# Interface

The main purpose of the CGI Studio Connector is to create an interface that allows to easily connect an external system, such as 3rd party Statemachines, custom middleware etc, to a CGI Studio Courier based application. Via this interface it is possible to share data as well as trigger actions and receive events.
To achieve this, multiple classes are generated by the tool which are to be integrated into the CGI-Studio application. These are further supported by multiple behavioral classes which can be implemented/modified by the user to define the exact behavior of the interface.

- Interface-Buffer
- Action-Context
- Scene-Context
- Data-Model

# Interface-Buffer

The main class of the generated interface is called “Interface-Buffer”. This class is responsible to handle the data, actions and events from both sides and provide them to the other side.

The Interface-Buffer class contains the following methods:

## Init

Initializes the Interface-Buffer.

| Argument Name | Type | Description |
|---|---|---|
| **dataModel** | DataModel | The generated Data-Model. Used to initialize data-values of the Interface-Buffer |
| **defaultActionQueueSize** | FeatStd::UInt32 | Size of the Action-Queue in bytes. The action queue is used to store actions for later execution. By pre-initializing it with a proper size, costly runtime memory allocations can be avoided. |
| **defaultEventQueueSize** | FeatStd::UInt32 | Size of the Event-Queue in bytes. The event queue is used to store event for later handling. By pre-initializing it with a proper size, costly runtime memory allocations can be avoided. |

## SetDataValue

Sets data values of the Interface-Buffer. This method is intended to be called by the external system to send data to the CGI application.

| Argument Name | Type | Description |
|---|---|---|
| **interfaceId** | InputInterfaceId::Enum | Id of the interface to set the value. |
| **data** | void* | Pointer to the data to be set. |
| **dataSize** | FeatStd::UInt32 | Size of the data to be shared in bytes.<br><br>This value will be ignored for static sized data unless INTERFACE_DEBUG_SIZECHECK_ENABLED is defined. |

**Result:** True if setting the data-value is successful. False otherwise.

## GetDataValue

Gets data values of the Interface-Buffer. This method is intended to be called by the external system to send data to the CGI application.

| Argument Name | Type | Description |
|---|---|---|
| **interfaceId** | OutputInterfaceId::Enum | Id of the interface to get the value. |
| **target** | void* | Pointer where the value should be written. |
| **maxTargetSize** | FeatStd::UInt32 | Maximum size of in bytes that the 'target' can store.<br><br>This value will be ignored for static sized data unless INTERFACE_DEBUG_SIZECHECK_ENABLED is defined. |
| **dataSize** | FeatStd::UInt32* | Actual size of the data to be written. |

**Result:** True if getting the data-value is successful. False otherwise. If the return value is false and the dataSize > maxTargetSize the cause for the failure was that not enough memory was available to store the output value.

## TriggerAction

Triggers the specified action.

| Argument Name | Type | Description |
|---|---|---|
| **interfaceId** | InputInterfaceId::Enum | Id of the action to trigger. |
| **data** | void* | Pointer to data needed for the action. |
| **dataSize** | FeatStd::UInt32 | Actual size of the data to be written.<br><br>This value will be ignored for static sized data unless INTERFACE_DEBUG_SIZECHECK_ENABLED is defined. |

**Result:** True if getting action was successfully triggered. False otherwise.

## SwapActionQueue

Swaps the action queue.

## ProcessEvents

Gets events of the Interface-Buffer. This method is intended to be called by the external system to send data to the CGI application.

| Argument Name | Type | Description |
|---|---|---|
| **eventHandler** | Internal::EventHandler* | Pointer to the Event-Handler object. This will be called for every event in the queue. |
| **maxCount** | FeatStd::Int32 | Maximum count of events to handle. Use '-1' for unlimited amount. |

The Event-Callback function must have the following arguments:

| Argument Name | Type | Description |
|---|---|---|
| **interfaceId** | FeatStd::UInt32 | Id of the event |
| **data** | void* | Pointer to the event data if available. |
| **dataSize** | FeatStd::UInt32 | Size of the event data if available. |

## ProcessInputDataValues

Synchronizes the Interface-Buffer input data values with the applications Data-Model. This method is intended to be called by the CGI application.

| Argument Name | Type | Description |
|---|---|---|
| **dataModel** | DataModel* | The generated Data-Model. |

## ProcessOutputDataValues

Synchronizes the Interface-Buffer output data values with the applications Data-Model. This method is intended to be called by the CGI application.

| Argument Name | Type | Description |
|---|---|---|
| **dataModel** | DataModel* | The generated Data-Model. |

## ProcessActions

Executes all queued actions. This method is intended to be called by the CGI application.

| Argument Name | Type | Description |
|---|---|---|
| **actionContext** | Action::ActionContext | Action-Context used to execute the action functions and basic actions (e.g. Message-Actions) |

| | | |
|---|---|---|
| **sceneContext** | Scene::SceneContext | Scene-Context used to execute the scene related actions (e.g. Scene-Interface or Scene-Table Interface) |
| **maxCount** | FeatStd::Int32 | Maximum count of actions to handle. Use '-1' for unlimited amount. |

## TriggerGenericEvent

Raises a generic event. This method is intended to be called by the CGI application.

| Argument Name | Type | Description |
|---|---|---|
| **interfaceId** | OutputInterfaceId::Enum | Id of the event to raise. |
| **data** | void* | Pointer to the data of the event if available. |
| **dataSize** | FeatStd::UInt32 | Size of the event data if available. |

**Result:** True if an event was raised (considering additional conditions). False otherwise.

## TriggerMessageEvent

Raises a message event. This method is intended to be called by the CGI application.

| Argument Name | Type | Description |
|---|---|---|
| **message** | Courier::Message* | Message to check if an event is to be raised. |

**Result:** True if an event was raised (considering additional conditions). False otherwise.

## TriggerFunctionCallEvents

Raises Function-Call events. This method is intended to be called by the CGI application.

| Argument Name | Type | Description |
|---|---|---|
| **actionContext** | Action::ActionContext* | Action-Context used to call related Event-Functions. |

**Result:** True if an event was raised (considering additional conditions). False otherwise.

## SwapEventQueue

Swaps the event queue.

# Action-Context

The action context is used whenever any actions (except scene related actions) are executed. Besides some basic methods it must also implement all native-functions that the user wants to call via the interface.

## Required Methods

There are some required methods that need to be implemented in the Action-Context by default.

**ProcessMessage**

Handles messages that are meant to be sent to the CGI application from the Interface-Buffer.

| Argument Name | Type | Description |
| --- | --- | --- |
| **message** | Courier::Message* | Messages to be sent by the method. |

# Scene-Context

The Scene-Context is used whenever scene related actions are executed. Here the user can provide his own implementation on how these actions are to be performed. Besides the Scene-Context itself there are also secondary classes (e.g. Scene/SceneTable) what the user can adapt to fit certain requirements.

## Required Methods

There are some required methods that need to be implemented in the Scene-Context by default.

### SetSceneState

Called when a Scene action is triggered, and the state of a scene is to be changed.

| Argument Name | Type | Description |
|---|---|---|
| **scene** | Scene& | Scene to be manipulated |
| **state** | FeatStd::Int8 | State of the scene to set. |

### SetSceneTableState

Called when a Scene-Table action is triggered, and the state of a Scene-Table is to be changed.

| Argument Name | Type | Description |
|---|---|---|
| **sceneTable** | SceneTable& | Scene-Table to be manipulated |
| **entryId** | FeatStd::Int32 | Id of the Scene-Table entry to be activated. |
| **hint** | const char* | Optional hint string. |

# Data-Model

The basic Data-Model used by the Interface-Buffer is completely generated. However, the user can still override the handling of update requests if needed.