



NetWeave Interface Class for .NET

Installation, Configuration and User's Guide

Revised 03 March 2007

www.netweave.com

Copyright © NetWeave Integrated Solutions, Inc. 2003-2008. All rights reserved.

NetWeave is a registered trademark of NetWeave Integrated Solutions, Inc.

Windows is a registered trademark of Microsoft Corporation.

CICS, MVS, and MQSeries are registered trademarks of the IBM Corporation.

UNIX is a registered trademark of X/OPEN Ltd.

Tandem, Guardian, VMS, and OpenVMS are registered trademarks of HP.

All other trademarks are noted in the text and are the property of their respective owners.

Table of Contents

INTRODUCTION.....	1
NWDS INTERFACE CLASS (N2IC)	2
Configuration.....	2
Programming - Exchanging Data.....	3
Programming - Transactions.....	4
THE DAEMON.....	7
Create a Daemon Service on Windows	7
Daemon's INI File.....	8
Daemon's Control Group.....	9
Audit the Daemon.....	9
Control the Daemon.....	9



Introduction

This document describes the enhancements of Netweave support for the .NET environment. This provides a stateless, context free .NET class library, NWDSInterface Class (N2IC), for communicating with other Netweave servers. N2IC also provides Netweave transaction support, so a .NET client can start a transaction against a Netweave supported transaction service, such as Tandem TM/MT (TMF). The local server daemon initiates and maintains connections with other Netweave servers and also maintains transaction handles.

The audiences for this document are .NET developers and administrators. It is assumed the reader is familiar with .NET concepts, and architecture. It is not the intent of this document to explain these.

In the previous implementation a developer had to incorporate the transport classes into his/her component. This required the developer to learn the evolve architecture, master several inter-related classes and create and maintain several configuration files. With the phase2 architecture the developer just instantiates the N2IC like any other component. The configuration is provided in the ASP .NET web.config file and contains only one entry, the port of the local server daemon. If one is not using transactions, there is only one method to call in the object, namely writeRead. It is a much more elegant approach that uses concepts that should be completely familiar to an ASP .NET developer.

The N2IC is distributed as **N2IC.zip** and is shipped together with the daemon's Windows executable. The distribution package also contains a simple control program for managing the daemon.

General features of NetWeave are described in its standard documentation. NetWeave documentation may be downloaded from our ftp site in PDF format or as Microsoft Word documents.

- [//ftp.netweave.com/pub/doc/200/pdf/NWDoc200_PDF.zip](http://ftp.netweave.com/pub/doc/200/pdf/NWDoc200_PDF.zip)
- [//ftp.netweave.com/pub/doc/200/msword/NWDoc200_MSWord.zip](http://ftp.netweave.com/pub/doc/200/msword/NWDoc200_MSWord.zip)



NWDS Interface Class (N2IC)

Configuration

N2IC classes and the sample web.config file exist in the zip file, N2IC.zip. The web.config file is an XML file used by ASP .NET for storing configuration settings for applications. Here are the set of XML tags required for describing the N2IC configuration parameters that should be included in your web.config file:

```
<configuration>
  <configSections>
    <sectionGroup name="Netweave">
      <sectionGroup name="N2IC">
        <section name="config" type=
          "System.Configuration.NameValueSectionHandler,System"/>
      </sectionGroup>
    </sectionGroup>
  </configSections>

  <Netweave>
    <N2IC>
      <config>
        <add key="localSvrPort" value="19344"/>
      </config>
    </N2IC>
  </Netweave>
</configuration>
```

There is one parameter to configure, localSvrPort.



Programming – Exchanging Data

We are providing VB .NET and C# examples but any language that supports Microsoft's CLR (Common Language Runtime) can use the N2IC library. To send a request and get a reply from a Netweave server, call the writeRead method of N2IC.

[VB]

```
Public Function writeRead(  
    ByVal connectionName As String,  
    ByVal wData As IOData,  
    ByVal timeout As Integer)  
    As IOData
```

[C#]

```
public IOData writeRead(  
    string connectionName,  
    IOData wData,  
    int timeout);
```

sends and receives data from remote Netweave server

Parameters:

connectionName - name of server as defined in local server's ini file

wData - io data written from .Net app to Netweave server

timeout - time in milliseconds to wait for read to complete.

Returns:

IOData data returned from Netweave server

Throws:

NwdsException

The writeRead method requires a connectionName string that needs to be defined in the local server daemon ini file, and an IOData object that contains the data to be written to the server. The timeout parameter is optional. If not provided your program will block until the Netweave server responds. If

successful `writeRead` returns an `IOData` object with the response, otherwise an `NWDSEException` is thrown. Here is a programming snippet that initializes the bean and exchanges data 1000 times with a server called `ECHO_SERVER`, which just returns the same data that was sent:

[VB]

```
Dim transport As New N2IC()
Dim i As New IOData(3000)
Dim b As [Byte] () = System.Text.Encoding.ASCII.GetBytes("hello world")
i.data = b
Dim o As IOData
Dim server As New String("ECHO_SERVER")
Dim x As Integer
For x = 0 To 1000
    o = transport.writeRead(server, i);
    Console.WriteLine("Response = " + o.data.ToString());
Next x
```


[C#]

```
N2IC transport = new N2IC();
IOData i = new IOData(3000);
Byte[] b = System.Text.Encoding.ASCII.GetBytes("hello world");
i.data = b;
IOData o;
String server = "ECHO_SERVER";
for (int x=0; x < 1000; x++) {
    o = transport.writeRead(server, i);
    Console.WriteLine("Response = " + new String(o.data));
}
```

Programming – Transactions

N2IC also supports transactions to Netweave supported transaction servers such as Tandem TM/MT (TMF) and IBM CICS. Transactions are a series of writes that are sometimes referred to as a “unit of work”. Either all the writes of the unit of work are accepted or rolled back by the transaction server.

There are three additional methods in N2IC for transaction support: `startTransaction`, `commitTransaction`, and `abortTransaction`. The `startTransaction` returns a `TxHandle` object. The



TxHandle object is passed to the writeRead method for all writes that are a part of the unit of work. The transaction must then be completed with a call to commitTransaction or all writes will be rolled back with a call to abortTransaction. Here is a programming snippet of a transaction:


[VB]

```
Dim transport As New N2IC()
Dim i As New IOData(3000)
Dim b As [Byte] () = System.Text.Encoding.ASCII.GetBytes("hello world")
i.data = b
Dim server As New String("ECHO_SERVER")
Dim node As New String("TANDEM::")
TxHandle tx = transport.startTransaction(node)
transport.writeRead(server, tx, i)
transport.writeRead(server, tx, i)
transport.writeRead(server, tx, i)
transport.commitTransaction(tx)
```

[C#]

```
N2IC transport = new N2IC();
IOData i = new IOData(3000);
Byte[] b = System.Text.Encoding.ASCII.GetBytes("hello world");
i.data = b;
String server = "ECHO_SERVER";
String node = "TANDEM::";

TxHandle tx = transport.startTransaction(node);
transport.writeRead(server, tx, i);
transport.writeRead(server, tx, i);
transport.writeRead(server, tx, i);
transport.commitTransaction(tx);
```

Since the local daemon server holds the transaction handle, an inactivity timer exists on the daemon to ensure that all transactions are eventually resolved. If the transaction is not used for the timer period, the daemon will abort the transaction. The timer can be set in the ini file. Its default value is 30 minutes.

The Daemon

The key to the NetWeave Transport is the daemon, a local process that provides the services for the N2IC instrumented ASP.NET code. The daemon uses the NetWeave IPC library to make TCP/IP connections to remote servers. It pools these connections for repeated use by the N2IC-instrumented components.

The daemon accepts local connections from the .NET component. A request message names the remote server to which the daemon forwards the request. When the reply returns from the remote server, the daemon relays it back to the specific Component that made the request. The local socket connection between a .NET Component and the daemon exists for the lifetime of the component.

As the name 'daemon' suggests, this process must be running when the application server is running. The daemon is a NetWeave application that uses information in its configuration file ("INI file") to make TCP/IP connections to one or more server applications on a remote computer. General information about NetWeave INI files is in the "NetWeave Configuration Guide" and not reproduced here.

The daemon implements a minimal management interface described in section "[Control the Daemon](#)". The control interface is shown in figure 1 as a remote application, but it may be a local application.

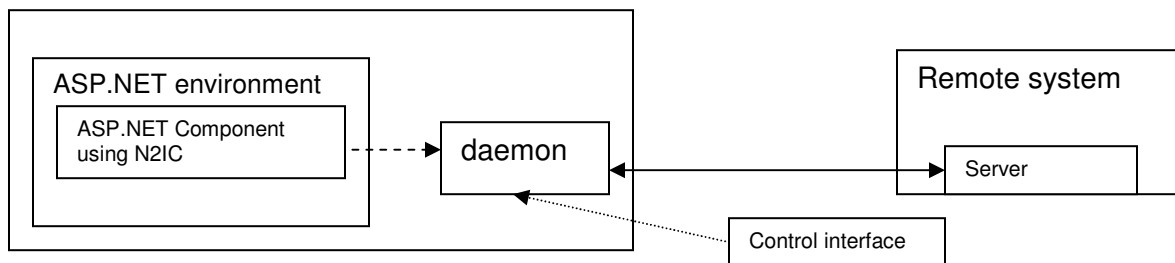


Figure 1

Figure 1 illustrates relationships among the ASP.NET environment, Component using N2IC, daemon, and the remote server application. The .NET programmer instantiates the N2IC class, connects to the daemon and sends the request. The daemon extracts the name of the remote server from the request and searches its pool for an available connection to that process. It creates a new one if the pool is empty. The daemon sends the request to the server who processes it and replies. The daemon relays the reply to the N2IC object.

Create a Daemon Service on Windows

The Windows version provides two service executables, localsrv.exe and vicelocl.exe. The localsrv executable runs from the command line interface exactly as the version described above. The vicelocl runs as a traditional Windows service. Both versions of the daemon share the same code for the core functionality. The vicelocl.exe includes an additional layer of functionality to interoperate with the Windows Service Control Manager. This "service shell" supports 5 commands to manage the daemon service.

- INSTALL – installs the service as NWDS_LCL
- UNINSTALL – uninstalls the service

- START – starts the service
- STOP – stops the service
- QUERY – tells whether the service is running or stopped.

A sixth command, `CONSOLE`, runs the service directly from the command line. It is the same as running `localsrv`.

The service shell has its own runtime syntax:

```
<service daemon> <service command> <INI file> <start group> <control group>
```

where

- `<service daemon >` is the full file name of the daemon executable. The full file name must include the path. NetWeave names the daemon “`vicelocl.exe`”, but you may rename to suit your installation.
- `<service command>` is one of the commands above.
- `<INI file>` is the full file name (including path) of the NetWeave INI file.
- `<start group>` is a combination of a typical NetWeave start group and the NetWeave IPC connection group to which the client stub object in the transport classes connect.
- `<control group>` is a connection group for the daemon’s command interface. This is described below.

Daemon’s INI File

Applications that use the former Evolve Transport classes must update the existing INI file and start group as defined in the protocol file. Change the start group to include connection information for the local connection between each client stub object and the daemon. For example, assume the current start group, `[START]`, contains only the `@TRACE_FILE@` macro.

```
[START]
@TRACE_FILE@ = c:\usr\nwds\daemon.err
```

Add connection parameters for a TCP/IP connection on localhost, i.e., IP address 127.0.0.1. This is the address at which the client expects to find the daemon. The port is selectable; in the following example it is arbitrarily set to 19344. As described below, this parameter must match the `localSvrPort` parameter in the `.NET web.config` configuration file.

Caution: unlike typical TCP/IP connections to a process on another computer, this localhost connection must specify `LOCAL_PROCESS = 1`. For example,

```
[START]
@TRACE_FILE@ = c:\usr\nwds\daemon.err
PROTOCOL = TCPIP
LOCAL_PROCESS = 1
TCPIP_ADDRESS = 127.0.0.1
TCPIP_PORT = 19344
```

The daemon has two inactivity timers for removing old connections to the Netweave servers and stale transaction handles. The default values for both are 30 minutes and can be modified in the `USER_NAME_GROUP`. For example,

```
[USER_NAME_GROUP]
# maximum time to maintain forward handles in seconds
FORWARD_EXPIRE_TIME=900
# maximum time to maintain tx handles in seconds
TX_EXPIRE_TIME=600
```

Daemon's Control Group

This connection group contains parameters similar to the start group. The `TCPIP_PORT` must be unique, and the `LOCAL_PROCESS` must be 0, the default. The `LOCAL_PROCESS = 0` instructs NetWeave to use its standard library. This example for a control group named `[ADMIN]` sets the port to 5002.

```
[ADMIN]
PROTOCOL = TCPIP
LOCAL_PROCESS = 0
TCPIP_ADDRESS = 127.0.0.1
TCPIP_PORT = 5002
```

Audit the Daemon

The daemon implements an audit logging feature that writes a message to the NetWeave trace file to identify each request and reply. Audit logging is enabled by the `AUDIT_ON` parameter in the `[USER_NAME_GROUP]`. Refer to the “Configuration Guide” for details of this special group.

```
[USER_NAME_GROUP]
AUDIT_ON = 1
```

Control the Daemon

The daemon control program (`dcp`) is an administrative tool that sends commands to the daemon.

Syntax of the control program:

```
<dcp> <INI file> <start group> <command> <daemon's control group>
```

The `dcp` implements three commands:

- S – “graceful shutdown” tells the daemon to close all connections and stop.
- X – “toggle logging” turns logging on or off. The `AUDIT_ON` parameter sets the initial state for logging.
- I – “info” reports whether logging is on or off. The report includes TCP/IP addresses of the current remote server connections.



The dcp may be run from a remote computer or on the same computer as the daemon. When the daemon and dcp run on the same computer, they should share a common configuration file. Create a unique start group for the dcp in order to specify a unique FILE_NAME for the traces. By sharing a common configuration file, there can be no mismatch on the parameters for the control group. When the dcp is run on a computer different from the daemon, the daemon control group must be identical in both INI files.