NetWeave
Transport Classes for EJBs

Installation, Configuration and User's Guide *Version 3*

Revised 10 March 2010

www.netweave.com

Table of Contents

Introduction

This document describes version 3 of Netweave support for the java j2ee enterprise java bean (ejb) specification. Versions 3 supports the ejb version 3 standard while continuing to support ejb version 2 access for clients that require the older ejb version 2 model. Both version 3 and 2 clients can run simultaneously within the ejb container.

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

There is a single bean, the NWDSBean, that provides Netweave access to other Netweave servers outside the ejb container. The developer just instantiates the NWDSBean like any other bean. It is loaded into the container like any other bean. The configuration is provided by a deployment descriptor that contains only one entry, the port of the local server daemon. The local server daemon must run on the same machine as the ejb container. It does all the heavy lifting of communicating with other Netweave servers. If one is not using transactions, there is only one method to call against the bean, writeRead. The writeRead method sends a message to the local server daemon which forwards it onto its ultimate destination and returns a reply to the caller.

NWDSBean is a stateless session bean, it can exist in the container for a long time period as determined by the container administrator. It can also service many clients. If more than one simultaneous session is required, the container will create a second bean.

The transport classes are distributed as netweave-ejb3-bean.jar and shipped together with the daemon's executable appropriate to the computer that runs the ejb container. The jar contains the Netweave classes and an ejb-jar.xml descriptor file. The jar will have to be deployed to your container using the specific container's deployment rules. The distribution package also contains a simple control program for managing the daemon.

To avoid confusion with the ejb 2 solution, note the jar's name is ejb3. The packaging has also been updated to com.netweave.ejb3.

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
- //ftp.netweave.com/pub/doc/200/msword/NWDoc200_MSWord.zip

NWDSBean

Configuration

NWDSBean classes and the supporting descriptors exist in the jar file, netweave-ejb-bean.jar. The container neutral file, ejb-jar.xml describes the bean classes and contains the configuration information. Here is the critical part of the file:

```
<enterprise-beans>
    <session>
            <display-name>EJB3 Netweave bean</display-name>
            <ejb-name>NWDS</ejb-name>
            <home>com.netweave.ejb3.NWDSHome</home>
            <remote>com.netweave.ejb3.NWDSEjbObject</remote>
            <local-home>com.netweave.ejb3.NWDSLocalHome</local-home>
            <local>com.netweave.ejb3.NWDSLocalEjbObject</local>
            <ejb-class>com.netweave.ejb3.NWDSBean</ejb-class>
            <env-entry>
                    <env-entry-name>localSvrPort</env-entry-name>
                    <env-entry-type>java.lang.Integer</env-entry-type>
                    <env-entry-value>19344</env-entry-value>
            </env-entry>
      </session>
    </enterprise-beans>
```

There is one parameter to configure, localSvrPort. The local server daemon listens on this port for connections from the NWDSBeans. In our example it is set to 19344. The corresponding daemon ini file will have to be configured to listen on the same port. Modifying the value and deploying the bean will depend on the container you use.

The home, remote, local-home, and local entries are for ejb 2 compatiblity. For a deployment that is 100% ejb 3 compliant they may be removed.

Programming – Exchanging Data

A session bean is a synchronous two-way mechanism. To send a reqeust and get a reply from a Netweave server, call the writeRead method of NWDSBean. See the javadoc for a complete description of NWDSBean.

```
public com.netweave.ejb.IOData writeRead(java.lang.String connectionName,
                                         com.netweave.ejb.IOData wData,
                                         long timeout)
                                  throws NWDSException,
                                         javax.ejb.CreateException
```

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 bean to Netweave server

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

Returns:

IOData data returned from Netweave server

Throws:

`RemoteException`

`NwdsException`

`javax.ejb.CreateException`

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 NWDSException is thrown. Here is an ejb 3 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:

```
Context initial = new InitialContext();
NWDSRemote transport = (NWDSRemote) initial.lookup("NWDS");

IOData i = new IOData(3000);
byte b[];
b = new String("hello world").getBytes();
i.data = b;
IOData o;
String server = “ECHO_SERVER”;
for (int x=0; x < 1000; x++) {
    o = transport.writeRead(server, i);
    System.out.println("Response = " + new String(o.data));
}
transport.remove();
```

Using the 2.1 client version, the bean lookup uses the 2.1 lookup calls. Check your specific container deployment for how the bean’s home interface is named.

```
Context initial = new InitialContext();
Object objref = initial.lookup("netweave/NWDS/home");
NWDSHome home =
        (NWDSHome)PortableRemoteObject.narrow(objref,
                NWDSHome.class);
NWDSEjbObject transport = home.create();
```

Programming – Transactions

NWDSBean also supports transactions to Netweave supported transaction servers such as Tandem TM/MT (TMF) and IBM CICS. This is independent of transaction support in ejb entity beans. 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 NWDSBean 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:


```
Context initial = new InitialContext();
NWDSRemote transport = (NWDSRemote) initial.lookup("NWDS");

IOData i = new IOData(3000);
byte b[] = new String("hello world").getBytes();
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);
transport.remove();
```

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 NWDS beans. The daemon uses the NetWeave IPC library to make TCP/IP connections to remote servers. It pools these connections for repeated use by the NWDS beans.

The daemon accepts local connections from the NWDS beans. 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 bean that made the request. The local socket connection between a bean and the daemon exists for the lifetime of the bean.

As the name 'daemon' suggests, this process must be running when the application server (the container for the ejbs) 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.

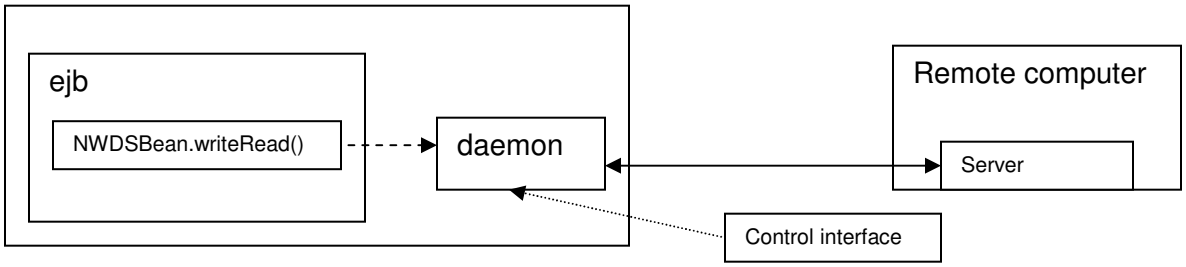

Figure 1

Figure 1 illustrates relationships among the ejb, NWDS bean, daemon, and the remote server application. The ejb programmer creates the NWDS Bean and 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 bean.

Start the Daemon on UNIX

The syntax to execute the daemon is:

```
<daemon name> <INI file> <start group> <control group>
```

where

- <daemon name> is any name you pick for the daemon executable. NetWeave names the daemon "localsrv", but you may rename to suit your installation.
- <INI file> is 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.

On Windows, Create a Daemon Service

The Windows version provides two service executables, localsrv.exe and vicelocl.exe. The localsrv executable runs from the command line interface exactly as the Unix 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@ = /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 stub 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 NWDSBean deployment descriptor.

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@ = /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.