Type Information

This chapter covers the following topics:

- Overview
- NaturalX and Type Information
- Using Type Information

Overview

Type information is a means to completely describe a class along with all of its interfaces, down to the names and types of the methods. It contains the necessary information about classes and their interfaces, for example, which interfaces exist on which classes, which member functions exist in those interfaces, and which argument those functions require.

This information is used by clients to find out details about a class and its methods, for example, by type-information browsers to present available objects, interfaces, methods and properties to an end user.

Another important area for using type information is the widely-used OLE automation technique which is also used by NaturalX.

There are several ways to store type information. A common way is generating the type information in type library (.TLB) files.

NaturalX and Type Information

Creating Type Information

For each Natural class, a type library file is created when the class is registered.

The type library is generated in the *$NATDIR/$NATVERS/etc/<serverid>/<classname>/<version>* directory and connected to the class via an entry in the registry.

The name of the class module is used, and the ".tlb" extension is appended unless the type library file name conflicts with an existing name. Then a number is attached to the class module name.

Using Type Information

Each interface defined in a Natural class is seen by clients as a dynamic interface (also called a "dispatch interface"). Each method of an interface is seen by clients under the name defined in the `METHOD` statement.

The first interface in a Natural class is marked as the default dispatch interface.

The support of type information also makes it possible to define multiple interfaces with identical method/property names. The Natural client simply addresses the corresponding method by using the interface name (as defined in the Natural class) as the prefix of the method name, as shown in the

following example:

```
CREATE OBJECT #O3 OF CLASS "DepartmentList"
SEND "Iterate.PositionTo" TO #O3 WITH "C" RETURN #DEPT
```

Natural clients use type information to find out to which interface a method or property belongs.

Note:
Natural clients do not use type information at catalog time to perform syntax checks.

Data Type Conversions

The following topics are covered below:

- Natural Data Formats to OLE Types
- OLE Types to Natural Data Formats

Natural Data Formats to OLE Types

In order to receive data from clients or to pass data to classes written in different programming languages, the Natural data formats are converted to so-called OLE Automation-compatible types. This table shows how COM clients see the method parameters or properties of a Natural class. For example, if a Natural class has a method parameter or a property with the format A, this is seen by a COM client as VT_BSTR.

Natural Data Format	Automation-Compatible Type
A	VT_BSTR
B1	VT_UI1
B2	VT_UI2
B4	VT_UI4
B*n* (*n* != 1, 2, 4)	SAFEARRAY of VT_UI1
C	not supported
D	VT_DATE
F4	VT_R4
F8	VT_R8
I1	VT_I2
I2	VT_I2
I4	VT_I4
HANDLE OF GUI	not supported
HANDLE OF OBJECT	VT_DISPATCH
L	VT_BOOL
N15.4	VT_CY
N*n.m* (*n.m* != 15.4)	VT_R8
P15.4	VT_CY
P*n.m* (*n.m* != 15.4)	VT_R8
T	VT_DATE
U	VT_BSTR

An array of a given Natural data format is mapped to a SAFEARRAY of the corresponding "VT" type.

There are, however, some special cases:

- A variable of format B*n* with fixed length, where *n* is not 1, 2 or 4, or an array of such a variable, is mapped to a one-dimensional SAFEARRAY of VT_UI1. This is for compatibility with previous versions of Natural, where large and dynamic variables were not yet supported. Therefore, large binary variables had to be simulated by arrays of variables of type B with fixed length.
- A dynamic variable of format B is mapped to a one-dimensional SAFEARRAY of VT_UI1.
- An array of dynamic variables of format B is mapped to a SAFEARRAY of variants, each containing a one-dimensional SAFEARRAY of VT_UI1.
- Attribute control variables are not mapped. They have no meaning outside of Natural. Variables of format HANDLE OF GUI are also not mapped. There is no corresponding Automation-compatible type. Therefore properties of the formats Attribute control variable or HANDLE OF GUI cannot be accessed by clients through COM/DCOM. Method parameters of these types should be marked as

optional in the parameter data area, so that clients can omit the parameters when calling the method through COM/DCOM.

OLE Types to Natural Data Formats

This table shows how parameters or properties of an external class can be addressed by Natural. For example, if an external class has a method parameter or property with type VT_R4, this parameter or property can be addressed in Natural as F4 or with a format that is MOVE-compatible to F4.

Automation -Compatible Type	Natural Data Format
VT_BOOL	L
VT_BSTR	A or U
VT_CY	P15.4
VT_DATE	T
VT_DISPATCH	HANDLE OF OBJECT
VT_UNKNOWN	HANDLE OF OBJECT
VT_I1	I1
VT_I2	I2
VT_I4	I4
VT_INT	I4
VT_R4	F4
VT_R8	F8
VT_U1	B1
VT_U2	B2
VT_U4	B4
VT_UINT	B4

A SAFEARRAY of up to three dimensions is converted into a Natural array with the same dimension count and the corresponding format. SAFEARRAYs with more than three dimensions cannot be used from within Natural.

There are, however, some special cases:

- A VT_BSTR maps either to a Natural variable of format A or to a one-dimensional array of Natural variables of format A with fixed length. The additional dimension is then used to store strings longer than 253 characters. This is for compatibility with previous versions of Natural, where large and dynamic variables were not yet supported. This mapping should no longer be used. Instead, a dynamic variable of format A should be used.
- A SAFEARRAY of VT_BSTRs maps either to an array of Natural variables of format A with the same dimension count, or to an array of Natural variables of format A with fixed length with one more dimension. The additional dimension is then used to store strings longer than 253 characters. This is for compatibility with previous versions of Natural, where large and dynamic variables were not yet

supported. This mapping should no longer be used. Instead an array of dynamic variables of format A should be used.

- A SAFEARRAY of VT_UI1 can be mapped to an array of Natural variables of format B with fixed length that has a matching total size. This is for compatibility with previous versions of Natural, where large and dynamic variables were not yet supported. This mapping should no longer be used. Instead a dynamic variable of format B should be used.