# Tips and Tricks for the DCOM Wrapper

This chapter provides the following tips and tricks for using EntireX DCOM Wrapper:

- IDL Parameter Definitions
- Groups in IDL Parameter Definition
- Arrays in Groups
- ActiveX Application Calling a DCOM Wrapper Method
- Problem with the #import Clause of the Client Application
- Syntax Errors
- ActiveX Automation Server
- Restrictions on Parameter Names
- Language-dependent Restrictions on the Client Side
- Using DCOM Wrapper Object with Microsoft Visual Studio .NET
- Using N, NU, P, PU Data Types

See also *Using DCOM Wrapper Objects with Web Scripting Languages*.

---

## IDL Parameter Definitions

The parameter definition in the IDL file must match exactly the parameter data area of the Natural subprogram with respect to format, length, dimensions, direction, number and order of parameters.

## Groups in IDL Parameter Definition

Because the number of required GUIDs may change if the parameter definition in the IDL file changes with respect to groups, you must deregister the old object and delete the *g<library>.h* file before (re-)generating the object.

## Arrays in Groups

From EntireX 5.3.1.10 on, methods for better array support are available for arrays defined in groups.

```
set property
<object name>.<access to group>.<array name>_indexAccess(<index list>) = value

get property
<variable> = <object name>.<access to group>.<array name>_indexAccess(<index list>)
```

# ActiveX Application Calling a DCOM Wrapper Method

When the ActiveX application calls a DCOM Wrapper method, the parameters must match exactly:

- the number of parameters
- the parameter type (string, int etc.)
- how these parameters are passed to the called method:
  - `IN` parameters should be passed by value;
  - `OUT` and `INOUT` parameters should be passed by reference.

If in doubt, look at the type library of the generated Wrapper object. Use the Microsoft C++ tool OLE/COM Object Viewer (*oleview.exe*) to inspect the type library, or look at the generated IDL file in the generated object directory.

# Problem with the #import Clause of the Client Application

If the dll generated by EntireX DCOM Wrapper is imported by a client application, the compiler can exit with error "error C2011:’IServiceProvider’ : ’struct’ type redefinition". If this happens, add the following to the #import clause:

```
rename("IServiceProvider","IServiceProviderX")
```

Example:

```
#import "<path>\DCOMWrapperObject.dll" no_namespace rename("IServiceProvider","IServiceProviderX")
```

If the dll generated by EntireX DCOM Wrapper is imported by a client application and this application is an ATL application, it may be necessary to import the dll type library with the following statement:

```
#include <dispex.h>
#import "DCOMWrapperObject.tlb"  no_namespace  exclude("IDispatchEx")
exclude("IServiceProvider") rename ("IDispatch","IDispatchEx")
```

# Syntax Errors

If syntax errors occur in the IDL file, the most likely cause is the use of reserved words as parameters. Not only the keywords of IDL (library, program, etc.) are reserved, but also all valid format-length combinations. For example, you cannot use "A1" as the name of a parameter.

If syntax errors occur during compilation of the generated C++ sources, a likely cause is the use of reserved C++ words as parameter names in the IDL file, or a library that contains a program with the same name.

# ActiveX Automation Server

If you generate an ActiveX automation server a second time, the existing GUIDs will be reused. To prevent this, delete the header file *g<library>.h* in the generated object directory.

# Restrictions on Parameter Names

Parameter names used in the IDL file conflict with the Visual C++ MIDL compiler, so avoid keywords used in the MIDL definition. The DCOM Wrapper uses the MIDL compiler from EntireX Version 6.1.1.8 on (only Windows platforms). The changes were necessary because of the internal change from the IDispatch to the IDispatchEx interface to support scripting languages correctly.

# Language-dependent Restrictions on the Client Side

On the client side, language-dependent restrictions may occur independent of the functionality offered by the DCOM Wrapper object.

# Using DCOM Wrapper Object with Microsoft Visual Studio .NET

While using the Wrapper-generated objects, the .NET application throws an error message queryinterface <interface name> failed. To prevent this, uncheck **ASP scripting support** in the *Setting DCOM Wrapper Properties*.

If the Extended Interface was used, the client application has to make sure that the Wrapper-generated object was called from an STA thread. In a ASP.NET application, the client application can use the `aspcompat` attribute on your ASP.NET page or if the application, such as Web service, does not provide an attribute such as `aspcompat`, use an STA thread in your application.

## ASP.NET Example

Include the attribute `aspcompat` into the file *WebForm.aspx*.

```
<%@ Page aspcompat=true Language="vb".........
```

## Web Service Example

```
' Create new thread class
Public Class STAThreadClass
    Dim Obj As EXXDCOMTypeLibrary.ProgClass
    Dim sUser As String

    Public Sub Run()
        ' Set thread state to STA
        System.Threading.Thread.CurrentThread.ApartmentState = Threading.ApartmentState.STA
        Obj = New EXXDCOMTypeLibrary.ProgClass()

        Try
            Obj.ServerAddress = "localhost@RPC/SRV1/CALLNAT"
            Obj.TEST(sUser)
            ' catch an error, if one thrown
        Catch ex As Exception
            ReturnValues.ErrorText = "Error: " + ex.ToString
```

```
            End Try
            Obj = Nothing
        End Sub
    End Class

    ' main web service function
    <WebMethod()> Public Function DoCall()

        Dim STA As New STAThreadClass()

        ' create the thread
        Dim InstanceCaller As New Thread(New ThreadStart(AddressOf STA.Run))

        ' start the thread and wait for execution
        InstanceCaller.Start() 'start thread
        InstanceCaller.Join() 'wait for called STA thread

    End Function
```

# Using N, NU, P, PU Data Types

When using N, NU, P or PU data types, the only valid decimal character is the decimal point.

Examples:

```
1234.34                 OK
1234,34                 not allowed
123,400.00              not allowed
```

Only the decimal point "." is allowed!