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);
```

Revision #4
Created 2024-08-22 23:06:01 UTC by Kanai Tomoaki
Updated 2025-04-14 03:35:33 UTC by Kanai Tomoaki