Converter

For conversion between the basic type and actual complex type, conversion functions needs to be available. This function must be implemented inside the “ComplexType” namespace (additionally to the default namespace for the Interface) and within the “ComplexType.h/cpp” files.

For fixed size data types the conversion looks like the following:

```
void Convert(FeatStd::Float* from, Candera::Vector3* to)
{
   to->SetX(from[0]);
   to->SetY(from[1]);
   to->SetZ(from[2]);
}

void Convert(Candera::Vector3* from, FeatStd::Float* to)
{
   to[0] = from->GetX();
   to[1] = from->GetY();
   to[2] = from->GetZ();
}
```

For login/tracing purposes a “PrintValue_CUSTOM_TYPE” function is also required. This function’s purpose is to print the given value to the provided buffer. The provided value is the basic type not the final complex type (e.g. float[3] instead of Candera::Vector3). The function must return the bytes written to the buffer.

```
FeatStd::Int PrintValue_Candera_Vector3(void* buffer, FeatStd::UInt32 bufferLength, const void* value)
{
   const Candera::Float* floatValues = (const Candera::Float *)value;
   return FeatStd::Internal::String::StringPrintf((char*)buffer, bufferLength,
                  "[%.2f, %.2f, %.2f]", floatValues[0], floatValues[1], floatValues[2]);
}
```

For dynamic arrays (such as used for strings) the “DataBuffer” class will be used when the type is used in Data-Interfaces and the “Queue” class will be used if this type is used in actions or events. Additionally, a function that converts a void* with size information to the target type is required in both cases.

```
void Convert(FeatStd::String* from, Internal::DataBuffer* to)
{
   to->CopyFrom(from->GetCString(), from->GetCharCount());
}

void Convert(FeatStd::String* from, Internal::Queue* to)
{
   to->WriteSizeData(from->GetCString(), from->GetCharCount());
}

void Convert(const void* data, FeatStd::UInt32 dataSize, FeatStd::String* to)
{
   FEATSTD_UNUSED(dataSize);

   *to = FeatStd::String((FeatStd::TChar*)(data));
}

FeatStd::Int PrintValue_FeatStd_String(void* buffer, FeatStd::SizeType bufferLength,
                                       const void* value, FeatStd::UInt32 valueSize)
{
   return FeatStd::Internal::String::StringPrintf((char*)buffer, bufferLength, "%s", (const char*)value);
}
```

Revision #4
Created 2024-08-22 23:05:21 UTC by Kanai Tomoaki
Updated 2025-04-14 03:35:33 UTC by Kanai Tomoaki