value and processing will be skipped if it has not changed. This can improve performance if the sent value is often the same/different.

Function Call (Action) Interface

The Function-Call (Action) interface allows to trigger a user defined function. To learn how to define functions to be used in the Connector refer to the Native Functions chapter.

Icon:



Properties:

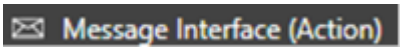


Property Name	Description
Single Instance	When selected, only a single instance of this action will be queue. If the action is triggered again before the previous instance is executed it will overwrite the previous action in the queue instead.
Value Check	If enabled, the sent value will be compared to the previous value and processing will be skipped if it has not changed. This can improve performance if the sent value is often the same/different.
Name	Name of the function to execute. Clicking on the '…' button next to the field opens a dialog window which lists all possible functions defined for this application.
Input Argument	When enabled, a single argument of the function can be defined as Input-Argument. The value of the argument will then depend directly on the data sent when triggering the interface.
Arguments	List of arguments for this function (excluding the Input-Argument if specified). Here the values for the arguments that will be used to execute the function are set. Depending on the type of the argument, the […] button will be displayed next to the field. When clicking it, it will open a dialog window which lists possible objects from the referenced Scene Composer solution that can be used as a value.

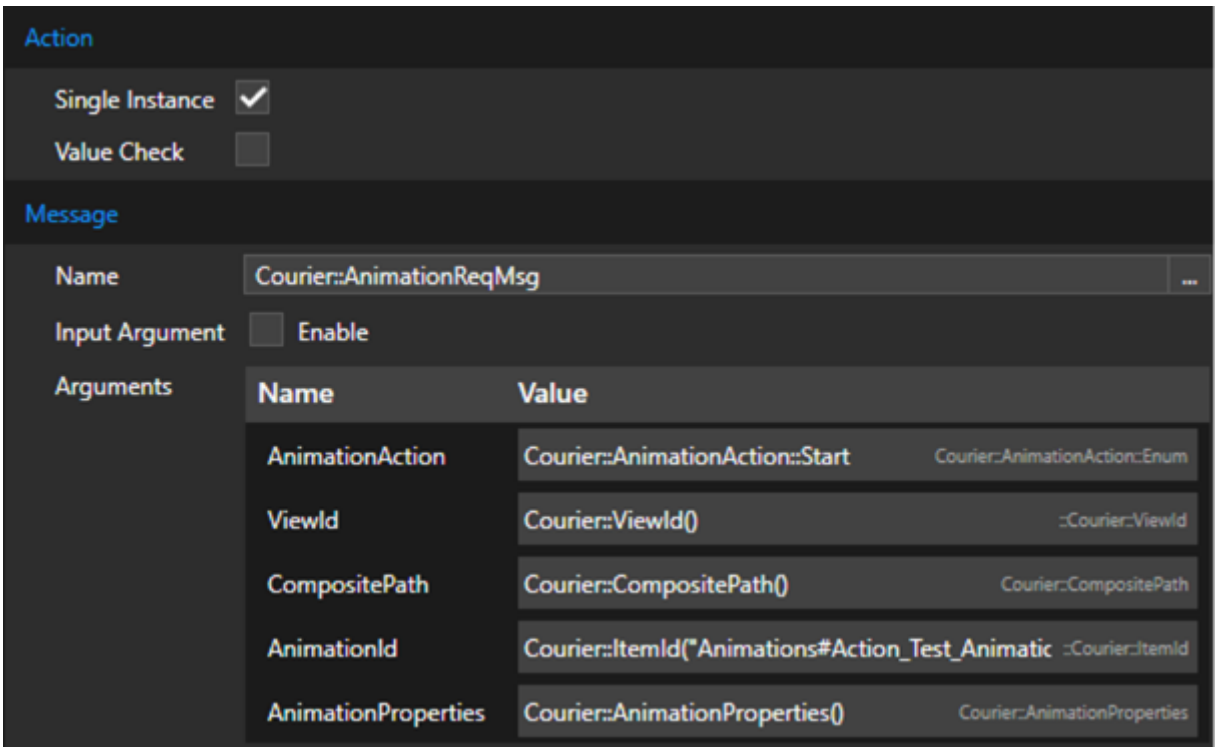
Message (Action) Interface

The Message (Action) interface allows to trigger the sending of a Courier message. For more information about Messaging in Courier please refer to the Courier documentation.

Icon:



Properties:



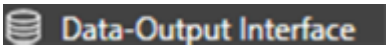
Property Name	Description
Single Instance	When selected, only a single instance of this action will be queue. If the action is triggered again before the previous instance is executed it will overwrite the previous action in the queue instead.
Value Check	If enabled, the sent value will be compared to the previous value and processing will be skipped if it has not changed. This can improve performance if the sent value is often the same/different.
Name	Name of the message to send. Clicking on the '...' button next to the field opens a dialog window which lists all possible messages defined for this application.
Input Argument	When enabled, a single argument of the message can be defined as Input-Argument. The value of the argument will then depend directly on the data sent when triggering the interface.

Arguments	List of arguments for this message (excluding the Input-Argument if specified). Here the values for the arguments that will be used to send the message are set. Depending on the type of the argument, the [...] button will be displayed next to the field. When clicking it, it will open a dialog window which lists possible objects from the referenced Scene Composer solution that can be used as a value.

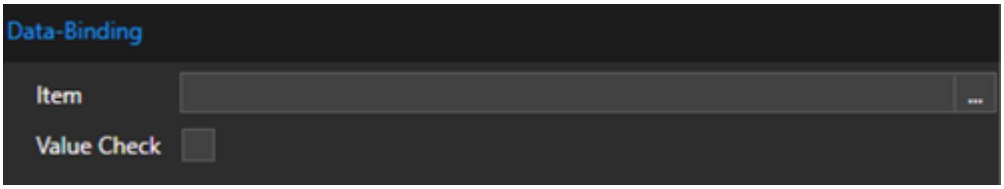
Data-Output Interface

The Data-Output interface is used to receive data values from CGI Studio application. Only data-items provided by Courier Data-Binding can be set as source. For more information about Data-Binding please refer to the Courier documentation.

Icon:



Properties:

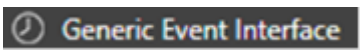


Property Name	Description
Item	Data-Item used as a destination for the data value transferred via this signal. Clicking on the '...' button next to the field opens a dialog window which lists all possible data items defined for this application.
Value Check	If enabled, the sent value will be compared to the previous value and processing will be skipped if it has not changed. This can improve performance if the sent value is often the same/different.

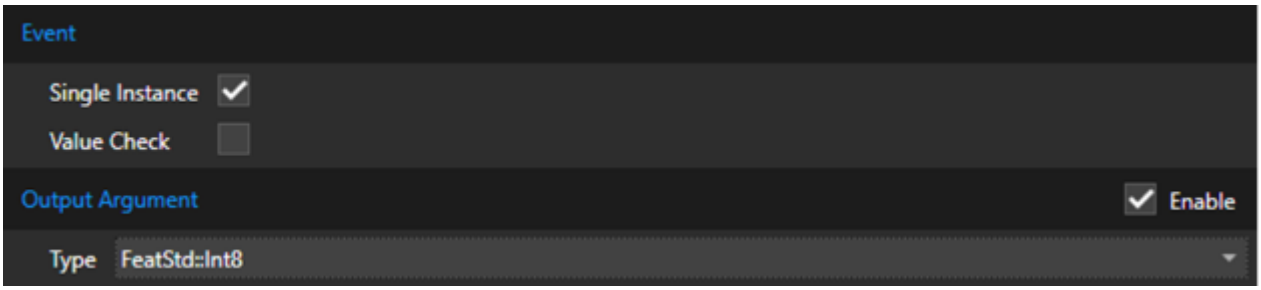
Generic Event Interface

The Generic Event interface is used to signal an event. Instead of being automatically triggered, this interface must be manually triggered from the CGI Application side. For more information on how to do this refer to the TriggerGenericEvent article.

Icon:



Properties:



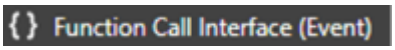
Property Name	Description
Single Instance	When selected, only a single instance of this event will be queue. If the event is triggered again before the previous instance is executed it will overwrite the previous event in the queue instead.
Value Check	If enabled, the sent value will be compared to the previous value and processing will be skipped if it has not changed. This can improve performance if the sent value is often the same/different.
Output Argument	If enabled the event will be accompanied by an output value.
Type	Data type of the value.

Function Call Interface (Event)

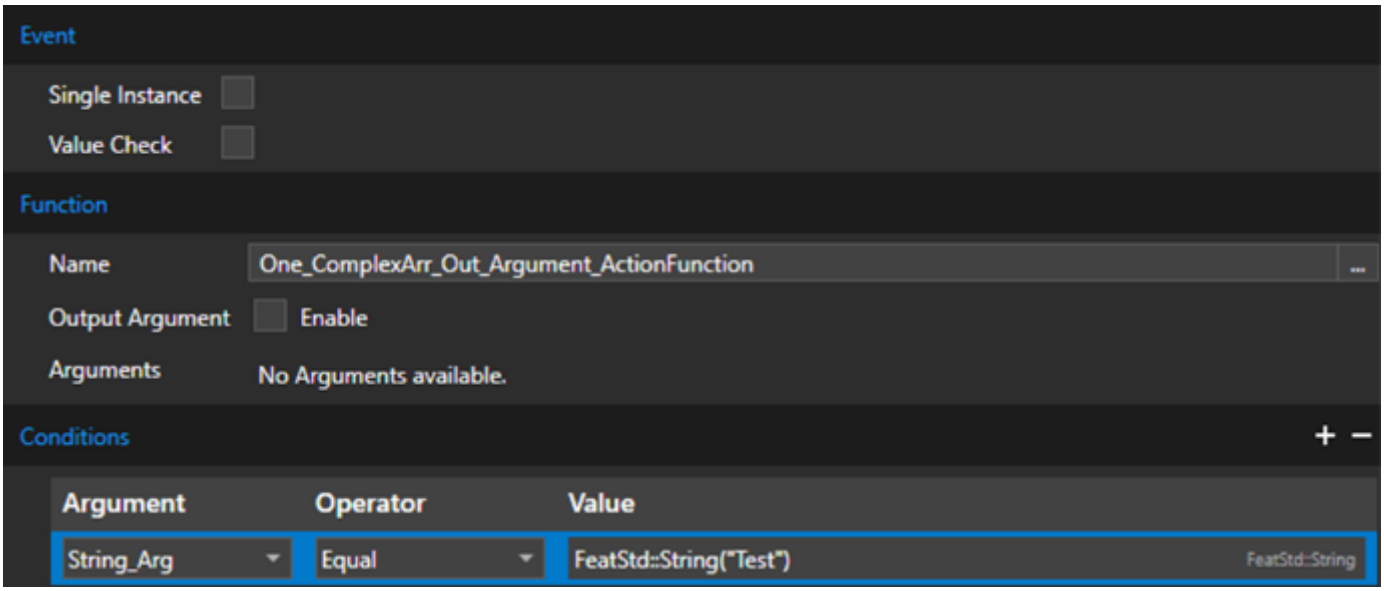
The Function-Call (Event) interface will execute a native function to check if an event has been triggered. Depending on the result of the function call (return value must be true) and specified conditions, an event will be queued.

To learn how to define functions to be used in the Connector refer to the Native Functions chapter.

Icon:



Properties:



Property Name	Description
Single Instance	When selected, only a single instance of this event will be queue. If the event is triggered again before the previous instance is executed it will overwrite the previous event in the queue instead.
Value Check	If enabled, the sent value will be compared to the previous value and processing will be skipped if it has not changed. This can improve performance if the sent value is often the same/different.
Name	Name of the function which triggers the event. Clicking on the '...' button next to the field opens a dialog window which lists all possible messages defined for this application.
Output Argument	If enabled, the event will be accompanied by an output value. This value can be selected from the available members of the related message.
Conditions	Conditions allow to specify additional conditions which the function must fulfill before the event is triggered. If multiple conditions are set, all conditions must be fulfilled. Conditions can be added and removed with the '+' and '-' buttons.
Argument	Name of the argument of the function this condition is applied to
Operator	Operator used to check the condition
Value	Value of the condition. Depending on the type of the argument, the [...] button will be displayed next to the field. When clicking it, it will open a dialog window which lists possible objects from the referenced Scene Composer solution that can be used as a value.

Message Interface (Event)

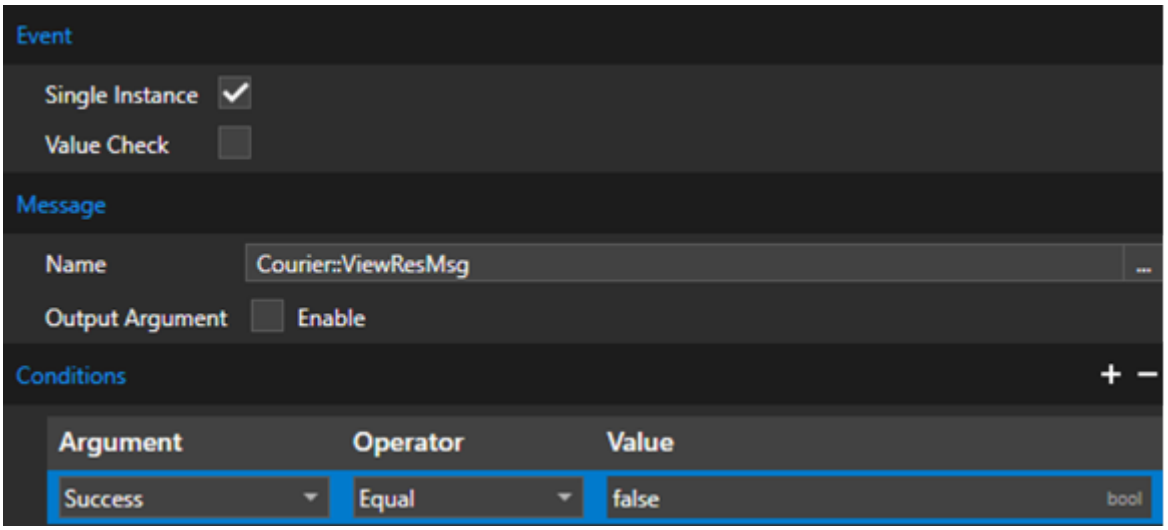
The Message (Event) interface allows to react to incoming Courier messages. If a matching message is received and all conditions are fulfilled an event will be queued.

For more information about Messaging in Courier please refer to the Courier documentation.

Icon:



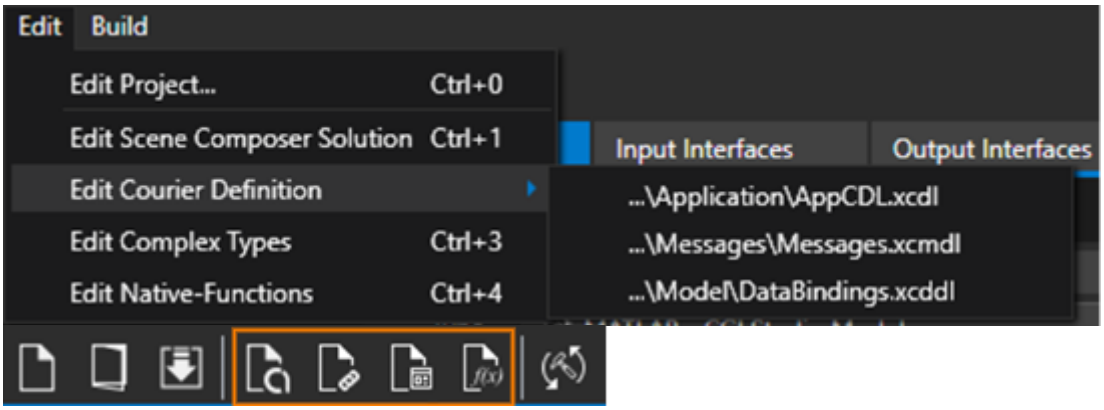
Properties:



Property Name	Description
Single Instance	When selected, only a single instance of this event will be queue. If the event is triggered again before the previous instance is executed it will overwrite the previous event in the queue instead.
Value Check	If enabled, the sent value will be compared to the previous value and processing will be skipped if it has not changed. This can improve performance if the sent value is often the same/different.
Name	Name of the message which triggers the event. Clicking on the '...' button next to the field opens a dialog window which lists all possible messages defined for this application.
Output Argument	If enabled, the event will be accompanied by an output value. This value can be selected from the available members of the related message.
Conditions	Conditions allow to specify additional conditions which the message must fulfill before the event is triggered. If multiple conditions are set, all conditions must be fulfilled. Conditions can be added and removed with the '+' and '-' buttons.
Argument	Name of the member of the message this condition is applied to
Operator	Operator used to check the condition
Value	Value of the condition. Depending on the type of the argument, the [...] button will be displayed next to the field. When clicking it, it will open a dialog window which lists possible objects from the referenced Scene Composer solution that can be used as a value.

Edit Menu

It is possible to quickly edit external resources with via the edit menu or the toolbar.

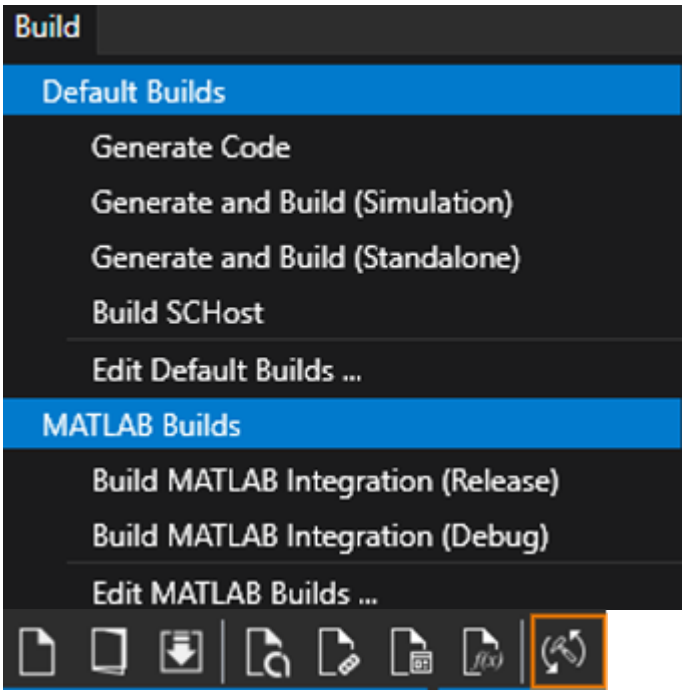


Property Name	Description
Edit Project	Opens the Project configuration window.
Edit Scene Composer Solution	Edits the configured Scene Composer solution.
Edit Courier Definition	Displays all Courier definition files based on the configured XCDL file and allows to quickly edit them using the Courier Editor.
Edit Complex Types	Edits the configured Complex-Types definition file.
Edit Native-Functions	Edits the configured Native-Functions definition file.

> If a resource or related script file is not available, the function is disabled.

Build Menu

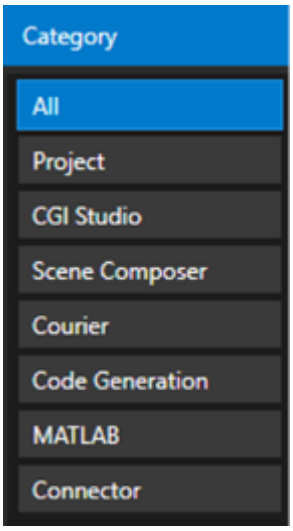
The build menu allows to quickly trigger a build from the specified Build-Projects. It lists all available default builds and, depending on the tool configuration, special tool specific builds. Same can be done via the toolbar.



Additionally, the related Build-Projects can be directly edited from the menu (not available via the Toolbar)

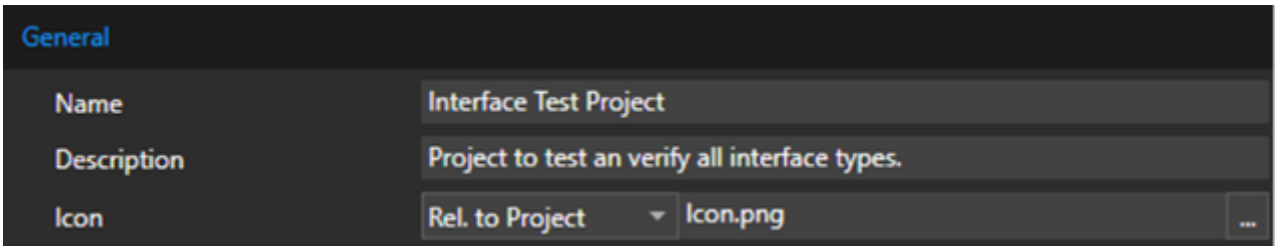
Project Settings

Several project related configurations can be performed in the project settings window which is accessible var the “Edit” menu or the shortcut “Ctrl + 0”.



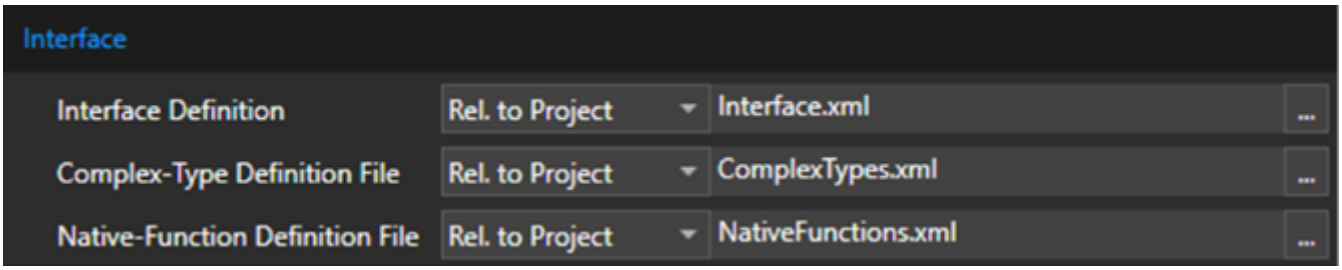
Via the List on the left the displayed options can be filtered to make it faster to find a related setting.
All project settings (with a few additions) are available in any build that is executed. For more details about this refer to Builds chapter.

General



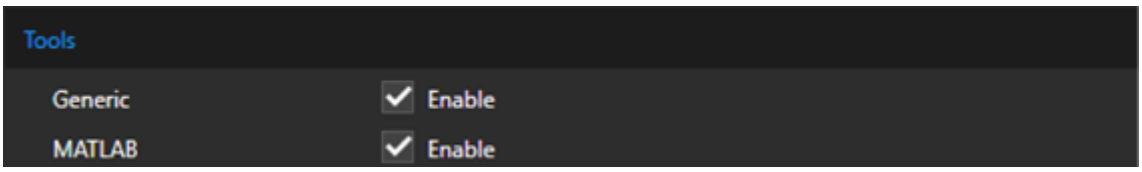
Property Name	Description
Name	Name of the project.
Description	Description of the project.
Icon	Custom icon used for the project.

Interface



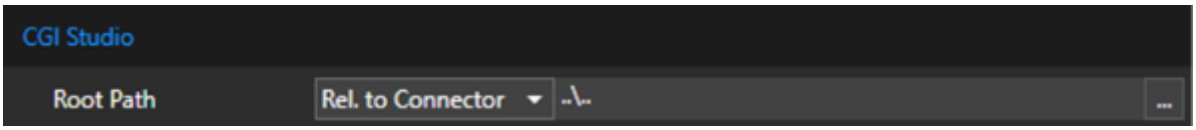
Property Name	Description
Interface Definition	File containing the Interface definition. The path can be either absolute or relative to CGI Connector/Project/CGI Studio path.
Complex-Type Definition	File containing the Complex-Type definition. The path can be either absolute or relative to CGI Connector/Project/CGI Studio path.
Native-Function Definition	File containing the Native-Function definition. The path can be either absolute or relative to CGI Connector/Project/CGI Studio path.

Tools



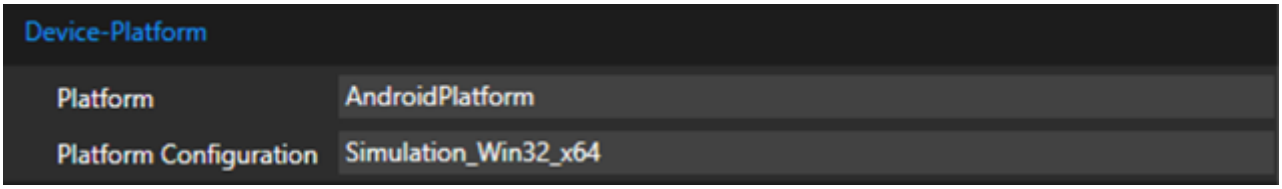
Property Name	Description
Generic	Enables generic support. This will provide a plain C/C++ interface.
MATLAB	Enables support for MATLAB. This will provide additional options/setting related to Mathworks MATLAB support

CGI Studio



Property Name	Description
Root Path	Path to CGI Studio package. The path can be either absolute or relative to CGI Connector/Project path.

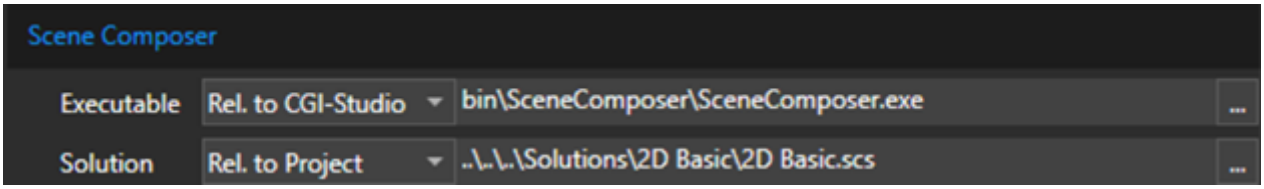
Device-Platform



Property Name	Description

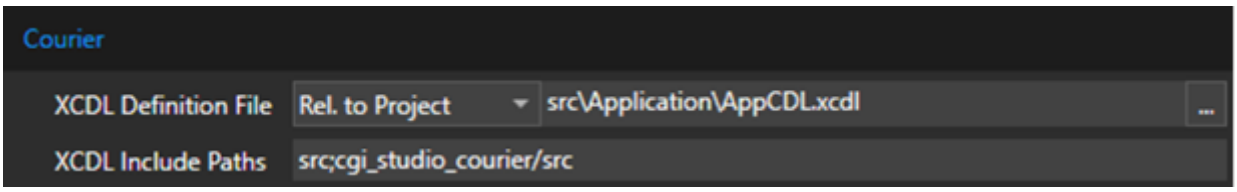
Platform	Name of the CGI Studio Platform (as found in CMake)
Platform Configuration	Name of the CGI Studio Platform configuration (as found in CMake)

Scene Composer



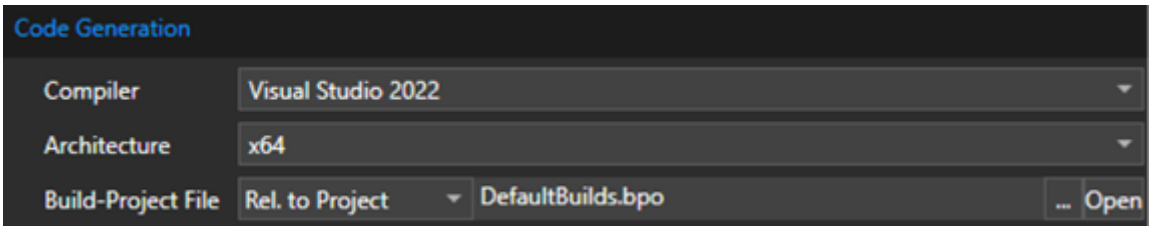
Property Name	Description
Executable	Executable used to edit the Scene Composer solution file.
Solution	Scene Composer solution file. The path can be either absolute or relative to CGI Connector/Project/CGI Studio path.

Courier



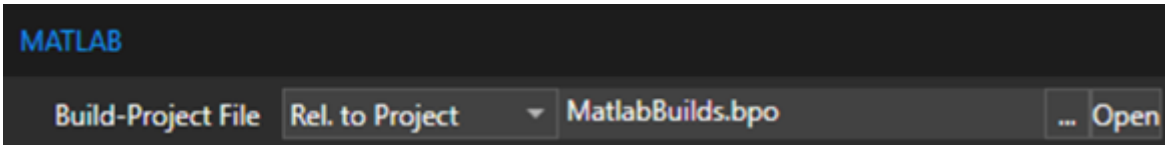
Property Name	Description
XCDL Definition File	File containing the Courier definition. The path can be either absolute or relative to CGI Connector/Project/CGI Studio path.
XCDL Include Path	Additional include path required to by the XCDL definition. Multiple path must be separated by a ‘;’. Paths must be relative to the project file or CGI Studio root path.

Code Generation



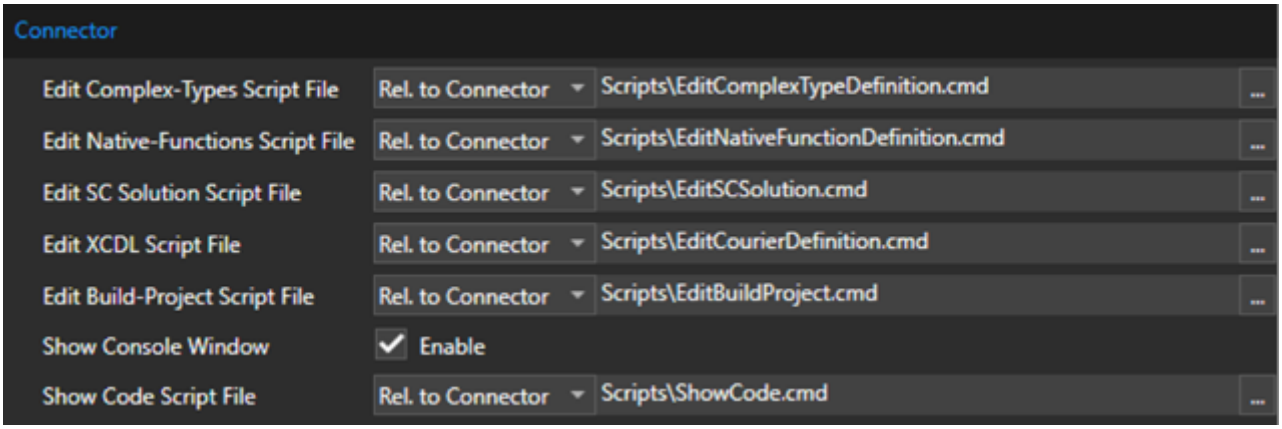
Property Name	Description
Compiler	Compiler used to build code.
Architecture	Architecture used to build the code.
Build-Project File	File containing default builds. The path can be either absolute or relative to CGI Connector/Project/CGI Studio path.

MATLAB



Property Name	Description
Build-Project File	File containing MATLAB related builds. The path can be either absolute or relative to CGI Connector/Project/CGI Studio path.

Connector

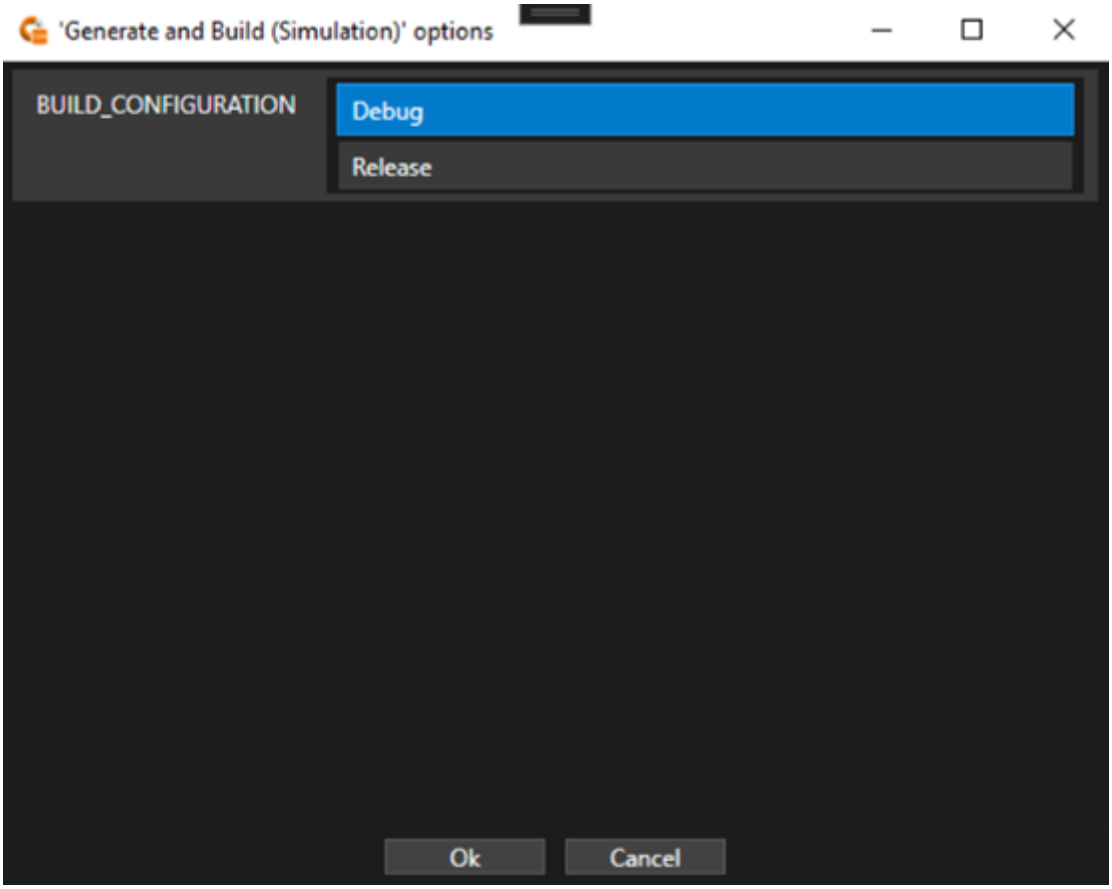


Property Name	Description
Edit Complex-Types Script File	Script file executed when editing Complex-Types. The path can be either absolute or relative to CGI Connector/Project/CGI Studio path.
Edit Native-Functions Script File	Script file executed when editing Native-Functions. The path can be either absolute or relative to CGI Connector/Project/CGI Studio path.
Edit SC Solution Script File	Script file executed when editing SC Solution. The path can be either absolute or relative to CGI Connector/Project/CGI Studio path.
Edit XCDL Script File	Script file executed when editing Courier definition. The path can be either absolute or relative to CGI Connector/Project/CGI Studio path.
Edit Build-Project Script File	Script file executed when editing a Build-Project. The path can be either absolute or relative to CGI Connector/Project/CGI Studio path.
Show Console Window	When enabled, shows a command line window executing scripts. This can be used to debug scripts.
Show Code Script File	Script file executed when editing code (via the log window). The path can be either absolute or relative to CGI Connector/Project/CGI Studio path.

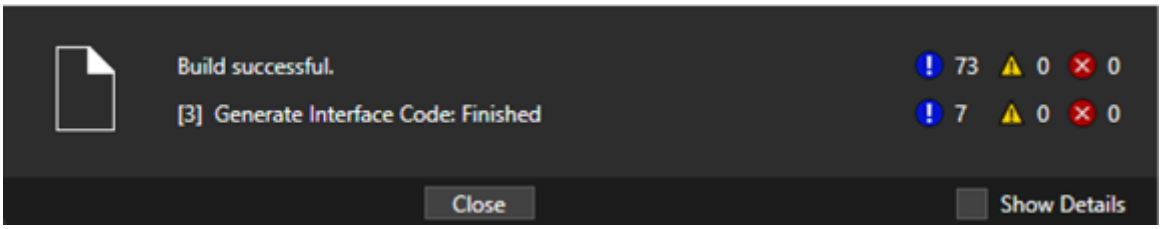
Builds

When a build is started the Build-Window will be shown and the build will be automatically started.

If the build contains some Option-Values an additional dialog will be displayed before the actual Build-Window, which allows the user to configure the build. Depending on the option, the user can either select from a set of predefined values or needs to enter a custom value.



A build is commonly separated into multiple steps each corresponding to a separate script file that is executed from the specified build-script folder.



The top row shows the current build state. It indicates the progress or if the build has finished/failed. On the right side the combined information about all build steps is shown. It displays the total amount of info, warning and error messages from the output. The second row provides information about the current executed step.

When enabling the details, by checking the option in the bottom right, the output of the build scripts can be seen, grouped by the step. This output is also saved into a log-file into the specified build folder.

Build Settings

When a build is started, the builds specific options defined in the related Build Project are set (See Build Projects for more information). Further the following settings are applied by the Connector.

Root-Paths:

Argument	Description
CONNECTOR_ROOT	Path to the Connector binary
PROJECT_ROOT	Path to the Connector project file
CGI_ROOT	Path to the CGI packages

Variables:

Argument	Description
CONNECTOR_EXECUTABLE_PATH	Path to the Connector binary
CONNECTOR_PROJECT_FILE	Connector project file
CONNECTOR_PROJECT_PATH	Path to the Connector project file
CONNECTOR_PROJECT_NAME	Connector project Name
CONNECTOR_PROJECT_DESCRIPTION	Connector project Description
CONNECTOR_PROJECT_ICON_FILE	Connector project icon file
CONNECTOR_INTERFACE_DEFINITION_FILE	Interface definition file
CONNECTOR_COMPLEXTYPE_DEFINITION_FILE	Complex-Type definition file
CONNECTOR_NATIVEFUNCTION_DEFINITION_FILE	Native-Function definition file
CONNECTOR_ENABLE_GENERIC_TOOL	Indicates if ‘Generic Tool’ is enabled
CONNECTOR_ENABLE_MATLAB_TOOL	Indicates if ‘MATLAB Tool’ is enabled
CONNECTOR_CGISTUDIO_ROOT_PATH	Path to CGI Studio root folder
CONNECTOR_CGISTUDIO_PLATFORM	CGI Studio Platform (e.g. Android, RCar,..)

CONNECTOR_CGISTUDIO_PLATFORM_CONFIG	CGI Studio Platform Configuration (x86/x64)
CONNECTOR_SCENECOMPOSER_EXECUTABLE_FILE	Scene-Composer executable file
CONNECTOR_SCENECOMPOSER_EXECUTABLE_PATH	Scene-Composer executable path
CONNECTOR_SCENECOMPOSER_SOLUTION_FILE	Scene-Composer solution file
CONNECTOR_COURIER_XCDL_DEFINITION_FILE	XCDL root definition file
CONNECTOR_COURIER_XCDL_INCLUDE_PATHS	XCDL include paths (separated by ';')
CONNECTOR_COURIER_XCDL_RESOLVED_INCLUDE_PATHS	Resolved XCDL include paths using available root paths
CONNECTOR_CODE_GENERATION_COMPILER	Compiler used for code generation/build
CONNECTOR_CODE_GENERATION_ARCHITECTURE	Architecture used for code generation/build
CONNECTOR_CODE_GENERATION_BUILD_PROJECT_FILE	Default Build-Project file
CONNECTOR_MATLAB_ENABLE_LEGACY_OPTIONS	Indicates if MATLAB legacy options are enabled
CONNECTOR_MATLAB_BUILD_PROJECT_FILE	MATLAB Build-Project file
CONNECTOR_EDIT_COMPLEX_TYPES_SCRIPT_FILE	Script for 'Edit Complex-Type definition'
CONNECTOR_EDIT_NATIVE_FUNTIONS_SCRIPT_FILE	Script for 'Edit Native-Function definition'
CONNECTOR_EDIT_SC_SOLUTION_SCRIPT_FILE	Script for 'Edit Scene Composer solution'
CONNECTOR_EDIT_XCDL_SCRIPT_FILE	Script for 'Edit Courier Definition'
CONNECTOR_EDIT_BUILD_PROJECT_SCRIPT_FILE	Script for 'Edit Build-Project'
CONNECTOR_SHOW_CONSOLE_WINDOW	Indicates if console window is shown when running scripts.
CONNECTOR_SHOW_CODE_SCRIPT_FILE	Script for triggering editor from Build details

The same environment variables are also set when any other script is executed (such as 'Edit Scene-Composer solution')

Close on Success

If a build contains the CONNECTOR_BUILD_CLOSE_ON_SUCCESS variable and sets a value of '1' then the Build-Window will be automatically closed when the build is finished successfully.

This can feature is very useful if Builds are used to just perform simple tasks or open other programs.

> Note that some programs do ‘block’ the command script execution until closed even when started using the START command. To workaround this problem the provided ‘ProcRun’ tool found in ‘cgi_studio_connector\BuildScripts\Common\ProcRun‘ can be used. The syntax to use it is:
> ProcRun <Executable> [“<Arguments>”]
