Schema Variant

Table of Contents

Introduction

A schema variant is an object that can hold a single piece of data of a particular type. That data can be assigned a new value of the same or different type.

The fundamental types are:

Type	Description	Function to set
bool	A standard C++ boolean type with true and false values	SchemaVariantSetBool
double	A double precision floating point	SchemaVariantSetNumber
LPCWSTR	A Unicode string	SchemaVariantSetString
void *	Any arbitrary data	SchemaVariantSetPointer

Creating/Destroying

To create a schema variant you call SchemaVariantCreate followed by a call to set some data into it with the appropriate function. Until you have set data into the newly created schema variant the type of the schema variant will be SystemType::Undefined. Calling schema variant functions on undefined schema variants may fail.

When you are finished with a schema variant call SchemaVariantDelete to free the schema variant.

Example

```
void MakeVariant()
{
 SchemaVariantHandle variant = SchemaVariantCreate();
// Type is SystemType::Undefined
 SchemaVariantSetNumber(variant, 123.45f);    // Type
is now SystemType::Number, value 123.45

 SchemaVariantDelete(variant);       // Remember to
delete the variant when finished
}
```

Setting/Changing

You can set/change the value and type of a schema variant with the following calls

Function	Description	Resulting type
SchemaVariantSetBool	Sets the schema variant to a boolean value	SystemType::Boolean
SchemaVariantSetNumber	Sets the schema variant to a double value	SystemType::ANumber
SchemaVariantSetString	Sets the schema variant to a string value	SystemType::AString

SchemaVariantSetPointer	Sets the schema variant to a pointer	The user defined type
SchemaVariantSet	Sets a schema variant from another schema variant	Same as source schema variant

Example

```
void ChangeVariant()
{
 SchemaVariantHandle variant = SchemaVariantCreate();
// Type is SystemType::Undefined
 SchemaVariantSetNumber(variant, 123.45f);    // Type
is now SystemType::Number, value 123.45

 SchemaVariantSetString(variant, L"Hello World");  //
Type is now SystemType::AString
 SchemaVariantDelete(variant);       // Remember to
delete the variant when finished
}
```

When setting the schema variable to an object pointer you specify the type of the stored object in the function call. This can be a TypeHash of your construction or one pre-defined types if the object you are storing matches the type.

Type	Object pointed to
SystemType::Byte	unsigned char
SystemType:Char	char
SystemType:;DateTime	Qbik private
SystemType::Int16	Signed 16 bit integer
SystemType::Int32	Signed 32 bit integer
SystemType::Int64	Signed 64 bit integer
SystemType::IP4Address	Qbik private
SystemType::UInt16	Unsigned 16 bit integer
SystemType::UInt32	Unsigned 32 bit integer
SystemType::UInt64	Unsigned 64 bit integer
SystemType::Single	Single precision float
SystemType::Double	Double precision float
SystemType::String	std::wstring
SystemType::StringA	std::string

> In typical situations the fundamental type Double can handle most needs. The other fundamental types are there primarily for support of schema members registered with SchemaClassRegisterMember so the schema framework knows how to access the member within the offset of the object.

Classes

Sometimes the schema framework may need to call functions for a schema variant in order to perform operations like casts. This may require the schema framework to know what class the schema variant belongs to. If you will need to perform class based operations on your schema variants then you should set a schema class for the schema variant by calling SchemaVariantSetSchema

Coercion

Coercion of schema variants is the act of forcing a schema variant to another type. The schema variant set functions (SchemaVariantSetBool etc) perform a coercion to the requested type before setting the value. The rules for coercion are as follows.

- If the requested type is the same as the schema variant type then no change to the schema variant occurs
- If the requested type is different to the schema variant type and the requested type is one of the standard types (Undefined, ANumber, AString or Boolean) then the type is changed and the value of the variant is set to the default for that type.
- If the requested type is any other type the type of the schema variant is changed but the value is not. In this situation you should not read from the schema variant as the value will likely NOT be what you want.

> You can also coerce a schema variant yourself with SchemaVariantCoerce