Native Functions

Native Functions allow to execute custom code directly from the interface or receive customizable events. To provide the necessary information to the tool they must be specified in an XML file and finally be implemented in the Action-Context class used by interface.

- XML Definition
- Implementation
- Editor

XML Definition

Native Functions are defined in an XML Schema file which is located at “cgi_studio_connector/schema/NativeFunction.xsd”. This file can be used to validate xml files and with some editors (e.g. Visual-Studio) and provide code-completion when writing the XML file.

NativeFunctionDefinition-Element

Root Element. List of Function types.

Attributes:
None

Children:

Argument	Occurrence	Description
Function	0 - *	A native function definition

Function-Element

Definition of a native Function.

Attributes:

Argument	Required	Description
name	Yes	Name of the function.
uid	Yes	Unique Id of function.
has_result	Yes	Indicates if function has a result. If set, the function needs to return a bool value and can be called for checking events. In this case an event is triggered if the function returns true and all conditions are valid.
description	No	Description of the function.

Children:

Argument	Occurrence	Description
Argument	0 - *	Argument of the function

Argument-Element

Definition of an argument of the function.

Attributes:

Argument	Required	Description
name	Yes	Name of the argument.
uid	Yes	Unique Id of argument.
type	Yes	Type of the argument. This can be any basic or complex type if intended to be used for dynamic input/output. For static values any type is supported.
description	No	Description of the argument.
is_output	No	If set, this argument is passed as a reference with the intend to use it as an output value or in a condition.

Children:

None

Implementation

The actual native functions must be implemented as part of the Action-Context class which will then be passed to the Interface to trigger them.

The implementation must match the name of the specified function in the XML as well as the names, types and order of the arguments. Note that Complex-Types must be passed as const references.

```
<?xml version="1.0"?>
<NativeFunctionDefinition xsi:noNamespaceSchemaLocation="
../../tools/Core/NativeFunction/Schema/NativeFunction.xsd" xmlns:xsi="
http://www.w3.org/2001/XMLSchema-instance" version="1">
   <Function name="FloatTestFunction" uid="test11" has_result="false">
      <Argument name="value" type="FeatStd::Float" uid="arg11" is_output="false"/>
   </Function>
   <Function name="StringTestFunction" uid="test12" has_result="false">
      <Argument name="value" type="FeatStd::String" uid="arg11" is_output="false"/>
   </Function>
   <Function name="MixedArgumentFunction" uid="test13" has_result="true">
      <Argument name="floatValue" type="FeatStd::Float" uid="arg11" is_output="false"/>
      <Argument name="intValue" type="FeatStd::Int32" uid="arg12" is_output="false"/>
   </Function>
   <Function name="EventTestFunction" uid="test26" has_result="true">
      <Argument name="conditionArg" type="FeatStd::Int32" uid="arg11" is_output="true"/>
   </Function>
</NativeFunctionDefinition>
```

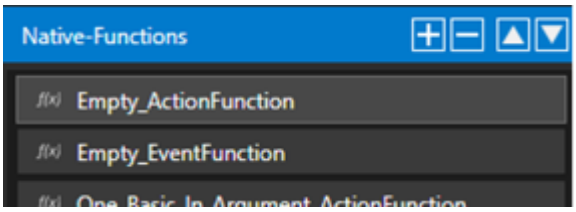
```
namespace Action {
   class ActionContext
   {
      public:
...
         // User Functions
         void FloatTestFunction(FeatStd::Float value);
         void Vector3TestFunction(const Candera::Vector3& value);
         void StringTestFunction(const FeatStd::String& value);
         void MixedArgumentTestFunction(FeatStd::Float floatValue, FeatStd::Int32 intValue);
         bool EventTestFunction(FeatStd::Int32& conditionArg);
```

Editor

An editor for the Native-Function definition can be found here:
“cgi_studio_connector/bin/NativeFunctionEditor.exe”

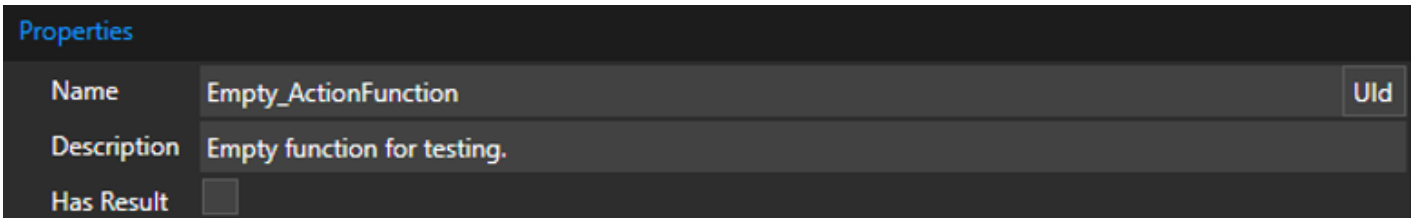
If called with a file as first argument the editor will try to open the specified file.

Native-Functions



On the left side, Native-Functions can be added/removed and reordered using the respective buttons.

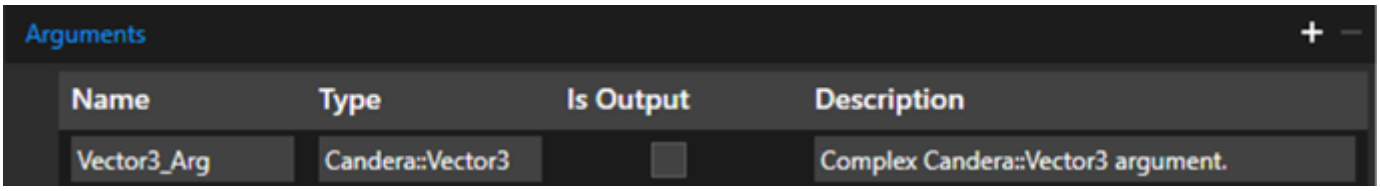
Properties:



Property Name	Description
Name	Name of the function. This name must be a valid C/C++ name and must be implemented in the Action-Context.
UId	The UId is used to identify the native function. It can be seen as the tooltip of the ‘UId’ button. When the button is clicked a new UId will be generated.
Description	Description of the function.
Has Result	If enabled, the function must return a bool value. This indicates that the function is used for Function-Call Events. The return value will indicate if the related event has been raised or not.

Arguments:

A function can have arguments. Arguments can be added/removed with the respective buttons.



Property Name	Description
Name	Name of argument. This name must be a valid C/C++ identifier.
Type	Data type of the argument. In principal all types can be used here. However if the argument is used for a dynamic input/output the type must either be a compatible basic FeatStd type or a supported Complex-Type.
Is Output	Indicates if the argument is used as an output. This can then be used to apply additional conditions when checking for an event.
Description	Description of the argument.