NEURON DATA

INSTALL NOTES

Elements Environments
Windows 95 and NT Users

Interoperable Objects OLE Server Installation

This note describes:

- the Elements Environment OLE implementation of Interoperable Objects
- installation of the OLE servers
- how to configure machines with Distributed COM to allow the Elements Environment OLE out-of-process servers to be accessed remotely over the network

Overview

The OLE implementation of Interoperable Objects provides access to the Neuron Data Elements Environment via OLE automation servers. The Elements Environment OOScript, scripting language can access these ND servers and any other OLE automation servers (including Excel, Word, and so forth).

In addition any OLE automation clients developed from the following tools can access the Elements Environment servers.

- Visual Basic
- Visual C++
- Visual J++

OLE allows the creation of several types of servers:

- In-process (DLL)
- Local (EXE)
- Local single use (EXE)

There are three variations of each Elements Environment server which correspond to the above OLE server types.

OLE in-process server

Is a DLL which is loaded into the client's address space and thus allows the creation and manipulation of the servers objects in the address space of the client. This is the most efficient way to access the objects and it is typically the way they will be accessed from the Elements Environment OOScript language. The Neuron Data in-process servers are registered under names beginning with `ND`.

OLE local server

Is a EXE which provides access to clients which are in a different address space from the server. Multiple client processes may access an OLE local server. The ND local servers are registered under names beginning with `NDLOCAL`.

OLE local single user server

Is a local server which only allows a single external client process to access the server. The ND local single user servers are registered under names beginning with `NDSINGLE`.

Any Elements Environment OOScript-based application can also be an automation server so that it can be manipulated by another Elements Environment application or any OLE client.

The table summarizes the servers provided by the Neuron Data Elements Environment.

Server	Name in scripting language	Server type	Id	Files
Core server (access to objects from the ndcore and ndres libraries)	ND.Core	In-process Core server	co	ndcosvr.dll, ndcosvr.lib
	NDLOCAL.Core	Local Core server		colocsv.exe
	NDSINGLE.Core	Local single use Core server		cosglsv.exe
Gui server (access to objects from the ndvgm, ndtkit and ndgw libraries)	ND.Gui	In-process Gui server	gu	ndgusvr.dll, ndgusvr.lib
	NDLOCAL.Gui	Local Gui server		gulocsv.exe
	NDSINGLE.Gui	Local single use Gui server		gusglsv.exe

Server	Name in scripting language	Server type	Id	Files
Data access server (access to objects from the nddb library)	ND.Dae	In-process Dae server	da	nddasvr.dll, nddasvr.lib
	NDLOCAL.Dae	Local Dae server		dalocsv.exe
	NDSINGLE.Dae	Local single use Dae server		dasglsv.exe
Data access gui server (access to objects from the nddbgw library)	ND.DaeGui	In-process DaeGui server	dg	nddgsvr.dll, nddgsvr.lib
	NDLOCAL.DaeGui	Local DaeGui server		dglocsv.exe
	NDSINGLE.DaeGui	Local single use DaeGui server		dgsglsv.exe
Rules Element server (access to Rule-based objects)	ND.Nx	In-process Nx server	nx	ndnxsvr.dll, ndnxsvr.lib
	NDLOCAL.Nx	Local Nx server		nxlocsv.exe
	NDSINGLE.Nx	Local single use Nx server		nxsglsv.exe

Requirements

Requires Windows 95 or Windows NT 3.51 or later.

Environment Variables

The directories containing the ND DLL's and any DLL's used by an Elements Environment OOScript application which is a server must be on the system path.

The directories containing the `NDXxx.dat` files and any `.dat` files used by an Elements Environment application which is a server must be specified by the environment variable ND_PATH.

The Path and ND_PATH environment variables must be defined **system wide at boot time** in order to be effective. This is done in `autoexec.bat` in Windows 95. In windows NT these variables must be set by the administrator in the System Environment Variables section of the System control panel.

These variables are normally set up by the installation process. If the user needs to change the Path or ND_PATH variables they must be changed as described above and then **the system must be rebooted for the change to be effective.**

Registration/Unregistration

Before a server can be accessed from the Elements Environment OOScript language or any OLE automation client it must be registered in the system registry.

Registration by the Installer

The servers are normally registered at installation time by the installer. The installer also creates batch files which can be used to unregister and register all the servers at a later time.

Product	Server Registration	Server Unregistration
C Libaries	`regc.bat`	`unregc.bat`
C++ Libraries	`regcpp.bat`	`unregcpp.bat`

NOTE: The servers for the C and C++ products cannot be registered at the same time. Registering the servers for one product erases the registration for the other product.

Switching between the Servers for the C and C++ Products

If you install both the C and C++ products on the same system, the installer will ask which set of servers should be registered (C or C++). If initially you will be running or developing an OOScript application using the C product, you should choose to register the C servers. If initially you will be running or developing an OOScript application using the C++ product, you should choose to register the C++ servers.

Registration/Unregistration of Individual Servers

To register a server the user must run each local server from the DOS prompt with the `/Register` switch.

For example to register the Core servers type the following in DOS window after installation:

```
colocsv /Register
cosglsv /Register
```

The servers can be unregistered using the `/Unregister` switch:

```
colocsv /Unregister
cosglsv /Unregister
```

Using `/Unregister` on the local server executable will unregister the local and in-process servers. Using `/Unregister` on the single user local server executable will unregister the single user local server.

The names of the local server executables for each server are listed in table 1.1.

Accessing OLE Servers from the OOScript Language

To access OLE servers from the Elements Environment OOScript language, use the built-in `getserver` function.

For example to access the Gui in-process server:

```
guiserver = getserver("ND.Gui");
```

to access the Rules Element local server:

```
rulesserver = getserver("NDLOCAL.Nx");
```

Any OLE automation server can be accessed from the OOScript language using the built-in `getserver` function.

For example to access the Microsoft Excel server

```
excelserver = getserver("Excel.Application");
```

Accessing ND Servers from any OLE Client

To access the ND servers from any OLE automation client, use the client's `create server` function.

For example to access the Rules Element local server from Visual Basic:

```
Dim rulesServer As Object
Set rulesServer = CreateObject("NDLOCAL.Nx")
```

to access the Core in-process server from Visual Basic:

```
Dim coreServer As Object
Set coreServer = CreateObject("ND.Core")
```

Turning any ND Application into an OLE Local Server

Any Elements Environment application can be turned into an OLE local server by adding server initialization calls to the application's initialization. An application may initialize more than one server. For example an application may provide access to its Gui and Data access objects by initializing both the Gui and Dae servers.

Server initialization and exit code must be added to the application's main program for each server for which the application wants to provide access.

- The main program must include the header file `xxsvrapp.h` where `xx` is one of: `co`, `gu`, `da`, `dg`, `nx`. See the Id's in table 1.1.
- The server should call the `XXServerApp_ProcessArgs` function

after the applications main initialization and before the call to `XXServerApp_Initialize`.

- The server must be initialized by calling the `XXServerApp_Initialize` function after the application's main initialization and before the main event loop (Run).
- The server must be uninitialized by calling the `XXServerApp_Uninitialize` function after the application's main event loop and before the applications exit code.

The `Initialize` and `ProcessArgs` calls must be passed an application name and a description string which is used to register the application's server in the system registry. For a detailed description of these calls see the corresponding *xx*`svrapp.h` file.

C Example

To make the application MyApp into a server which allows access to the application's Core and Gui objects via the names `MyApp.Core` and `MyApp.Gui`:

Listing 2-1 C Example

```
#include <cosvrapp.h>
#include <gusvrapp.h>

.
.
.

int main L2(int, argc, char C_FAR* C_FAR*, argv)
{
ERR_MAININIT;
ERR_MODULEUSE;

ND_Init(argc, argv);

/*
 | Process /Register, /Unregister switches
 | for MyApp.Core server.
   */
   COServerApp_ProcessArgs(argc, argv, òMyAppó,
                              òMy application Core Serveró);
/*
 | Process /Register, /Unregister switches
 | for MyApp.Gui server.
   */
GUServerApp_ProcessArgs(argc, argv, òMyAppó,
                              òMy application Gui Serveró);


/* Initialize and create factory for MyApp.Core server */
   COServerApp_Initialize(òMyAppó, òMy application Core Serveró);

/* Initialize and create factory for MyApp.Gui server */
   GUServerApp_Initialize(òMyAppó, òMy application Gui Serveró);
.
.
.

ND_Run();

/* Uninitialize MyApp.Core server */
   COServerApp_Uninitialize();

/* Uninitialize MyApp.Gui server */
   GUServerApp_Uninitialize();

ND_Exit();

return EXIT_OK;
}
```

C++ Example

To make the application MyApp into a server which allows access to the application’s Data Access and Rules Element objects via the names `MyApp.Dae` and `MyApp.Nx`:

Listing 2-2 C++ Example

```
#include <dasvrapp.h>
#include <nxsvrapp.h>

.
.
.

int main L2(int, argc, char C_FAR* C_FAR*, argv)
{
ERR_MAININIT;
ERR_MODULEUSE;

ND::Init(argc, argv);

/*
 | Process /Register, /Unregister switches
 | for MyApp.Dae server.
    */
   DAServerApp::ProcessArgs(argc, argv, òMyAppó,
                               òMy application Data Access Serveró);
/*
 | Process /Register, /Unregister switches
 | for MyApp.Nx server.
    */
NXServerApp::ProcessArgs(argc, argv, òMyAppó,
                              òMy application Rules Serveró);


/* Initialize and create factory for MyApp.Dae server */
   DAServerApp::Initialize(òMyAppó, òMy application Data Access Serveró);

/* Initialize and create factory for MyApp.Nx server */
   NXServerApp::Initialize(òMyAppó, òMy application Rules Serveró);
```

```
.
.
.

ND::Run();

/* Uninitialize MyApp.Dae server */
   DAServerApp::Uninitialize();

/* Uninitialize MyApp.Nx server */
   NXServerApp::Uninitialize();

ND::Exit();

return E
```

Turning any ND Application into an OLE Remote Server

Elements Environment applications running under Windows NT 4.0 can be turned into an OLE remote server by configuring both the client and server machines for Distributed COM (DCOM) from Microsoft.

The addition of Distributed COM (DCOM) to Windows NT 4.0 allows the Elements Environment-OLE, out-of-process servers to be accessed remotely over the network. In order for a Visual Basic or OOScript client to access an Elements Environment server over the network, both the client and server machines need to be configured using the procedure below.

Since the remote servers are necessarily out-of-process (either `NDLOCAL.`*`xxx`* or `NDSINGLE.`*`xxx`*), the client is subject to the same restrictions as described in the Elements Environment OOScript Programmer's Guide for out-of-process servers. In particular callbacks (notifications) cannot be used.

Note: To use the servers remotely, it is necessary to configure both the client and server machines using the `dcomcnfg.exe` utility in NT 4.0. Typically this utility is located in `C:\winnt\system32`.

Configuring the Server Machine

The server machine must have Windows NT 4.0 running to configure DCOM.

1. Make sure that the Elements Environment servers have been installed and registered on the server machine.
2. Run the `dcomcnfg` utility.
3. Select the server you wish to make available for remote access from the displayed list of applications and click the **Properties...** button.

If both the local and single-use servers are registered, you will see two entries for each server. For example:

```
Neuron Data Core Server Object(1)
Neuron Data Core Server Object(2)
```

To determine which one to use, look at the local path of the executable displayed in the **General** tab:

– If the name of the executable is of the form `xxLOCSV.EXE`, then you have selected the local server.

– If it is of the form `xxSGLSV.EXE`, then you have selected the single use server.

4. Click on the **Location** tab and make sure that **Run application on this computer** is checked.
5. Click on the **Identity** tab and choose the account to launch the server when it is requested by a client:

   – If you select **The launching user**, the user of the client must have an account on the server machine.

   – Otherwise select **This user** and specify a user account to be used to run the server. The server will be run in the background under the specified account.

   Selecting **The interactive user** will cause the server to be launched on the desktop. This can be useful for debugging to see output from the server or for servers which display monitoring windows.
6. You can use the **Security** tab to specify any additional security restrictions to be used for the server.
7. Click on OK.

   The server machine is now configured to run the server.

Configuring the Client Machine

The client machine must have Windows NT 4.0 running to configure DCOM.

Note: The Elements Environment servers **must** be installed and registered on the client machine. This is necessary to setup the registration used by `dcomcnfg` even if you do not intend to use the servers locally on the client.

If you only intend to use the servers remotely from a Visual Basic client, you can remove the Elements Environment software from the machine **after** registering the servers. Do **not** unregister the servers when removing the Elements Environment software.

1. Run the `dcomcnfg` utility.
2. Select the server you want to make available for remote access from the displayed list of applications and click the **Properties...** button.

If both the local and single use servers are registered you will see two entries for each server. For example:

```
Neuron Data Core Server Object(1)
Neuron Data Core Server Object(2)
```

To determine which one to use, look at the local path of the executable displayed in the **General** tab:

– If the name of the executable is of the form *xx*`LOCSV.EXE`, you have selected the local server.

– If it is of the form *xx*`SGLSV.EXE`, you have selected the single-use server.

3. Click on the **Location** tab and check **Run application on the following computer**.
4. Enter the name or IP address of the server machine in the text entry field.
5. Uncheck the other options.
6. Click on OK.

   The client machine is now configured to run the server remotely.

Accessing a Remote Elements Environment Server from Visual Basic or OOScript

After configuring the server that you want to access as described in the previous sections it is only necessary to use **CreateObject** in Visual Basic or **getserver()** in OOScript to connect to the server.

You must specify either the local, out-of-process server or the single-use, out-of-process server. For example:

```
nxSvr := getserver(“NDLOCAL.Nx”);  // Local server
```

or

```
nxSvr := getserver(“NDSINGLE.Nx”); // Single-use server
```

Using the local server permits multiple clients to share the same server.

Using the single-use server will result in each client getting its own server process.

You can also build a custom local or single-use server which contains one or more EE servers in the same address space. For example, Nx and Dae. See the EE OLE server documentation for more information.