Novell tm

DirXML® Driver 2.0 for Java* Message Service (JMS) and WebSphere* MQ

Driver Overview and Configuration

Version 2.0

Table of Contents

Preface

This document is for network administrators, consultants, and JMS administrators. DirXML® Driver 2.0 for Java* Message Service (JMS) and WebSphere* MQ is designed to share data between eDirectory™ and applications that are interfaced to the many different messaging bus applications supported by the JMS standard.

This configurable solution gives organizations the ability to increase productivity and streamline business processes by integrating any business process and information system platform in the enterprise with Directory Services.

The previous version of the driver is the DirXML Driver 1.0 for WebSphere MQ.

Documentation Conventions

The term driver refers to all components of the DirXML Driver 2.0 for Java Message Service (JMS) and WebSphere MQ and not to any one particular component.

In Novell documentation, a greater-than symbol (>) is used to separate actions within a step and items in a cross-reference path.

In this documentation, a trademark symbol (®,™, etc.) denotes a Novell trademark. An asterisk (*) denotes a third-party trademark.

Introducing the Driver

Overview

The DirXML® Driver for Java Message Service (JMS) and WebSphere MQ, subsequently referred to as the driver or JMS driver, allows the creation of automated data synchronization links between any applications supported by a JMS-compliant message bus and Novell® eDirectory™.

The JMS driver offers several key advantages for Secure Identity Management (SIM) integration, including the following:

- Native XML-based API connectivity to Enterprise Resource Planning applications from SAP*, Oracle*, J.D. Edwards* and many others. This is often the only method authorized by customers and application vendors for interfacing with this class of application, because the XML APIs can be pre-validated prior to processing. Hundreds of XML APIs have been defined collaboratively by vendors and customers through The Organization for the Advancement of Structured Information Standards (OASIS) (http://www.oasis-open.org/).

  In addition to protecting data integrity, an XML-based API is highly abstracted (that is, much less likely to change than the underlying table structure), which means that database modifications or even a wholesale application replacement do not necessarily require that the DirXML solution be modified in any way.

- Native XML-based connectivity to and from Web application servers including IBM* WebSphere, BEA WebLogic*, iPlanet* AS, Oracle AS, Novell exteNd™, and others.

- Simplified integration into IBM Mainframe (OS/390) and Midrange (AS/400) environments as well as integration with security frameworks such as IBM RACF.

- Integration with the vast majority of Enterprise Application Integration (EAI) architectures, including TIBCO Rendezvous, IBM's Websphere MQ and Websphere Business Integration (CrossWorlds), Oracle Advanced Queuing, Novell jBroker™, and many other such technologies.

The driver also provides Enterprise-class features that are needed in mission-critical situations:

- **Automated Failover**: Multiple JMS driver instances work cooperatively to ensure high availability.

- **Load Balancing**: Any number of JMS driver instances can be clustered to provide very high performance.

In short, this DirXML driver enables identity profile information managed within business processes across the enterprise to be integrated into a common directory services strategy. It works with, rather than in place of, existing customer EAI infrastructures; it allows the guaranteed delivery, exchange and processing of Identity information among applications and eDirectory; and can greatly reduce the complexity and total number of DirXML drivers needed to implement an enterprise SIM solution.

What Is JMS

The Java Message Service (JMS) API is an API for accessing enterprise messaging systems. It is part of the Java 2 Platform, Enterprise Edition (J2EE*). This makes it easy to write business applications that asynchronously send and receive critical business data and events.

JMS defines a common enterprise messaging API that is designed to be easily and efficiently supported by a wide range of enterprise messaging products. It supports both message queuing and publish-subscribe styles of messaging.

Java Message Service (JMS): http://java.sun.com/products/jms

Java 2 Platform, Enterprise Edition (J2EE): http://java.sun.com/j2ee

JMS Frequently Asked Questions: http://java.sun.com/products/jms/faq.html

Additional Information on JMS: http://java.sun.com/products/jms/docs.html

High Availability JMS

High Availability JMS Servers provide failover, high availability, and load balancing by implementing clustering. A common high availability implementation is a cluster of two JMS Server nodes, one of which is the Active node and the other a "warm" standby node. The standby node does not participate in any operation, except sync up, when the active node is up.

When the Active node fails, a standby node takes over transparently and functions as the Active node. The first node, when it resumes operation, subsequently functions as a standby node. The standby node remains aware of the status of the active node and takes over in the event of a failure.

Understanding JMS Driver Concepts

The driver is a bidirectional synchronization product that establishes a link between JMS-supported systems and eDirectory. This solution uses XML to provide data and event transformation capabilities and can convert eDirectory data and events into XML data and vice-versa.

The JMS-compliant messaging queue (MQ) middleware generally acts as a hub that applications and directories can send data to and receive data from. The driver acts as a specialized connector into this hub that exchanges data among consumer applications and foreign directories and eDirectory. This results in two main flows of data for Identity Manager: the Publisher channel and the Application Subscriber channel.

The driver supports both point-to-point and publish-subscribe messaging models. Applications can publish information in any available form to queues and topics. The Publisher channel can also consume messages from queues or topics.

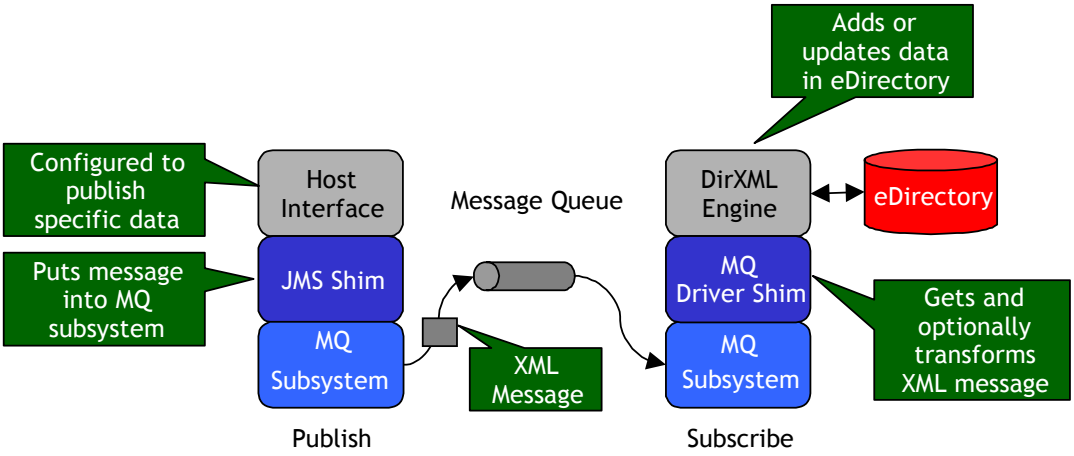
If the message queue system supports publish and subscribe transactions, multiple consuming applications can listen for and receive the same XML documents. This type of message configuration has the ability to simplify or eliminate unnecessary DirXML application development by allowing more than one application to consume documents produced by DirXML. Likewise, DirXML might tap into an existing message queue in order to consume its documents.

JMS Engine Processing

The DirXML engine processes the XML document by sequentially applying all configured rules based on the standard DirXML process flow. The driver can then manipulate the information using various rules, filters, and style sheets defined by the system administrator or developer. The driver then submits the data to eDirectory. When used with other DirXML drivers or driver instances, the data can be shared among many business applications and directories. Based on business rules, these other applications can add additional data that can, in turn, be submitted to other JMS-connected applications.

The DirXML Publisher Channel

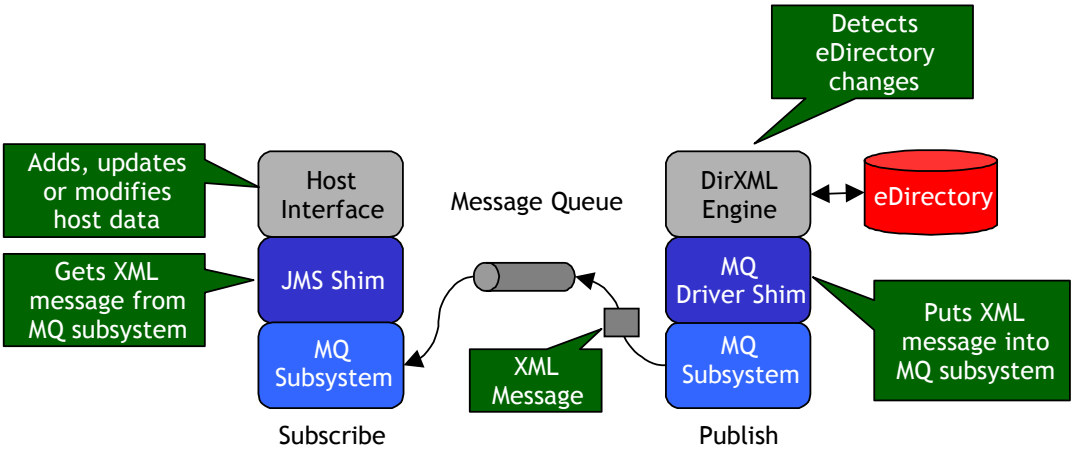
The Publisher channel is used to integrate application data with eDirectory:



The JMS shim publishes the application's output, converts it into XML document format, and places the document into a specified JMS message queue or topic. The MQ driver shim consumes the message from the specified queue or topic and submits XML-formatted changes to the DirXML engine for publication into eDirectory.

The DirXML Subscriber Channel

The Subscriber channel is used to integrate eDirectory data with host applications:



The Subscriber channel receives XML-formatted eDirectory events from the DirXML engine, converts these documents to an appropriate data format, and publishes them to a JMS queue or topic where an application can consume them.

All eDirectory events that are filtered by the DirXML driver are submitted to the JMS-interfaced application via the Subscriber channel. It is up to the application's JMS interface to interpret these messages correctly.

Publishing to eDirectory

When a JMS-connected application is determined to be an authoritative source of user profile data, that system can propagate all Add, Delete, and Modify object event data to eDirectory. The Publisher channel is used for propagation into eDirectory. For data to flow from the JMS-connected application to eDirectory, the driver utilizes the JMS interface to place XML documents into the appropriate queue for the application's interface.

JMS interfaces can be developed on a custom basis or purchased commercially as off-the-shelf connectors from third parties. In addition, many third-party Extraction, Transformation, and Loading (ETL) tools support the JMS interface standard and can readily be configured to communicate with many messaging middleware products. JMS ensures that an XML document is securely and reliably transported from the application or host system to eDirectory.

More information on using Java Message Service can be found at:

http://java.sun.com/products/jms/

Subscribing from eDirectory

The Subscriber channel of the driver is the component responsible for synchronizing data from eDirectory into a specific JMS message queue. This data can then be used to query, update, and delete data managed by the host application.

Any combination of attributes, including those containing character string and binary data, can be subscribed from eDirectory via a JMS message queue. Additionally, JMS header information can be set by the driver and read by the subscribing application. The underlying JMS transport ensures that the XML payload is securely and reliably delivered from eDirectory to the host application.

Benefits

The driver uniquely enables the automation and maintenance of identity management processes with a very wide range of Enterprise applications:

- Message-based integration is inherently abstracted, which provides for greater data integrity and insulation against change. Greater abstraction also simplifies DirXML development, including collaboration and delegated work processes.

- The existing EAI infrastructure, which is increasingly found in public and private enterprises, can be fully leveraged for integration between DirXML and Applications and Data Warehouses.

- ERP applications often provide their own EAI and JMS messaging capabilities that can be fully leveraged at no additional purchase cost to the customer. For

example, integrating with Oracle Financials or Oracle Workflow can be performed using Oracle's Advanced Queuing (AQ) interface.

- Native connectivity to IBM S/390 Mainframe or AS/400 Midrange host platforms can be provided via IBM JMS middleware.

Features

The driver provides the following features:

- Support for JMS providers
- Support for Novell Remote Loader

Java-Enabled Enterprise Application Integration (EAI) Systems

- IBM WebSphere MQ
- IBM*WebSphere MQ (Remote Connections)\
- Novell exteNd Enterprise
- OpenJMS
- Oracle* Advanced Queuing (AQ)
- SoftWired iBus Message Server*
- SpiritSoft SpiritWave TIBCO-JMS Bridge
- TIBCO JMS Server
- Sun* Message Queuing
- Sonic MQ
- SeeBeyond*
- BEA
- JBOSS*
- Support for additional JMS providers available upon request

Supported Platforms

- Linux*
- NetWare®
- Solaris*
- Windows* NT* 2000
- Windows 2003

JMS Driver Features

- Subscribe and Publish to Queues
- Subscribe and Publish to Topics
- Custom String Properties
- Message Priority
- TTL (Time To Live) JMS header.
- Query interface
- Publisher Tracking Interface (Queues and Topics)
- Publisher Auditing Interface (Queues and Topic)
- Reflexive Acknowledgement Queue
- Performance Statistics

Functional Channels operate independently (Examples: subscribe to both a queue and a topic simultaneously; publish to both a queue and topic simultaneously; publish to a queue and subscribe to a topic; publish to a topic and subscribe to a queue.)

For More Information about DirXML

For more information about DirXML, refer to the *DirXML Administration Guide (*http://www.novell.com/documentation/lg/dirxml11a/index.html) or the *Nsure Identity Manager Administration Guide* (http://www.novell.com/documentation/lg/dirxml20/index.html).

For more information about eDirectory, refer to the Novell® eDirectory documentation (http://www.novell.com/documentation/lg/edir871/index.html, or the documentation for a later version).

For more information about Java Message Service, refer to http://java.sun.com/products/jms/. For more information about specific message queue middleware, refer to your vendor's specific documentation.

JMS Configuration and Setup

You can integrate enterprise applications with the DirXML Driver for JMS to enhance and secure your organization's business processes. Before installing and configuring the driver, you must evaluate and design the processes that will underlie the integration. The design must define the driver's configuration, rules, and style sheets to automate these processes as necessary. Additionally, queues or topics must be defined for your specific messaging middleware product.

Using iManager

Novell iManager is a tool for managing eDirectory. Additional administration and management capabilities are added to iManager through plug-ins.

This Web-based management tool was introduced with eDirectory 8.7. Because it is Web-based, you can do DirXML tasks from outside the firewall. Part of the iManager interface for DirXML is a helpful graphical representation of the rules and style sheets for each instance of the driver. Prior to installing and configuring the driver, you should install the DirXML plug-ins found on the DirXML CD.

If the version of DirXML you are using supports both administration tools, you can use both ConsoleOne® and iManager to manage the same DirXML drivers. The tools are not mutually exclusive.

Product Components

The driver contains the following components:

- Driver configuration import files
- Driver shim

Driver Configuration

The driver configuration contains required driver parameters and policies. Driver configuration files are provided with the driver for installation on a system that includes DirXML, JMS, and the driver. Additional environments are also supported via manual configuration.

Driver Shim

The driver shim handles communication between the JMS message queue and the DirXML engine. It also allows for manipulation and querying of the JMS header for use of JMS-specific features by application developers and systems integrators.

Installing and Configuring the Driver

This section helps you do the following:

- Understand prerequisites for the driver
- Planning for installation
- Install driver components
- Configure driver parameters

Prerequisites

The DirXML® Driver for JMS requires the following:

- eDirectory™ 8.6.1. or higher
- Novell® DirXML® 1.1 or higher, or Identity Manager 2
- Novell iManager 1.5
- Java Virtual Machine (JVM*) 1.2 or higher
- Java Message Systems Libraries JMS.Jar

It is recommended that you create the Driver Set object before you install the driver. For more information about the Driver Set object, refer to Creating Driver Sets and Objects in the *DirXML Administration Guide* at:
http://www.novell.com/documentation/lg/dirxml11a/index.html

Enterprise Messaging Buses support the Java Message Service interface, such as WebSphere MQ, TIBCO Rendezvous, Oracle AQ, and Novell Messaging Platform. The necessary vendor-specific JMS class libraries must be obtained to allow the JMS driver to function properly. (In some cases, there might be a licensing fee for the use of a vendor's JMS classes. Be sure to verify licensing issues with your specific vendor as part of any project planning.)

iManager

Novell iManager 1.5 or later with the DirXML plug-ins installed

Supported Platforms

The driver runs on all DirXML-enabled platforms, including Windows* NT* 2000, NetWare®, Solaris*, and Linux*.

Planning for Installation

Before you install and use the driver, you must first plan for the installation.

JMS Checklist

- Establish or identify the queues to be used either to hold event messages published to eDirectory and/or event messages subscribed from eDirectory™.
- Ensure that the appropriate vendor-specific JMS classes are installed on the server hosting the driver in accordance with the vendor's recommendation.
- Vendor-specific features: Ensure that any vendor-specific supporting components have been installed and configured properly on the JMS server. Example: when using topics with IBM WebSphere MQ, ensure that the message broker (strmqbrk) process has been started.

Planning a Local or Remote Installation

This section explains the difference between a local and remote installation of the driver.

Local Installation

A local installation installs the driver on a server machine where you have JMS, DirXML® and eDirectory installed.

Remote Installation

A remote installation installs the driver on a different computer than the one where DirXML and eDirectory are installed. When using NetWare®, you might want to use this type of installation when installing the JMS driver if your JMS version is not supported to run under NetWare.

Installing Components

The driver installation program installs the following components on the server:

Component	Description
Driver Shim	A Java driver shim that communicates between the JMS message bus and the DirXML engine.
Driver Configuration XML File	All eDirectory objects, including the appropriate rules and style sheets for adding, modifying, and deleting or disabling objects. Controls the information being sent from JMS to eDirectory and from eDirectory to JMS.

Although the installation program installs the components, setup is not complete until you properly configure the Driver object and the JMS system.

Installing or Upgrading The Driver

This section Helps you use the Application Driver Creation Wizard to install and configure the driver.

Installing the Driver

1. Download jms_mq_Install.exe or jms_mq_Install.bin and run it.
2. Run install.exe, found in the product distribution.
3. Click next on the Introduction screen.
4. Accept the license agreement, then click Next.
5. Select the directory you would like to use,then click Next.
6. Click Finish to perform the installation.

Component	Filename
Driver Shim	Jms.jar (supporting files) Jmsdriver.jar
Driver Configuration File	Jmsdriver20.xml

The Install package copies the files Jms.jar, Jmsdriver.jar and jmsdriver20.xml to the default \novell\nds\lib directory.

To complete the installation, manually copy the jmsdriver20.xml file to the appropriate directory. If you are using ConsoleOne®, copy the file to novell\consoleone\1.2\snapins\dirxml. If you are using Novell iManager, copy the file to tomcat/4/webapps/nps/dirxml.drivers/

Upgrading from 1.0 to 2.0

After you download the CD image, perform the following steps to upgrade a previous version of the driver:

1. Stop the drivers that you want to upgrade. Select Manual for the driver's startup option.
2. Stop eDirectory.
3. Copy jmsdriver.jar and jms.Jar into the appropriate directory for your platform. Use the following table to determine the appropriate directory:

Platform	Directory Path
NetWare	sys:\system\lib
Solaris or Linux	/usr/lib/dirxml/classes
Windows NT/2000	novell\nds\lib

4. Restart eDirectory.

5. (Optional) Install the driver configuration.
6. Export a copy of the current driver configuration.
7. In the Drive Module, replace com.novell.nds.dirxml.driver.mq.MqDirXMLDriverShim with com.novell.nds.dirxml.driver.JMS.JMSDirXMLDriverShim
8. Make a copy of the current driver parameters.
9. Replace the driver parameters with the new driver parameters shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<driver-config name="JMS Driver">
        <driver-options>
                <qmgr display-name="Queue Manager"/>
                <msgbrk display-name="Message Broker"/>
                <mqchannel display-name="WebSphere MQ Channel"/>
                <purl display-name="Provider Url"/>
                <port display-name="Port"/>
                <vb display-name="Verbose">TRUE</vb>
                <osid display-name="OracleSID"/>
                <ouser display-name="Oracle-Schema"/>
                <otable display-name="OracleQueueTable"/>
                <clientid display-name="JMS Client ID"/>
                <vendor display-name="Vendor">Jbroker</vendor>
        </driver-options>
        <subscriber-options>
                <qsend display-name="Queue Sender">queue0</qsend>
                <tpub display-name="Topic Sender"/>
                <nonper display-name="Message persistence"/>
                <bytemessaging display-name="Byte Messaging"/>
                <streammessaging display-name="Stream Messaging"/>
                <jmsclient display-name="Client  Mode"/>
                <mqccsid display-name="WebSphereMQ CCSID"/>
                <mqencoding display-name="WebSphereMQ Encoding"/>
                <subqueryq display-name="Subscriber Query Queue"/>
                <subqueryreplyq display-name="Subscriber Query Reply Queue"/>
                <subquerytimeout display-name=" Subscriber Query Reply Timeout"/>
                <ttl display-name="JMS Time To Live"/>
                <pri display-name="JMS Priority"/>
                <marker display-name="Remove Scripting Markers"/>
        </subscriber-options>
        <publisher-options>
                <qrec display-name="Publisher Queue Receiver"/>
                <queryq display-name="Publisher Query Queue"/>
                <queryreplyq display-name="Publisher Query Reply Queue"/>
                <querytimeout display-name="Publisher Query Reply Timeout"/>
                <trec display-name="Topic Receiver"/>
                <headers display-name="Retrieve Headers"/>
                <jmsclient display-name="Client  Mode"/>
                <auditq display-name="Error Auditing Queue"/>
                <trackingq display-name="Transaction Tracking Queue"/>
                <correlationid display-name="Use Correlation ID'S"/>
                <performance display-name="Performance Queue"/>
                <ackqueue display-name="Reflexive Acknowledgement Queue"/>
                <ackheader display-name="Reflexive Acknowledgement Header"/>
        </publisher-options>
7</driver-config>
```

10. Set the Client Mode to MQ if you need to have native mode JMS Messages.
11. Set the driver's startup options to their previous values.
12. Restart the drivers.

Activating DirXML Products

Activation must be completed within 90 days of installation, or the driver will not run.

****NOTE:** *Activating a driver does not change your current configuration or install a newer version of the driver shim. It simply changes the driver to an activated state*.

The following examples describe various activation scenarios you might encounter:

- You purchase a DirXML bundled activation. This includes the activation of multiple drivers and the DirXML engine.
- You purchase individual driver activation for DirXML. This includes the activation of a single driver and the DirXML engine.
- You purchase driver group activation for Identity Manager.
- You purchase customized or third party driver activation. This includes the activation of a customized or third-party driver and the DirXML engine.

For information about activating with DirXML, refer to Activating DirXML Products (http://www.novell.com/documentation/lg/dirxml11a/dirxml/data/agoppxb.html) and Viewing Product Activations for DirXML and DirXML Drivers (http://www.novell.com/documentation/lg/dirxml11a/dirxml/data/agoppxb.html#agfhtax).

For information about activating with Identity Manager, refer to Activating Novell Identity Manager Products (http://www.novell.com/documentation/lg/dirxml20/admin/data/afbx4oc.html).

For additional information about Activation, refer to Activation Basics (http://www.novell.com/partners/partnerplace/epd/product_activation_basics.html) and Activation Troubleshooting (http://www.novell.com/partners/partnerplace/epd/troubleshooting_activation.html).

To purchase DirXML licenses, see the information on How to Buy (http://www.novell.com/products/edirectory/dirxml/howtobuy.html).

Post- Installation Tasks

Now that you have installed the driver, you must do the following:

- Import the Driver Configuration file.
- Configure the Remote Loader on the remote system.

Importing the Driver Configuration

The Application Driver Creation Wizard helps you import a Driver Configuration file. This file creates and configures objects needed in eDirectory to make the driver work properly.

- Right-click Network Neighborhood, then click NetWare Connections.

  Ensure the following:

  - You are authenticated to the eDirectory server you are installing to.
  - The server where the driver will be running is set as the primary server.

1. In IManager, click Wizards > Create a New Application Driver.
2. Select the driver set where you want the driver installed, then click Next.
3. Enter the following driver set properties as prompted:

   - Name
   - Context
   - Whether to create a new partition on the driver set

4. Select Jmsdriver.xml driver configuration file, then click Next.

   Follow the on-screen prompts to enter information about eDirectory structure and driver connectivity.

5. Click Next, then enter the following information as prompted:

   - The name you want to use for the driver

   - The driver password

   The shim authentication password is used for logging into JMS. The driver password is used to authenticate to the DirXML server.

- When the driver import is finished, click Yes to define security equivalences on the driver.

- Click Add, then select an object with Admin rights (or any other rights you want the driver to have).

- Click Apply, then click Close.

6. Exclude Administrative Roles from replication, click Apply, then click Close.

7. Click Finish.

Configuring the Remote Loader on the Remote System

The optional Remote Loader can be configured to use an SSL connection for secure data transfer between eDirectory and the JMS system. This section explains what needs to be completed to establish an SSL connection.

1. Generate a security certificate on the eDirectory container.
2. Create an organizational trusted root certificate.
3. Validate the certificate file.
4. Run Remote Loader Wizard and configure Remote Loader options.
5. Configure Driver Object properties in iManager.

Generating a Security Certificate on the eDirectory Container

In iManager, select the Container in which the eDirectory server resides.

1. Right-click the container, select New Object, then create the NDSPKI: Key Material object.

2. Enter a name for the certificate, select the creation method, then click Next.

   IMPORTANT: Write down the name of the certificate. You will need this information when configuring the driver.

3. Select the Standard Creation Method, then click Next to view the Confirmation Screen.

4. Click Finish to generate a certificate for the eDirectory container.

Creating an Organizational Trusted Root Certificate

In IManager, browse to the Security container and locate the Organizational CA for the tree.

1. Right-click the certificate object, then click Properties.

2. Click the Certificate tab and go to the Self-Signed Certificate option.

3. Select Export, then click Next.

4. Select the Base64 option, then click Next.

5. Click Finish.

6. Click Validate to verify the Certificate’s validity.

7. Copy the certificate file to the Remote Loader directory on the Remote Loader host system.

Using the Wizard to Configure Remote Loader Properties on the Remote Loader Host Server

From a DOS command prompt on the Remote Loader host system, change directory to the Remote Loader directory and enter: dirxml_remote

1. Click Next to begin the wizard.

- Following the wizard prompts, configure the following items:

  - Command Port. Used to differentiate between instances of Remote Loader.

  - Increments. The first instance of Remote Loader defaults to port 8000 with each additional instance incrementing in number by one.

  - Configuration File. The configuration file is a filename that is automatically generated. Verify the filename.

  DirXML Driver. Select the Java option. Enter the case-sensitive java class name: com.novell.nds.dirxml.driver.JMS.JMSDirXMLDriverShim.

  - Connection to DirXML. Enter the port number. The default is 8090, but you can enter a different, unique port number. Select the IP address to which you want to apply changes. You should select the All IP addresses option.

  - Certificate. Select Use SSL. Browse to the certificate generated in the previous section, then click Next. The browser automatically looks for Base64 files.

  - Tracing. Choose a level of tracing. In addition to trace files, you can log trace events to a log file. Configure this log file if you want. Trace level 0 provides no trace application window.

  - Install as a service. When you install Remote Loader as a service, it automatically launches when the machine boots.

  - Passwords. Enter a password for the Remote Loader and Driver object.
    **Note: IMPORTANT please write down these passwords. You need them for the next step in driver configuration.

2. Summary. Click Next to continue with the DirXML Remote Loader configuration.

3. Click Finish to complete the configuration. When prompted to start the DirXML Remote Loader, click Yes.

1. Open the Windows Service Control Manager.

2. Select the DirXML Loader (com.novell.nds.dirxmldriver), click the Startup Option, then click Start.

Starting the Remote Loader Manually

1. From a command prompt (wherever Remote Loader is installed), enter:

   -dir *.txt

   This displays the name of the configuration file (for example, config8001.txt).

2. Enter the following:

   dirxml_remote config <configfilename.txt>

   For example:

   dirxml_remote -config config8001.txt

   You can open the Windows Task Manager to verify that Remote Loader is running.

   **Note: If the Java Message Service factory classes have been installed into eDirectory, the driver retrieves these classes from the directory prior to initialization. If the remote driver exists in a DMZ or is separated by a firewall, the default TCP ports 8090 for the remote loader, 389 for LDAP, and optionally 636 for LDAP/S must be open to the server that hosts the driver.

Configuring Driver Object Properties for the Remote Loader

1. In iManager, select the JMS Driver object, then click Properties.

2. Click the Authentication tab.

- Enter the following Remote Loader connection parameters with a space between each parameter:

    - Hostname. Specifies the address or name of the machine on which the Remote Loader will run. For example, hostname=192.168.0.1

    - Port. Specifies the port on which the Remote Loader will accept connections from the remote interface shim. For example, port=8090

    - KMO. Specifies the key name of the Key Material Object containing the keys and certificate used for SSL. For example: kmo=remotedrivercert.

      **NOTE: If SSL is enabled, you must enter the hostname, port, and KMO information. If SSL is not enabled, you only need to enter the hostname and port information.

3. Enter passwords for the application and the Remote Loader.

4. Click the Driver Parameters tab, then scroll to Publisher Settings.

5. Verify that the Publisher queue is configured. This queue contains XML document to be processed against eDirectory. Refer to Configuring Driver Object Properties for more information.

Configuring The JMS Driver Parameters

This section helps you to configure the JMS Driver properties. In addition to the following driver-specific configuration parameters, you should configure basic DirXML Driver object fields. For more information regarding these basic configuration fields, see the *DirXML Administration Guide*.

Configuring Driver Parameters

In iManager, select the JMS Driver object, then click Properties and configure the following:

Tabs/Parameters	Description
Driver Module	
JMS	The type and name of the DirXML driver: com.novell.nds.dirxml.driver.JMS.JMSDirXMLDriverShim
Authentication	
Authentication ID	The user name to authenticate against the JMS provider installation.
Authentication Context	An optional directory context of the authentication object.
Application Password	The password used in conjunction with the Authentication ID.
Driver Settings	
Queue Manager	The name of a queue manager.
Message Broker	The name of message broker. For non-JMS implementations, this field must be filled in with the element Message Broker if you want to send or receive using topics.
Verbose	Enables verbose logging to the DirXML trace to enhance debugging. The only valid value is True. Otherwise,leave it empty.
Port	This element is used only for connecting to JMS Server Installation on non standard/default ports.
Oracle SID	Used only when connecting to Oracle Advanced Queuing. The element represents the Oracle SID Identifier.
Oracle-Schema	Used only when connecting to Oracle Advanced Queuing. This element is used to specify the schema tag the queues reside under.
OracleQueueTable	Used only when connecting to Oracle Advanced Queuing. This element represents the actual queue table name that will hold the JMS messages.
JMS Client ID	The JMS Client identifier used to connect to a queue or topic. If left blank, the shim generates a random JMS Client ID.
Vendor	The JMS Vendor Implementation. Valid values are MQSeries,

	Oracle, Tibco, TIBCOJMS, Jbroker, OpenJMS, and IBUS.
Subscriber Settings	
Queue Sender	The name of the queue to send DirXML messages.
Query Queue	The queue used to query the remote application via JMS.
Query Reply Queue	The queue used to receive the reply message from the remote application.
Query Reply Queue Timeout Period	The number of seconds the Subscriber channel waits for the query reply message from the JMS application. This value has no effect when enabled with pub/sub messaging. The only valid value is an integer value for seconds. Otherwise, leave it empty for the default value of 10 seconds to be used.
Topic Sender	The topic to publish NDS event information to.
Client Mode	The mode or style to use in placing messages on a queue or topic. The valid styles are WebSphere MQ native message or JMS Messages. The only valid values are blank or MQ.
Byte Messaging	Used to send a message containing a stream of uninterrupted bytes. The receiver of the message is responsible for the interpretation of the bytes. This should only be used if DirXML needs to exchange non-text messages with your application. Input True for sending a byte stream, or leave blank for the default (text) option.
JMS Time To Live	The maximum lifetime for a message. The input is expected to be in milliseconds. If the element is blank, the message has an unlimited lifetime.
JMS Priority	The priority of the message. Valid values are from 0 through 9. If the element is left blank, the message priority defaults to 4.
Message Persistence	Whether messages will be non-persistent. The default mode is persistent. The only valid value is True. If the value of True is set, the message is non-persistent
Publisher Settings	
Topic Receiver	A JMS queue to receive XML formatted DirXML commands.
Queue Receiver	A JMS queue to receive XML formatted DirXML commands from.
Query Queue	The queue to be used to query the remote application via JMS.
Query Reply Queue	The queue to be used receive the reply message from the remote application.
Query Reply Queue Timeout Period	The number of seconds the Subscriber channel waits for the query reply message from the JMS application. This value has no effect when enabled with pub/sub messaging. The only valid value is an integer value for seconds. Otherwise, leave it

	empty for the default value of 10 seconds to be used.
Audit Queue	A JMS queue to use if the submitted DirXML transaction returned an error.
Tracking Queue	A JMS queue in which to place a copy of the message that was successfully processed by the DirXML engine.
Use Correlation IDs	Whether to append a correlation identifier to both the tracking and audit queue Messages.
Performance Queue	A JMS queue in which to place the Performance XML document.
Reflexive Acknowledgement Queue	The JMS queue in which to place the reflexive acknowledgement message.
Reflexive Acknowledgement Header	The header to retrieve from the originally submitted message to place in the CorrelationID property of the reflexive acknowledgement message.

Troubleshooting the Driver

This section contains potential problems and error codes you might encounter while configuring or using the driver.

Using the DSTrace Utility

You can troubleshoot the driver by using the DSTrace utility. You can configure the utility's options by selecting Edit > Properties > DirXML Drivers.

For each event or operation received, the driver returns an XML document containing a status report. If the operation or event is not successful, the status report also contains a reason, and a text message describing the error condition. If the result is fatal, the driver shuts down.

Common Errors

Driver Load Errors	Solution
java.lang.ClassNotFoundException: com.novell.nds.dirxml.driver.JMS.JMSDirXMLDriverShim	This is a fatal error that occurs when JMSdirXMLDriverShim.jar or JMS.Jar is not installed properly. You should ensure that the file is in the proper location for either a local or remote loader configuration.
java.lang.ClassNotFoundException: com.novell.nds.dirxml.driver.JMS.JMSDirXMLDriverShim	This is a fatal error that occurs when the class name for the JMSDirXMLDriverShim.jar is incorrect. You should ensure that the Java class name is set on the Driver Module tab in a local installation and that the -class parameter is set in a Remote Loader configuration. The proper class name is: com.novell.nds.dirxml.driver.JMS.JMSDirXMLDriverShim
-295	Almost all-295 errors indicate a missing supporting java class library.
Driver Initialization Errors	**Solution**
java.lang.ArrayIndexOutOfBoundsException # > #	There is a problem with the JDK* version when trying to retrieve JMS QueueConnectionFactory object via JNDI. This does not apply to a local JMS connection. Replace the <Novell_Install_Path>\NDS\jre\lib\rt.jar with a more JDK version 1.3.1 and above rt.jar.
'no mqjbnd02 in java.library.path'	Ensure that the <MQSeries_Install_Path>\java\lib is in your System path.
MQJMS2008: failed to open MQ queue. MQJMS2005: failed to create MQQueueManager for <>	Ensure the specified queue in the 'Driver Parameters Xml' exists on the WebSphere MQ server. Ensure that the WebSphere MQ is running.
MQJMS2006: MQ problem: com.ibm.mq.MQException: Completion Code 2, Reason 2085	Regarding MQ JMS Pub/Sub calls, ensure the mq jms system queues are created. Example: C:\<MQSeries_Install_Path>\java\bin\runmqsc <queue_manager> < MQJMS_PSQ.mqsc
-295 errors	Almost all-295 errors indicate a missing supporting Java class library. Exceptions: A –295 Error with the Driver issuing a System.Exit(-1) indicates that the Novell MP Mporb12.dll is not in the system path.
-295 javax/transaction/xa/XAException	The JVM 1.4 requires the use of the JTA classes available from java.sun.com
mqjms1046: The character set 437 is not supported.	Set the sending client ccsid to 819 (the default).

Developing Solutions with the DirXML Driver for JMS

JMS-enabled applications that need to exchange data with the driver must use messages created in XML format. The driver passes XML messages to the DirXML engine and returns the results in XML.

The desired XML message format should comply with the Novell nds.dtd document type definition. If your JMS application is unable to work with the XML format specified within the nds.dtd, it can use any style of XML. However, DirXML transformation style sheets must be created to allow the driver to do the conversion of data from one format to another.

****Note for JMS Users**: The driver for JMS can work with either MQ Series style messages or regular JMS style messages. However, each instance of the driver for JMS can work with only one type of message.

Using JMS Features

In addition to the configurable parameters, the driver supports the use of many additional Java Message Service features that enhance the developer's ability to create solutions.

Message Properties

A JMS Message contains a built-in facility for supporting application-defined property values. This provides a mechanism for adding application-specific header fields to a message. Properties allow an application, via message selectors, to have a JMS provider select/filter messages on its behalf by using application-specific criteria.

Custom Message properties can be set through a DirXML Style sheet. This method is needed to pass name-value pairs to systems that are JMS header dependent.

The following example illustrates how to set a customer header of Last Name to Smith:

{LastName|Smith}

The JMS Shim looks for name-value pairs contained by brackets { } and separated by a '|'/

{FieldName|FieldValue}

Setting Binary Values in a Message

The JMS Driver 2.0 supports setting binary content in Subscriber channel messages. Many legacy application use nonprintable ASCII characters as record delimiters. The current XSLT specification does not contain a mechanism for handling non-printable characters natively.

Custom bnary characters can be set by using specific markers that the JMS Driver knows how to interpret. This method is needed to pass binary value pairs to systems that depend on nonprintable characters for their record delimiters.

The following example illustrates how to set a bnary vlue of 255 or FF.

{#CC|255}

The JMS Shim looks for name-value pairs contained by brackets { } and separated by a '|'/

{FieldName|FieldValue}

In this case, the JMS Shim knows that the specific FieldName of #cc indicates it will insert a character of (FieldValue) in the message. The FieldValue Is expressed in integer format . Therefore, if you wanted to insert a character of FF equivalent, you would express this as {#CC|255}

Message Priority

JMS defines a 10 level priority value with 0 as the lowest and 9 as the highest. JMS clients or DirXML style sheet logic should consider 0-4 as lower–to-normal priority and 5-9 as normal-to-higher priority. Priority is set to 4, by default.

TTL (Time To Live)

JMS defines the length of time in milliseconds that a message should be retained by the message system. Time to Live is set to zero by default.

JMSMessageID

A JMS Message ID property contains an optional value that uniquely identifies each message sent by a provider. This can be used to historically identify each message or in concert with Correlation ID to establish a relationship among requests and replies in an asynchronous messaging environment.

Correlation ID.

The JMS CorrelationID property is used for linking one message with another. It typically links the Message ID of the reply message with the Correlation ID of the requesting message.

Associations

Associations are made between messages and eDirectory objects through a style sheet. A unique ID can be created for records relating to each system connected via JMS.

The association attribute received from a queue is unique to the interfaced application, based on each driver instance that you install and enable.

If other drivers are installed, they should use an association specific to that application.

The association attribute is multi-valued. Therefore, if DirXML is being used to connect multiple applications, all of their associations can be stored on this attribute.

The unique ID association links objects in the JMS-connected application to their objects in eDirectory. When an ADD event occurs, the association is made and referenced in the style sheets. This association allows the driver to perform subsequent tasks on the appropriate object.

If a User object is added in eDirectory, the unique user's ID might be populated so that the ID can be linked to the individual's record in the JMS-connected application. The matching rule is executed and, if the ID is found within the application, the association is then created. The Matching rule matches the eDirectory user ID and the appropriate unique ID on the host system.

Setting the Global Association Attribute Value

A global association attribute value can be set by modifying a value in the command transformation style sheet.

The association value can be statically set by modifying the association state element, or dynamically by computing this value based on additional logic that you can add to the style sheet to meet an application's requirements.

Overriding the Global Association Attribute Value

The global association value that you establish in the style sheet can be dynamically overridden within a Publisher channel message. This is a powerful feature that is made possible by the open-ended nature of the driver's design.

To override the global value, you must include an association attribute value inside an XDS message submitted to the Publisher channel as follows:

<association> value <association/>

The Role of the Subscriber Channel in Setting Associations

In the JMS driver, the Subscriber channel plays an important role is setting the association values. At first, this might seem counter-intuitive, but it takes advantage of DirXML engine features and is very efficient.

Because of the nature of marrying DirXML to the flexible and asynchronous JMS interface, the driver automatically creates a dummy association for each Add event. This association value is then updated according to application needs, either by setting a global, static value in the spreadsheet or dynamically within a Publisher channel message (as described above).

When setting the association attribute value using the style sheet, a filter for the Subscriber channel must be enabled. This allows the relevant XSL script to fire in order to correctly update the association values.

Using The Advanced Features

Special Subscriber Channel Commands

The special Subscriber channel commands were implemented to provide the capability to output non-XDS formatted documents to a JMS queue or topic. The Identity Manager engine arbitrarily wraps the outgoing document with the standard <NDS> tags as show in the example below. The engine also adds an arbitrary event-id=0 data value as an attribute on the <Input> node.

Example:

<NDS><Input>submitted message</Input></NDS>

{RemoveInputTags|True}

The ***RemoveInputTags*** command instructs the JMS driver shim to remove the <nds><input> tags from the beginning of the message and remove the </input></nds> tags from the end of the message.

{RemoveEventID|True}

The ***RemoveEventID*** command instructs the JMS driver shim to remove the Event-ID Attribute data value from the current message.

Performance Data

The JMS driver captures and reports performance data in several different styles. The performance data can be requested on demand by simply submitting a special document to the publisher shim. The performance data is also set on a per transaction basis both in the publisher and subscriber channels.

On Demand Performance Data

The On Demand performance data is returned on demand by submitting the "<Performance Metrics/>" document to the Publisher channel queue or topic. The performance data returned on the "Performance Queue" is defined in the Publisher channel driver properties.

The specifics of the On Demand Performance data are itemized and explained in the table below.

Performance Tag	**Explanation of Data Elements**
Driver Start Time	Contains the system time when the driver is started. It is reinitialized each time you start and stop the driver. Reported in milliseconds.
Driver Up Time	Contains the total time the driver has been running continuously since the last start. Reported in milliseconds.
Current Time	Contains the current system time when the performance metrics were requested. Reported in milliseconds.
Total Number of Messages	Contains the counter indicating the total number of messages processed by the Publisher channel queue or topic.
Total Success Messages	Contains the counter indicating the total number of message successfully processed by the Publisher channel queue or topic. A successful message is defined by the DirXML engine returning a code indicating that no error, warning, or retry codes were returned during the processing of a submitted document to the Publisher channel.

Total Error Messages	Contains the counter indicating the total number of messages where a code other than success was returned by the DirXML engine during the processing of a submitted document to the Publisher channel.
Average Message Processing Time	Contains the average time to process a message in the Publisher channel. The data element is calculated by dividing the Total Message Processing Time by the Total Number of Message submitted.

Per Transaction Performance Data

Per Transaction performance data is calculated in both the Publisher and Subscriber channel and stored on the header of each individual message. The metrics stored on the individual message are discussed in the table below.

- **Publisher**

  The Publisher channel transaction performance data is calculated and then placed on the audit or transaction messages.

- **Subscriber**

  The Subscriber channel per transaction performance data is calculated and stored in the headers of each outgoing message.

Transaction Data	**Explanation of Data Elements**
Processing Start Time	Contains the system time when the driver received the message from the queue or topic.
Processing End Time	Contains the system time when the driver has finished processing the received message. Reported in milliseconds.
Total Processing Time	Contains total processing time for a received message. The Total Processing time is calculated by subtracting the processing start time from the processing end time. Reported in milliseconds.

Association Style Sheets

Overview

There are 3 primary ways to set an association value with the JMS driver:

- Subscriber Channel [Outgoing Document is in XDS format]
- Subscriber Channel [Outgoing Document is in non-XDS format]
- Publisher Channel

Association StyleSheet (XDS Format)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
	<xsl:template match="add">
		<xsl:copy>
			<xsl:apply-templates select="@*"/>
			<association state="associated">MQAssociation</association>
			<xsl:apply-templates select="node()"/>
		</xsl:copy>
	</xsl:template>
	<xsl:template match="node()|@*">
		<xsl:copy>
			<xsl:apply-templates select="node()|@*"/>
		</xsl:copy>
	</xsl:template>
</xsl:stylesheet>
```

Association Write Back (Non-XDS Format)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:cmd="http://www.novell.com/nxsl/java/com.novell.nds.dirxml.driver.XdsCommandProcessor"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
	<xsl:strip-space elements="*"/>
	<xsl:preserve-space elements="value component"/>
	<xsl:output indent="yes" method="xml"/>
	<!-- ****************** -->
	<!-- DirXML passed parameter for cmd processing -->
	<!-- ****************** -->
	<xsl:param name="srcCommandProcessor"/>
	<!-- ****************** -->
	<!-- Global Variables. -->
	<!-- ****************** -->
	<xsl:variable name="debug">false</xsl:variable>
	<xsl:variable name="message">true</xsl:variable>
	<!-- ****************** -->
	<!-- ****************** -->
	<xsl:template match="add">
		<!-- convert modify or sync with an association to an instance so that -->
		<!-- output transform can create a complete output record -->
		<xsl:variable name="associationValue" select="string(association/text())"/>
		<xsl:choose>
			<xsl:when test="association[@state = 'disabled']">
```

```
                                <!-- ignore if the association is disabled -->
                        </xsl:when>
                        <xsl:otherwise>
                <!-- if a modify on an associated object the association replace it with the instance -->
                                <!-- returned by querying the object -->
                                <xsl:variable name="assoc-cmd">
                                <add-association dest-dn="{@src-dn}" dest-entry-id="{@src-entry-id}">
                                        <xsl:value-of select="JMSAssociation"/>
                                </add-association>
                                </xsl:variable>
                                <!-- query NDS variable can be called anything -->
<xsl:variable name="do-cmd" select="cmd:execute($srcCommandProcessor, $assoc-cmd)"
xmlns:cmd="http://www.novell.com/nxsl/java/com.novell.nds.dirxml.driver.XdsCommandProcessor"/
>
                        </xsl:otherwise>
                </xsl:choose>
        </xsl:template>
</xsl:stylesheet>
```

Association (Publisher Channel)

To set the association via the Publisher channel, add an association tag into the XDS document, as shown below:

```
<association state="associated">MQAssociation</association>
```

Channel Write-Back

The JMS driver has support for the Channel Write Back on both the Publisher and Subscriber channels.

When using the Channel Write Back to send messages to the JMS provider, you must tell the command processor the name of the queue. You must specify the name of the queue by placing the following string '{cmdQueue|Queue Name}' at the beginning of the variable. The command processor removes the special command tags when processing the message.

The command processor invocation automatically encapsulates the message with the '<nds><input> </input></nds>' tags. You can instruct the command processor to remove the tags by including the following command in the variable definition '{RemoveInputTags|True}'. A more in-depth discussion about how to use the Channel write back functionality is included below.

XSLT Channel Write-Back In Depth

Now that you've seen a simple example of XSLT Channel Write-Back, let's dig in deeper and understand how it all works. There are four items that Channel Write-Back uses in any style-sheet:

- Name Space Declarations
- Parameter Declarations
- XDS Fragment Document
- Channel Write-Back Execution

Name Space Declaration

The first item you need to add is a name space declaration that instructs the NovellXSLT processor to bind the prefix 'cmd' to a Java class that implements the Channel Write-Back functionality.

xmlns:cmd="http://www.novell.com/nxsl/java/com.novell.nds.dirxml.driver.XdsCommandProcessor"

Typically, the <*xsl:stylesheet*> element's start tag looks something like this:
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
To enable Channel Write-Back in a style sheet, the <*xsl:stylesheet*> element's start tag should look like this:

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:cmd="http://www.novell.com/nxsl/java/com.novell.nds.dirxml.driver.XdsCommandProcessor" >

In terms of DirXML, the <*xsl:stylesheet*> and <*xsl:transform*> are identical and can be used interchangeably.

Parameter Declaration

In addition to adding the name space declaration, two style sheet parameters must be declared:

- SrcCommandProcessor
- destCommandyProcessor

These two parameters are passed by the DirXML engine to the styles sheet. They are used to allow the style sheet to specify which datastore should be written back to (modified). The terms "source" and "destination" are used in context of the origin of the event. In other words, if you need to query the datastore that generated the event, you send the XDS Fragment query to the srcCommand- Processor. If you need to modify the datastore that receives the event, you send the XDS Fragment to the destCommandProcessor.

Here's a hint that might help you. Because the Publisher Channel is always used for events coming into eDirectory, the destination datastore is always eDirectory. Therefore, the destCommandProcessor is used to modify eDirectory on the Publisher channel. Conversely, because the Subscriber channel is always used for events coming from eDirectory, the source datastore is always eDirectory. Therefore, the srcCommandProcessor is used to modify eDirectory.

These two parameters should be declared towards the top of your style sheet, after the <*xsl:stylesheet*> element. Although it is only necessary to declare the parameters that are actually used, it's good practice to declare them both:

```
<xsl:param name="srcCommandProcessor"/>
<xsl:param name="destCommandProcessor"/>
```

The XDS Fragment Document

Now that you've properly prepared your style sheet for Channel Write-Back, you can now delve into the actual Channel Write-Back itself. The structure of the XDS Fragment document is the same as any XDS document. This means it must comply with the NDS.DTD. The full details of the XDS Document structure can be found in the NDS.DTD file located at

http://developer.novell.com/ndk/doc/dirxml/dirxmlbk/api/ndsdtd/query.html.

After the XDS Fragment Document is constructed, it is typically stored in an <*xsl:variable*> so that it can be passed to the Command Processor for execution.

Sample XDS Fragment Documents

Below are some sample XDS Fragment Documents to better illustrate how to properly construct such a document.

Sample 1. This is a document to modify a user's Description attribute.

```
<modify class-name="User" dest-dn="\AMCE\Users\Sam">
<association>JMS Assocation<association>
<modify-attr attr-name="Description">
<remove-all-values/>
<add-value>
<value>This is a new Description</value>
</add-value>
</modify-attr>
</modify>
```

Sample 2. This document sends a message to a queue named "TestQueue". The content of the variable does not need to be in XML format. The contents can be in any format needed or desired.

```
<xsl:variable name="TextMessage">
 {cmdQueue| TestQueue }{RemoveInputTags|True}
   <TextMessage>
              <Test text message/>
  </ TextMessage >
</xsl:variable>
```

Sample 3. This is a document to add an Alias to an object.

```
<add class-name="Alias" dest-dn="\ACME\Aliases\Sam">
<add-attr attr-name="Aliased Object Name">
<value type="dn">
<xsl:value-of select="\ACME\Users\Sam"/>
</value>
</add-attr>
</add>
```

Sample 4. Examples of a non-XML formatted message

```
<xsl:variable name="TextMessage">
 {cmdQueue| TestQueue }{RemoveInputTags|True}
   This is a test Text Message
</xsl:variable>
```

The Channel Write-Back Execution

After you've properly constructed your XDS Fragment Document and stored it in an <*xsl:variable*> called *$cmd-Update*, you need to submit it to the Command Processor for execution. There are a few components to the execution:

- The name of the variable to contain the results. (Unlike queries, the result variable is never used after execution. It is used to execute the command.)

- The *cmd:* prefix to indicate that this command is to be handled by the Command Processor as declared in the Name Space Declaration.
- A reference to either the *srcCommandProcessor* or the *destCommand- Processor* to instruct the DirXML engine as to which datastore the XDS Fragment should be applied.
- A reference to the variable in which the XDS Fragment Document is stored.

When you put these elements together, you get the following command:
<xsl:variable name="result" select="cmd:execute($srcCommandProcessor, $cmd-Update)"/>
$result is the <*xsl:variable*> to store the Channel Write-Back results in. This variable has no function after the command is executed.

Query interface

Overview

The JMS driver has two query interfaces: one on the Subscriber channel, and one on the Publisher channel.

Each query interface has three components: Query Queue, Query Reply Queue, and a Query Timeout value. The query interface uses a synchronous transaction mode.

The query reply queue interface operates in blocking (synchronous) mode. When the query interface is invoked, it places a query message on the Query Queue and then waits for a reply to this message on the designated Query Reply Queue until the value of the timeout has been exceeded.

Query Queue

The Query Queue defines the queue that query documents are to be placed on. Typically, these documents are in XML or XDS format, but they can be in any format. The remote JMS client is then responsible for interpreting this query document and placing the response to the query on the Query Reply Queue.

Query Reply Queue

The Query Reply Queue interface defines the actual queue used for the query reply documents that are sent by the remote JMS client. This interface is used instruct the JMS driver where to look for the response document to the original query document.

Query Timeout Value

The Query Timeout specifies the maximum length of time the Query Reply Queue interface will wait to receive a reply from the remote JMS client responsible for responding to query messages. No other DirXML processing or blocking occurs during the specified period . If this field is left blank, it defaults to 10 milliseconds.

Sample Query Style Sheet

```
<xsl:template name="query-object-name">
<xsl:param name="object-name"/>
 <!-- build an xds query as a result tree fragment -->
 <xsl:variable name="query">
        <nds dtdversion="1.0" ndsversion="8.5">
        <input>
                <query>
        <search-class class-name="{ancestor-or-self::add/@class-name}"/>
        <!-- NOTE: depends on CN being the naming attribute -->
        <search-attr attr-name="CN">
                <value>
                        <xsl:value-of select="$object-name"/>
                </value>
        </search-attr>
                <!-- put an empty read attribute so only object is returned. -->
        <read-attr/>
        </query>
        </input>
        </nds>
 </xsl:variable>

<!-- query eDirectory -->

 <xsl:variable name="result" select="query:query($destQueryProcessor,$query)"/>
 <!-- return an empty or non-empty result tree fragment depending on result of query
-->
        <xsl:value-of select="$result//instance/@class-name"/>
 </xsl:template>
```

Sample Query Reply Documents

Included below are sample documents representing the default format for the returned query documents. However, a returned query document can be in any desired format.

One Instance Found

If a single object is found based on your search criteria, the following document will be returned:

```
<instance class-name="User" event-id="0" src-dn="\TEST_TREE\TEST\Users\JDoe">
<association state="associated">JDoe@test.com</association>
    <attr attr-name="Surname">
         <value timestamp="1016241258#16" type="string">Doe</value>
     </attr>
</instance>
```

Because only a single object matched your search criteria, there is only a single <*instance*> element in the document above.

Multiple Instances Found

This example shows you the default format for a response to query command when the search returns multiple results. Remember that this is the default format and you can return the document in any form necessary.

```
<instance class-name="User" event-id="0" src-dn="\TEST_TREE\TEST\Users\JDoe">
<association state="associated">JDoe@test.com</association>
    <attr attr-name="Surname">
        <value type="string">Doe</value>
    </attr>
</instance>
<instance class-name="User" event-id="0" src-dn="\TEST_TREE\TEST\Users\JDoe2">
<association state="associated">Jdoe2@test.com</association>
    <attr attr-name="Surname">
        <value type="string">Doe</value>
    </attr>
</instance>
```

Audit Features

The JMS driver provides a mechanism that helps to ensure best practices for handling transactions submitted to the Publisher channel. The features are:

- Audit Queue Interface
- Audit Topic Interface
- Tracking Queue Interface
- Tracking Topic Interface

Audit Queue Interface

When documents are submitted to the Publisher channel but fail to be processed for any reason, the Audit Queue Interface places a copy of the original message and a copy of the returned error message on the indicated queue.

Audit Topic Interface

When documents are submitted to the Publisher channel but fail to be processed for any reason, the Audit Topic Interface places a copy of the original message and a copy of the returned error message on the indicated topic.

Tracking Queue Interface

The Tracking Queue interface captures all messages that processed successfully and places a copy of the message on the tracking queue.

Tracking Topic Interface

The Tracking Topic interface captures all messages that processed successfully and places a copy of the message on the tracking topic.

Automatic Failover and Load Balancing

Multiple JMS drivers can be configured to retrieve messages from a single queue. Under this configuration, the queue feeds messages to each attached driver in a "round-robin" manner.

Automatic Failover

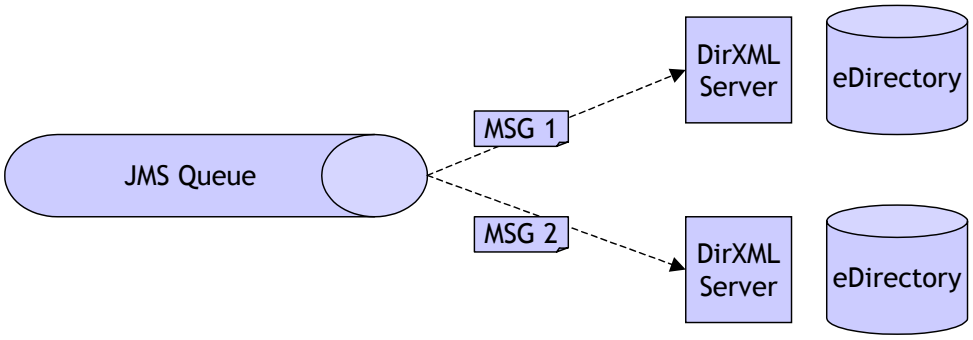
Fully automatic failover can be supported by configuring two JMS drivers that fetch messages from a single queue (or queue cluster). If one of the drivers fails for any reason, the second automatically continues processing. No messages can be lost during this process.

Load Balancing

When configured on multiple DirXML servers, the JMS driver supports true load balancing that can greatly scale the performance of DirXML. Where XML transaction volumes exceed the ability of the engine to keep pace, simply add one or more instances of the driver without modification.

****Note:** The load-balanced drivers must be hosted on separate DirXML servers in order to obtain the desired performance benefits.

The diagram below illustrates how the JMS Driver can be readily configured for automatic failover and load balancing.



Vendor Configurations

General Driver Configuration

All implementations of the JMS driver require these files to be copied to the \novelll\nds\lib directory or equivalent location, depending on the platform required.

- JMSDriver.Jar
- JMS.JAR
- Vendor specific JMS libraries

Supported JMS Vendor Implementations

- JMS
- Oracle Advanced Queuing
- IBUS Message Server
- TIBCOJMS
- TIBCO
- Novell
- Open JMS
- Sonic MQ
- Sun Message Queuing
- JBOSS
- Bea WebLogic
- SeeBeyond

JMS Summary

IBM WebSphere MQ (MQ Series) is the defining message queue product in its class and used in the majority of messaging implementations. WebSphere MQ implements full JMS compatibility in both standard and WebSphere MQ Everyplace (lightweight) versions.

Company Web site: www.ibm.com

MQ Series Specific Configurations

Versions 5.2 and below need the MA88 Support Pack (Java Messaging System support).

The Support pack is available as a free download from
http://www-3.ibm.com/software/integration/support/supportpacs/individual/ma88.html

For version 5.3 and higher, you must select custom installation and select the option to install Support for JMS.

If you want to use topics, you need the MA0C support pack, and you must start the Strmbrk process. The support pack is available as a free download from
http://www-3.ibm.com/software/integration/support/supportpacs/individual/ma0c.html

Vendor-Specific Support Files

- Connector.jar
- Com.ibm.jms.jar
- Com.ibm.jmsbind.jar
- Com.ibm.mqjms.jar

Authentication Information

The User ID that authenticates to the JMS environment must be a member of the MQM group.

Windows Environment Configuration

Add the JMS\Java\Lib directory to the Windows path environment variable.

Solaris Environment Configuration.

Add the following paths to the Java support files.

- /opt/mqm/bin:
- /opt/mqm/lib:
- /opt/mqm/java/lib

The path /opt/mqm/java/lib must be added to the LD_LIBRARY_PATH in the NDSD startup script for the MQ driver to find the supporting c libraries.

****Note:** You must also make sure you export the LD_LIBRARY_PATH.

JMS Remote Configuration

MQ Series Specific Configurations

Remote connections require that the JMS Client be installed.

Versions 5.2 and below need the MA88 Support Pack (Java Messaging System support). Later versions have integrated JMS support within the standard installation procedure.

For those that require it, the Support pack is available as a free download from http://www-3.ibm.com/software/integration/support/supportpacs/individual/ma88.html

For version 5.3 and higher, you need to select the custom installation and select the option to install support for JMS.

If you want to use topics, you need the MA0C support pack, and you must start the Strmbrk process. The support pack is available as a free download from http://www-3.ibm.com/software/integration/support/supportpacs/individual/ma0c.html

Vendor-Specific Support Files

- Connector.jar
- Com.ibm.jms.jar
- Com.ibm.jmsbind.jar
- Com.ibm.mqjms.jar

Authentication Information

Provide the User ID that authenticates to the JMS environment. The user must be a member of the MQM group.

Windows Environment Configuration

Add the JMS\Java\Lib directory to the Windows path environment variable.

Solaris Environment Configuration

Add the following paths to the Java support files.

- /opt/mqm/bin:
- /opt/mqm/lib:
- /opt/mqm/java/lib

Oracle Advanced Queuing (AQ)

Oracle AQ is the preferred approach used by Oracle to develop highly abstracted, efficient, and safe application integration. AQ is integral to all Oracle databases from 8.04 onwards and supports the JMS standard.

Company Web site: www.oracle.com

Oracle-Specific Configurations

Java Messaging System support is included in every version of Oracle from 8.04.

Supporting Files

- aqapi.jar (Advanced Queuing files)
- ojdbc14.jar (Oracle JDBC* thin driver support)

Authentication Information

The User ID that authenticates to the Oracle Advanced Queuing environment must be a member of the AQ Users Role or AQ Administrators Role.

Oracle Advanced Queuing Message Format

Some difficulty might be experienced when de-queuing a message generated by workflow and formatted by the queue handler. With some versions of Oracle, It is necessary to populate the agent section, header section, property section and the text section of the message. A JMS-120 is generated when the suggested sections were not populated appropriately. The error is as follows:

JMS-120: Dequeue failed
oracle.jms.AQjmsException: JMS-120: Dequeue failed
at oracle.jms.AQjmsError.throwEx(AQjmsError.java:233)
at oracle.jms.AQjmsConsumer.dequeue(AQjmsConsumer.java:1429)
at oracle.jms.AQjmsConsumer.receiveFromAQ(AQjmsConsumer.java:691)
at oracle.jms.AQjmsConsumer.receive(AQjmsConsumer.java:628)

An example of populating the message with each section defined correctly is as follows:

```
l_jms_text_message:= sys.AQ$_JMS_TEXT_MESSAGE ( sys.AQ$_JMS_HEADER
        (sys.AQ$_AGENT(' ', NULL, 0),
        NULL,
        'apps',
        NULL,
        NULL,
        NULL,
        sys.AQ$_JMS_USERPROPARRAY
          ( sys.AQ$_JMS_USERPROPERTY('jmsobject', 200, NULL, 12345, 23)
        )
       length(p_jms_text_message.text_vc), p_jms_text_message.text_vc, NULL
      );
```

Soft-Wired iBUS

iBus is a highly robust and cost effective java messaging product. Unique features include both Internet (SSL) and Mobile device connectivity.

Company Web site: www.soft-wired.com

iBUS Specific Configurations
Special Vendor Installation Options

Supporting Files

- IBUSmsrvClt.jar

Authentication Information

By default security is not implemented on the IBUS message server. If you want to enable username password access controls, you must explicitly enable the security manager module in the administration tool. This module allows you to assign ACLs (Access Control Lists) to queues or topics.

Environment Settings

- Windows

  The IBUSSRV_HOME environment variable must be set before running the server scripts. The installer sets the environment variable IBUSSRV_HOME to INSTALL_DIR\server. Use the *startserver.bat* batch file in the directory INSTALL_DIR\server\bin and the *stopserver.bat* batch file located in INSTALL_DIR\client\bin.

- UNIX

  Set the environment variable IBUSSRV_HOME to INSTALL_DIR/server. To start the server, use the startserver.sh shell script in INSTALL_DIR/server/bin. To stop the server, use the *stopserver.sh* shell script in INSTALL_DIR/client/bin. This script is an iBus client and can be run from any machine.

TIBCO JMS Server

This message bus provides real-time connectivity to dozens of platforms and applications.

Company Web site: www.tibco.com

Tibco JMS Specific Configurations

Vendor: TibcoJMS

Special Vendor Installation Options

Supporting Files

- TIBCOJMS.JAR

Authentication Information

By default, security is not implemented on the TIBCO JMS message server. If you want to use security, you must assign ACLs (Access Control Lists) to queues or topics.

Environment Settings

None.

Spiritsoft SpriritWave TIBCO-JMS Bridge

This third-party product provides native connectivity to the TIBCO Rendezvous Message Bus without requiring the TIBCO JMS Server.

Company Web site: www.spiritsoft.com

Special Vendor Installation

To properly register the Spiritsoft-TIBCO Bridge license file with the eDirectory instance, you must copy the Properties directory to the \novell\nds\jre\lib\ext directory.

Supporting Files

- rvconfig.jar
- tibrvj.jar
- tibrvjsd.jar

Novell Messaging Platform

Su

This product provides a high-performance JMS-compliant message bus that integrates with all Novell exteNd™ technologies as well as all other J2EE technology.

Company Web site: www.novell.com

Novell MP Specific Configurations

Vendor: Novell

Special Vendor Installation Options

Supporting Files

- MP-JMS.jar
- MPORB12.dll

Authentication Information

By default, security is not implemented on the Novell message server. If you want to use security, you must assign ACLs (Access Control Lists) to queues or topics.

Environment Settings

A search path must be set for the MPORB12.dll.

Sonic MQ

Sonic Message Queuing is a highly robust Java messaging product. Unique features include Sonic Continuous Availability Architecture and advanced clustering technologies.

Company Web site: http://www.sonicsoftware.com/

Sonic Specific Configurations

Special Vendor Installation Options

Supporting Files

- Sonic_Client.jar

Authentication Information

By default, security is not implemented on the Sonic message server. If you want to use security, you must enable the security manager module in the administration tool. This module allows you to assign ACLs (Access Control Lists) to queues or topics.

Sun Message Queuing

This product includes support for multi-broker message services, HTTP/HTTPS connections, secure connection services, scalable connection capability, and client connection failover .

Company Web site: http://www.sun.com/

Sun Message Queue Specific Configurations

Special Vendor Installation Options

Supporting Files

- IMQ.jar
- IMQBroker.JAR

Authentication Information

By default, security is not implemented on the SUN Message Queuing server. If you want to use security, you must the security manager module in the administration tool. This module allows you to assign ACLs (Access Control Lists) to queues or topics.

Environment Settings

Setting	Description
IMQ_HOME	• On Solaris, there is no root Message Queue installation directory. Therefore, IMQ_HOME is not used in Message Queue documentation to refer to file locations on Solaris. • On Windows, the root Message Queue installation directory is set by the Message Queue installer (by default, C:\Program Files\Sun\MessageQueue3). • On Windows, for Sun Java System Application Server, the root Message Queue installation directory is /imq under the Application Server base directory. • On Linux, there is no root Message Queue installation directory. Therefore, IMQ_HOME is not used in Message Queue documentation to refer to file loca-

	tions on Linux.
IMQ_VARHOME	The /var directory in which Message Queue temporary or dynamically-created configuration and data files are stored. It can be set as an environment variable to point to any directory. • On Solaris, IMQ_VARHOME defaults to the /var/imq directory. • On Solaris, for Sun Java System Application Server, Evaluation Edition, IMQ_VARHOME defaults to the IMQ_HOME/var directory. • On Windows, IMQ_VARHOME defaults to the IMQ_HOME\var directory. • On Windows, for Sun Java System Application Server, IMQ_VARHOME defaults to the IMQ_HOME\var directory. • On Linux, IMQ_VARHOME defaults to the /var/opt/imq directory
IMQ_JAVAHOME	An environment variable that points to the location of the Java runtime (JRE) required by Message Queue executables: • On Solaris, IMQ_JAVAHOME defaults to the /usr/j2se/jre directory, but you can optionally set the value to wherever the required JRE resides. • On Windows, IMQ_JAVAHOME defaults to IMQ_HOME\jre, but you can optionally set the value to wherever the required JRE resides. • On Linux, Message Queue first looks for the java runtime in the /usr/java/j2sdk*Version* directory, and then looks in the /usr/java/j2re*Version* directory. However, you can optionally set the value of IMQ_JAVAHOME to wherever the required JRE resides.

JBOSS MQ

A clean room implementation of the Java Message Service API part of the J2EE specification. It allows asynchronous delivery of messages in distributed systems with optional QOS parameters such as persistence, guaranteed delivery[comma] or transactions.

- JBoss-4.x supports the JMS1.1 version of the spec.
- JBoss-3.2.x supports the JMS1.0.2b spec.

Features

- Fire and Forget for asynchronous delivery.
- Guaranteed Delivery using persistent messages and durable subscriptions.
- JTA XA integration used by JBoss's JMS Resource Adapter.
- Pluggable Security to support different security mechanisms.
- Pluggable Persistence to support different persistence mechanisms.
- High Availability in a clustered environment.

Company Web site: http://www.jboss.org/

JBOSS MQ Specific Configurations

Special Vendor Installation Options

Supporting Files

- jbossall-client.jar

Authentication Information

By default, security is not implemented on the JBOSS Application server. If you want to use security, you must use the security manager module in the administration tool. This module allows you to assign ACLs (Access Control Lists) to queues or topics.

Bea WebLogic

BEA MessageQ is easy-to-use, fast, and reliable message software that allows applications to communicate using the industry-leading queued message bus technology. A proven and widely deployed middleware solution for distributed enterprise applications, BEA MessageQ allows the reliable exchange of guaranteed application messages across heterogeneous platforms. BEA MessageQ provides a robust application integration architecture for building high-performance message-based applications using multi-mode communications.

BEA MessageQ offers the following features:

- Interoperability with IBM platforms via TCP/IP, LU6.2.
- Publish and subscribe
- Large message size: up to 4 MB
- Self-describing messages for automatic data conversion between heterogeneous platforms
- Connectivity to BEA Tuxedo*, BEA eLink Platform*, MQSeries*, Legacy Applications, SAP R/3, and more

Company Web site: http://www.beasys.com

BEA Weblogic MessageQ Specific Configurations

Special Vendor Installation Options

Supporting Files

- WLClient.jar

Authentication Information

By default, security is not implemented on the Bea WebLogic server. If you want to use security, you must use the security manager module in the administration tool. This module allows you to assign ACLs (Access Control Lists) to queues or topics.

SeeBeyond

eGate* Integrator is a J2EE compliant and Web services based distributed integration platform that serves as the foundation of the SeeBeyond ICAN Suite 5, and is designed to dramatically lower the total cost of ownership (TCO) of developing, deploying, and managing integrations over time. eGate provides core integration including comprehensive systems connectivity, guaranteed messaging, and robust transformation capabilities while providing a unified, single sign-on environment for integration development, deployment, monitoring and management.

eGate Integrator is the first and only J2EE certified integration platform to support native operation of its integration technology over third-party J2EE application servers and includes the industry's first enterprise-scale integration change management tools, significantly lowering TCO.

Company Web site: http://www.seebeyond.com/

SeeBeyond Specific Configurations

Special Vendor Installation Options

Supporting Files

- TDB

Authentication Information

By default, security is not implemented on the SeeBeyond JMS server. If you want to use security you must use the security manager module in the administration tool. This module allows you to assign ACLs (Access Control Lists) to queues or topics.

OpenJMS

Product Web site: openjms.sourceforge.com

OpenJMS Specific Configurations

Vendor: Open Source

Authentication Information

By default, security is not implemented on the OpenJMS message server. If you want to use security, you must assign ACLs (Access Control Lists) to queues or topics.

Appendix A: JMS Overview

What Is JMS Messaging?

Messaging is a method of communication between software components or applications. A messaging system is a peer-to-peer facility; a messaging client can send messages to, and receive messages from, any other client. Each client connects to a messaging agent that provides facilities for creating, sending, receiving, and reading messages.

Messaging enables distributed communication that is *loosely coupled*. A component sends a message to a destination, and the recipient can retrieve the message from the destination. However, the sender and the receiver do not need to be available at the same time in order to communicate. In fact, the sender does not need to know anything about the receiver: nor does the receiver need to know anything about the sender. The sender and the receiver need to know only what message format and what destination to use. In this respect, messaging differs from tightly coupled technologies, such as Remote Method Invocation (RMI), which require an application to know a remote application's methods.

Messaging also differs from electronic mail (e-mail), which is a method of communication between people or between software applications and people. Messaging is used for communication between software applications or software components.

Message Types

- In JMS, a message is a Java object with 3 parts:Message header
- Message properties
- Message body

Message Header

Every JMS message includes message header fields that are always passed from producer to consumer. The purpose of the header fields is to convey extra information to the consumer outside the normal content of the message body. The JMS provider sets some of these fields automatically after a message is sent to the consumer, but the message producer has the opportunity to set some fields programmatically.

JMSCorrelationID

The JMSCorrelationID header field provides a way to correlate related messages. This is normally used for a request/response scenario. This can either be a vendor-specific ID, an application-specific string, or a provider-native byte value.

JMSExpiration or Time to Live

The JMSExpiration header field specifies the expiration or time-to-live value for a message. If the value is set to 0, the message never expires. When a message does expire, the JMS provider typically discards the message. Also, any persistent messages are deleted based on expiration values.

JMSMessageID

The JMSMessageID header field contains a value that uniquely identifies each message sent by a provider. This value is automatically set by the provider and returned to the message producer when the send method completes. All JMSMessageID values must start with an ID: prefix.

JMSPriority

JMS defines 10 priority levels, 0 through 9. 0 is the lowest priority and 9 is the highest. Levels 0–4 indicate a range of normal priorities, and levels 5–9 indicate a range of expedited priority. Priority level 4 is typically the default for a message producer.

Message Properties

The message property fields are similar to header fields described previously in the Message Header section, except these fields are set exclusively by the sending application.

Message Body

The message body carries free form application data, which can take several forms: text, serializable objects, byte streams, etc. The JMS API defines several message types (TextMessage,ByteMessage MapMessage, and others) and provides methods for delivering messages to and receiving messages from other applications.

JMS Messaging Models: Publish-and-Subscribe and Point-to-Point

JMS provides two types of messaging models: *publish-and-subscribe* and *point-to-point*. The JMS specification refers to these as *messaging domains*. In JMS terminology, publish-and-subscribe and point-to-point are frequently shortened to pub/sub and p2p (or PTP), respectively. This chapter uses both the long and short forms throughout.

In the simplest sense, publish-and-subscribe is intended for a one-to-many broadcast of messages, while point-to-point is intended for one-to-one delivery of messages.

Publish-and-Subscribe

In publish-and-subscribe messaging, one producer can send a message to many consumers through a virtual channel called a *topic*. Consumers can choose to *subscribe* to a topic. Any messages addressed to a topic are delivered to all the topic's consumers. Every consumer receives a copy of each message. The pub/sub messaging model is a push-based model, where messages are automatically broadcast to consumers without consumers requesting them or polling the topic for new messages.

In the pub/sub messaging model, the producer sending the message is not dependent on the consumers receiving the message. Optionally, JMS clients that use pub/sub can establish durable subscriptions that allow consumers to disconnect and later reconnect and collect messages that were published while they were disconnected.

Point-to-Point

The point-to-point messaging model allows JMS clients to send and receive messages both synchronously and asynchronously via virtual channels known as *queues*. The p2p messaging model has traditionally been a pull- or polling-based model, where messages are requested from the queue instead of being pushed to the client automatically. (The JMS specification does not specifically state how the p2p and pub/sub models must be implemented. Either one may use push or pull, but at least conceptually pub/sub is push and p2p is pull.)

- A queue canhave multiple receivers, but only one receiver canreceive each message. The JMS provider takes care of doling out the messages among JMS clients, ensuring that each message is consumed by only one JMS client. The JMS specification does not dictate the rules for distributing messages among multiple receivers.

Appendix B: Supporting National Language Code Pages

The driver supports different code pages for national languages such as Japanese by setting the JMS Queue Manager CCSIS to the code page corresponding to the UTF-8 character format for the target language.

To set the Queue Manager CCSID:

1. Execute the runmqsc program. This will bring up a DOS-like box
2. Enter the command: ALTER QMGR CCSID (1208)
   (In the example, the value 1208 represents the Japanese language UTF-8 code page. Substitute the correct UTF-8 code page ID for the required target language.)

Messages sent to the queue need to be in Unicode* format. Information on the Unicode standard is found at http://www.unicode.org/

Appendix C: Driver Parameters XML

```
<?xml version="1.0" encoding="UTF-8"?>
<driver-config name="JMS Driver">
	<driver-options>
		<qmgr display-name="Queue Manager">QM_appbox</qmgr>
		<msgbrk display-name="Message Broker"/>
		<mqchannel display-name="WebSphere MQ Channel"/>
		<sp display-name="Security Principal"/>
		<purl display-name="Provider Url">127.0.0.1</purl>
		<sc display-name="Security Credentials"/>
		<ss display-name="Security Protocol"/>
		<vb display-name="Verbose">TRUE</vb>
		<port display-name="OraclePort"/>
		<osid display-name="OracleSID"/>
		<ouser display-name="Oracle-Schema"/>
		<otable display-name="OracleQueueTable"/>
		<clientid display-name="JMS Client ID"/>
		<vendor display-name="Vendor">JBROKER</vendor>
	</driver-options>
	<subscriber-options>
		<qsend display-name="Queue Sender">subq</qsend>
		<tpub display-name="Topic Sender"/>
		<nonper display-name="Message persistence"/>
		<bytemessaging display-name="Byte Messaging"/>
		<jmsclient display-name="Client  Mode"/>
		<subqueryq display-name="Subscriber Query Queue"/>
		<subqueryreplyq display-name="Subscriber Query Reply Queue"/>
		<subquerytimeout display-name=" Subscriber Query Reply Timeout"/>
		<ttl display-name="JMS Time To Live"/>
		<pri display-name="JMS Priority"/>
	</subscriber-options>
	<publisher-options>
		<qrec display-name="Publisher Queue Receiver">pubq</qrec>
		<queryq display-name="Publisher Query Queue"/>
		<queryreplyq display-name="Publisher Query Reply Queue"/>
		<querytimeout display-name="Publisher Query Reply Timeout"/>
		<trec display-name="Topic Receiver"/>
		<headers display-name="Retrieve Headers"/>
		<jmsclient display-name="Client  Mode"/>
		<auditq display-name="Error Auditing Queue"/>
		<trackingq display-name="Transaction Tracking Queue"/>
		<correlationid display-name="Use Correlation ID'S"/>
	</publisher-options>
```

Appendix D: Solutions

RACF & CICS Integration

The RACF & CICS integration solution is a mainframe-administrator: friendly approach to integrating RACF and other host systems by using CLIST files. A set of included ReXX scripts extract messages containing information for the RACF, CICS, or other databases on the host. The solution uses IBM JMS to transport the information from the source (eDirectory™) to the mainframe and ReXX to execute. This is a very non-intrusive solution because it provides the mainframe administrators total control over the integration.

IBM Support Packs required:

- MA12 (Batch Trigger Monitor)

MA18 (MQ ReXX Support)
Both of these support packs are found on IBM's Web site at http://www-3.ibm.com/software/integration/support/supportpacs,
under Category 2, AS-IS SupportPacs

This solution uses trigger monitor submits a batch job that invokes a batch TSO TMP, where a ReXX exec program fetches messages from a queue.

Sample ReXX Script:

```
/* REXX */
/* TRACE !RESULTS */

PARSE UPPER ARG ,
  USID "|" ,
  PSWD "|" ,
  UNAME "|" ,
  BG "|" ,
  DG "|" ,
  OG "|" ,
  TSOFLG "|" ,
  PRACNO "|" ,
  JOBF "|" ,
  REG "|" ,
  LOC "|" ,
  VMID "|" ,
  EMPTYPE "|",
  HRT "|"

 MAXCC = 0
 UEXIST = BLANK

 SAY "*** START PROCESSING USID="USID "***"
```

```
 SAY "UNAME="UNAME "VMID="VMID
 SAY "BG="BG "DG="DG "OG="OG
 SAY "TSOFLG="TSOFLG "PRACNO="PRACNO "JOBF="JOBF
 SAY "REG="REG "LOC="LOC "EMPTYPE="EMPTYPE "HRT="HRT

 IF HRT = TER THEN DO
   MAXCC = 0
   SAY "*** NOT PROCESSED: TERMINATION ***"
   EXIT MAXCC
   END

 IF USID = '' THEN DO
   MAXCC = 0
   SAY "*** PROBLEM: NO USERID ***"
   EXIT MAXCC
   END

 AP = POS("'",UNAME)
 IF AP > 0 THEN UNAME = INSERT("'",UNAME,AP)

 X = OUTTRAP('LUOUT',0)
"LU" USID
 LURC = RC
 X = OUTTRAP('OFF')

 SELECT
   WHEN LURC = 0 THEN DO
    "CO" USID "GROUP("DG")"
    "ALU" USID "RESUME"
    "RE" USID "GROUP(@ISTERM)"
     UEXIST = Y
     IF MAXCC < 4 THEN MAXCC = 4
     SAY "*** NOTICE:" USID "EXISTED ***"
     END
   WHEN LURC = 4 THEN DO
    "AU" USID "OW("OG") DFLTGRP("DG")"
     AURC = RC
     UEXIST = N
     END
   OTHERWISE SAY "*** PROBLEM: LU ***"
 END

 IF HRT = REH THEN "CO" USID "GROUP(@ISTERMX) OW(ISSD01)"

 IF UEXIST = N THEN ,
   IF AURC ¬= 0 THEN DO
     IF MAXCC < AURC THEN MAXCC = AURC
     SAY "*** PROBLEM: AU ***"
     END
```

```
"ALU" USID "OW("OG") DFLTGRP("DG") PASSWORD("PSWD")" ,
   "NAME(""UNAME"") DATA(""VMID"") CICS(OPIDENT(HAL))"
 ALRC = RC

 IF ALRC ¬= 0 THEN DO
   IF MAXCC < ALRC THEN MAXCC = ALRC
   SAY "*** PROBLEM: ALU ***"
   IF MAXCC > 4 THEN EXIT MAXCC
   END

 SELECT
   WHEN SUBSTR(USID,1,2) = @N THEN DO
     UUID = 80||SUBSTR(USID,3,4)
    "ALU" USID "OMVS(UID("UUID") HOME(/) PROGRAM(/bin/echo))"
     END
   OTHERWISE DO
     UUID = 1||SUBSTR(USID,2,5)
    "ALU" USID "OMVS(UID("UUID") HOME(/) PROGRAM(/bin/echo))"
     END
 END

 IF TSOFLG ¬= Y THEN EXIT MAXCC

"CO" USID "GROUP("BG") OW("OG")"

 SELECT
   WHEN OG = @TBADM01 THEN DO
     PRCD = LOGONU
     CMND = "EX "BAS0000.P.TOOLS.CLIST(LOGON)""
     END
   OTHERWISE DO
     PRCD = LOGONI
     CMND =
     END
 END

"ALU" USID "TSO(ACCTNUM(00000-00521109-001-51-238S)" ,
  "SIZE(8192) MAXSIZE(8192) JOBCLASS(A) MSGCLASS(S) SYSOUTCLASS(S)" ,
  "DEST(LOCAL) UNIT(SYSDA) PROC("PRCD") COMMAND(""CMND""))"

 V1 = RANDOM(0,7)
 ACCNT = 0

 X = OUTTRAP('LCAT',0)
"LISTCAT ENTRIES(""USID"")"
 LCRC = RC
 X = OUTTRAP('OFF')
```

```
 SELECT
   WHEN LCRC = 0 THEN DO
     IF MAXCC < 4 THEN MAXCC = 4
     SAY "*** NOTICE: CAT ENTRY EXISTED ***"
     END
   WHEN LCRC = 4 THEN DO
     DO UNTIL ACRC = 0 | ACCNT = 8
       TUCAT = SYS1.UCATTSO||V1
       IF TUCAT = SYS1.UCATTSO0 THEN TUCAT = SYS1.UCATTSO
      "DEFINE ALIAS (NAME('"USID"') RELATE('"TUCAT"'))"
       ACRC = RC
       ACCNT = ACCNT + 1
       V1 = V1 + 1
       IF V1 = 8 THEN V1 = 0
     END
     IF ACRC ¬= 0 THEN DO
       IF MAXCC < ACRC THEN MAXCC = ACRC
       SAY "*** PROBLEM: DEFINE ALIAS ***"
       EXIT MAXCC
       END
     END
   WHEN LCRC > 4 THEN DO
     IF MAXCC < LCRC THEN MAXCC = LCRC
     SAY "*** PROBLEM: CATALOG ENTRY ***"
     END
 END

"ALLOC DA('"USID".ISPPROF') NEW SPACE(2,1) DIR(35) TRACKS" ,
  "DSORG(PO) RECFM(F B) LRECL(80) BLKSIZE(6160)"
"FREE DA('"USID".ISPPROF')"
"ADDSD '"USID".*' OWNER("USID") UACC(NONE) AUDIT(NONE)"
"PE '"USID".*' ID("BG") ACC(READ) GEN"
"ADDSD '"USID".SYSOUT.*' OWNER("USID") UACC(NONE) AUDIT(NONE)"
"PE '"USID".SYSOUT.*' ID("BG") ACC(READ) GEN"

 SELECT
   WHEN PRACNO = 11 THEN NOP
   WHEN PRACNO = 14 THEN NOP
   WHEN PRACNO = 15 THEN NOP
   WHEN PRACNO = 17 THEN NOP
   WHEN PRACNO = 19 THEN ,
     TBIDFLAG = YES
   WHEN PRACNO = 20 THEN NOP
   WHEN PRACNO = 24 THEN ,
     TBIDFLAG = YES
   WHEN PRACNO = 27 THEN NOP
   WHEN PRACNO = 32 THEN NOP
   WHEN PRACNO = 40 THEN NOP
   WHEN PRACNO = 41 THEN NOP
```

```
  WHEN PRACNO = 43 THEN NOP
  WHEN PRACNO = 44 THEN NOP
  WHEN PRACNO = 64 THEN ,
   "PE '"USID".SYSOUT.*' ID(@HETEC01) ACC(ALTER) GEN"
  WHEN PRACNO = 67 THEN NOP
  OTHERWISE
 END

 IF TBIDFLAG = YES THEN DO
  USIB = USID||'B'
  X = OUTTRAP('LUBOUT',0)
 "LU" USIB
  LUBRC = RC
  X = OUTTRAP('OFF')
  SELECT
    WHEN LUBRC = 0 THEN DO
     "ALU" USIB "RESUME"
     "RE" USIB "GROUP(@ISTERM)"
      IF MAXCC < 4 THEN MAXCC = 4
      SAY "*** NOTICE:" USIB "EXISTED ***"
      EXIT MAXCC
      END
    WHEN LUBRC = 4 THEN DO
     "AU" USIB "OW(@BAADM01) DFLTGRP(@TBHLD) PASSWORD("PSWD")" ,
       "NAME('"UNAME"') DATA('TBA B ID') CICS(OPIDENT(HAL))" ,
       "TSO(ACCTNUM(00000-00521109-001-51-238S) PROC(LOGONB)" ,
       "SIZE(2048) MAXSIZE(8192) JOBCLASS(A) MSGCLASS(S) " ,
       "SYSOUTCLASS(S) DEST(LOCAL) UNIT(SYSDA))"
      ABRC = RC
      IF ABRC ¬= 0 THEN DO
        IF MAXCC < ABRC THEN MAXCC = ABRC
        SAY "*** PROBLEM: AU BID ***"
        EXIT MAXCC
        END
      ELSE DO
       "CONNECT" USIB "GROUP(@ISTBAB) OW(@ISADM01)"
        IF TUCAT = TUCAT THEN TUCAT = SYS1.UCATTSO
       "DEFINE ALIAS (NAME('"USIB"') RELATE('"TUCAT"'))"
       "ALLOC DA('"USIB".ISPPROF') NEW   SPACE(2,1) DIR(35) TRACKS " ,
         "DSORG(PO) RECFM(F B) LRECL(80) BLKSIZE(6160)"
       "FREE DA('"USIB".ISPPROF')"
       "ADDSD '"USIB".*' OWNER("USIB") UACC(NONE) AUDIT(NONE)"
       "PE '"USID".*' ID("USIB") ACCESS(ALTER) GEN"
       "ADDSD '"USIB".P.AQUA.*' OWNER("USIB") UACC(NONE) AUDIT(NONE)"
       "PE '"USIB".P.AQUA.*' ID(@ISPBIDS) ACCESS(READ)"
       "PE '"USIB".P.AQUA.*' ID(@DBDBA01) ACCESS(READ)"
      "ADDSD '"USID".P.AQUA.*' OWNER("USID") UACC(NONE)" ,
        "FROM('"USID".*')"
       "PE '"USID".P.AQUA.*' ID(@DBDBA01) ACCESS(READ)"
```

```
       "RACLINK ID("USID") DEFINE(HALINMA1."USIB") PEER(PWSYNC)"
        END
      END
    OTHERWISE SAY "*** PROBLEM: LU B ID ***"
  END
END

RETURN MAXCC
```

Oracle Integration

Company Web site: www.oracle.com
Oracle provides a set of JMS interfaces and associated semantics called Advanced Queue (AQ) that define how a JMS client accesses the facilities of Oracle by using industry-standard Java messaging.

Oracle JMS supports the standard JMS interfaces and has extensions to support the AQ administrative operations and other AQ features that are not a part of the JMS standard.

The Oracle Advanced Queuing interfaces allow the developer to quickly and safely integrate with a number of Oracle applications through messaging, as opposed to directly updating database tables. This allows APIs developed in PLSQL and Java to prepare, validate, execute, roll-back and post-process all DirXML updates to Oracle to ensure complete data integrity. Applications that can be integrated in this fashion include:

- Oracle HR
- Oracle Financials
- Oracle E-Business Suite

Oracle Workflow

Oracle Workflow is an increasingly popular toolset that automates and streamlines business processes both internal and external to the enterprise. It supports traditional applications-based workflows as well as Identity Management integration workflows.

The Workflow Engine embedded in the Oracle database server monitors workflow states and coordinates the routing of activities for a process. Changes in workflow state, such as the completion of workflow activities, are signaled to the engine via a Java API. Based on flexibly defined workflow rules, the engine determines which activities are eligible to run and the schedule that they use.

Oracle Workflow leverages Oracle Advanced Queuing and enables DirXML® application integration at the business process level. Business event messages from DirXML can be placed on or received from Oracle Advanced Queues and consumed by Oracle Workflow. The HTTP and HTTPS communications protocols are also supported.

Oracle Workflow Business Event System

A business event is triggered in an application whenever something of significance happens. An example of a business event is the hiring of an employee or the creation of a customer. The Business Event System consists of the Event Manager, which lets

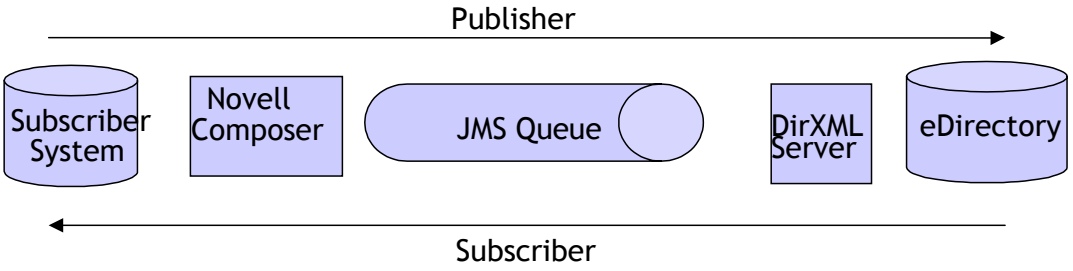
you register subscriptions to significant events, and event activities, which let you model business events within workflow processes.

When a local event occurs, the subscribing code is executed in the same transaction as the code that raised the event, unless the subscriptions are deferred. Subscription processing can include executing custom code on the event information, sending event information to a workflow process, and sending event information to other queues or systems.

Novell exteNd Composer

Company Web site: http://www.novell.com/

Novell **exteNd Composer**™ is an award-winning extraction, loading, and transformation (ETL) product that provides powerful data manipulation and extensive connectivity tools. When configured with the DirXML Driver for JMS/MQ, it enables eDirectory to connect to hundreds of applications, platforms, and data formats.



Novell exteNd Composer Connectors are available for a wide variety of back-end environments, user interfaces, and data formats. They are also very easy to use. With these Connectors and the exteNd Composer's simple drag-and-drop functionality, you can transform an application's input and output into Extensible Markup Language (XML), thereby providing the Web connectivity today's users and business climate demand. Novell exteNd Composer Connectors are available for the following environments and applications:

exteNd Composer 3270 Connect
With exteNd Composer 3270 Connect, you can XML-enable data from systems that provide a 3270 Terminal Data Stream (TDS) interface, which connects these systems to IBM mainframe dumb terminals.

exteNd Composer 5250 Connect
The NovellexteNd Composer 5250 Connect XML-enables data from AS/400* legacy systems that provide a 5250 TDS interface. (AS/400 systems use this interface to connect dumb terminals to IBM mainframes.)

exteNd Composer Data General Connect
This exteNd Composer Connector XML-enables data from legacy Data General and DG/UX applications, which run on Dasher 412-based terminal systems.

exteNd Composer EDI Connect
The exteNd Composer EDI Connect converts messages from Electronic Data Interchange (EDI) systems to XML documents. Conversely, this Connector converts XML documents to messages that comply with American National Standards Institute (ANSI) X.12 and Electronic Document Interchange for Administration, Commerce and Transportation (EDIFACT) specifications.

exteNd Composer HP3000 Connect
Using the exteNd Composer HP3000 Connect, you can XML-enable data from HP* 700/92 terminal-based systems, which run on HP3000 and HP-UX* operating systems.

exteNd Composer HTML Connect

exteNd Composer HTML Connect transforms Hypertext Markup Language (HTML) navigation and interactions into Extensible Hypertext Markup Language (XHTML). In other words, this Connector XML-enables these transactions, which you can then repurpose via Web services.

exteNd Composer JDBC Connect

exteNd Composer includes this valuable Connector, which XML-enables data from databases that expose a Java Database Connectivity (JDBC) interface. You can use exteNd Composer JDBC Connect to create application components that process data from, and return processed data toJDBC databases.

exteNd Composer LDAP Connect

Included with exteNd Composer, this useful Connector enables you to XML-enable directory-aware applications. You can use exteNd Composer LDAP Connect to create applications that read information from, or write information to, any LDAP-compliant directory.

exteNd Composer JMS Connect

With exteNd Composer JMS Connect, you can XML-enable Message Oriented Middleware (MOM) applications, which send and receive messages via Java Message Service (JMS).

exteNd Composer SAP Connect

exteNd Composer SAP Connect enables SAP R/3 applications to receive XML requests and return XML responses. You can use this Connector to repurpose the core business processes—such as accounts receivable and payable processes—currently in your organization's SAP applications.

exteNd Composer T27 Connect

With exteNd Composer T27 Connect, you can XML-enable applications running on Unisys* mainframe computers—such as A Series, V Series and Clearpath NX Series mainframes. This Connector enables you to include many Unisys applications—including LINC and MARC—in your systems integration or Web services solution.

exteNd Composer Tandem Connect

Using exeteNd Composer Tandem* Connect, you can XML-enable data from applications running on Tandem 6530 terminal-based systems.

exteNd Composer CICS RPC

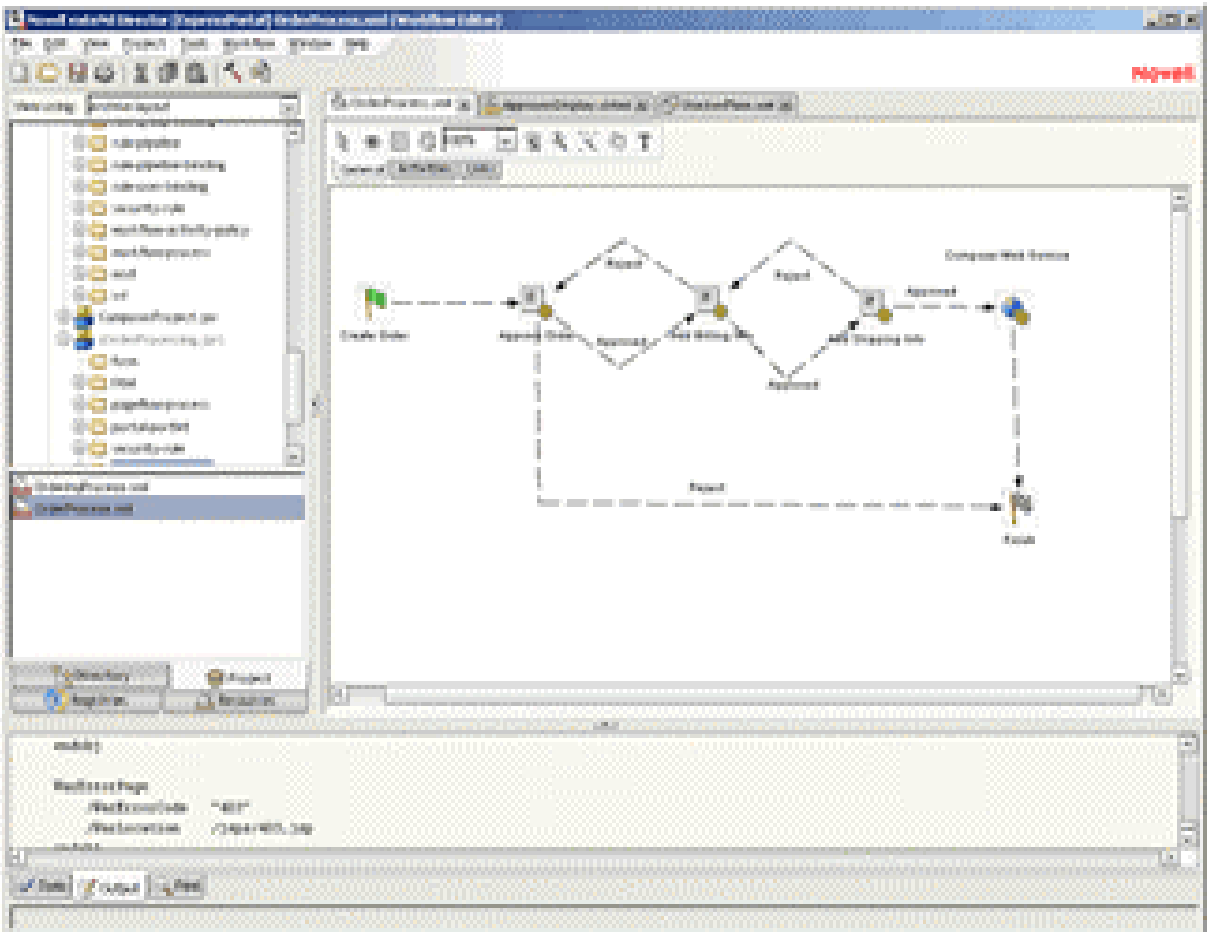
exteNd Composer CICS RPC is a Connector that XML-enables data from Customer Information Control System (CICS) Remote Procedure Call (RPC) systems. These transaction-processing systems receive XML requests and return XML responses through IBM's CICS Java Gateway.

exteNd Composer Telnet Connect

exteNd Composer Telnet Connect XML-enables data from ANSI terminal [hyphen] based and VT-series systems-[comma instead of hyphen]which are systems that use the Virtual Telecommunications Access Method (VTAM).

The Novell exteNd Director Workflow system

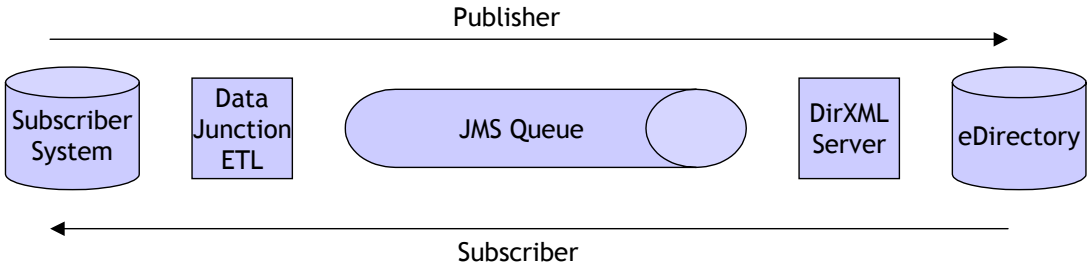
- An intuitive design and development workflow process designer. Allows you to quickly and visually create a workflow process, bringing the power of visual design to workflow development.
- Browser-based workflow administration console. Includes a user interface that makes it exceptionally easy to remotely administer running workflow queues, processes and the workflow engine.
- An open, extensible, and flexible architecture. Delivers integration with Nsure™ Identity Manager via JMS Java Messaging Systems.
- J2EE compatibility. Ensures that your workflow applications are portable, scalable, secure, and vendor-neutral.



Data Junction Integration

Company Web site: www.datajunction.com

Data Junction is an award-winning extraction, loading, and transformation (ETL) product that provides powerful data manipulation and extensive connectivity tools. When configured with the DirXML Driver for JMS/MQ, it enables eDirectory to connect to hundreds of applications, platforms, and data formats.



The Data Junction product suite provides a Visual Map Designer, Process Designer, Structure Schema Designer, Document Schema Designer, and Extract Schema Designer.

In addition, the Data Junction suite includes highly specialized adapters for a variety of application services, all of which will integrate seamlessly with DirXML by using the JMS driver. These include the following:

Dialog Adaptors:

- Biographical
- Business & News
- Commerce Business Daily
- Company Directory
- Federal Registry
- Market Research
- Patents
- Sci/Tech & Bio/Med
- Trade Names & Products
- Trademarks

Dodge Adaptors

- Bidders
- Firms
- Firms & Bidders
- Projects

Application Adaptors:

- AccPac
- EDI (X12)
- EDI (EDIFACT)
- HIPAA
- HL7
- SAP (IDoc)I
- SWIFT (Target only)
- STN Chemical
- Dow Jones*
- Internet Email
- Reuters*
- Xerox*

HIPAA Adaptors

- HIPAA Adapter users: Refer to HIPAA for a detailed list of instructions on how to connect to HIPAA Health Claim Source Files.

General Adaptors

- Access 2000
- Access 97
- Access XP
- AccountMate
- ACT! for Windows
- Acucobol (ODBC 3.x)
- Adabas (NatQuery)*
- Alpha Four
- Apache Common Logfile Format
- ASCII (Delimited)
- ASCII (Fixed)
- BAF
- Binary Line Sequential
- Binary
- BizTalk* XML
- Btrieve* v6
- Btrieve v7
- C-ISAM
- C-tree 4.3
- C-tree Plus
- Cambio
- Clarion
- Clipper*
- Cloudscape* (ODBC)
- Cobol Flat File
- Common Logfile Format Webserver
- Content Extractor
- Data Junction* Log File
- DataEase*
- DataFlex (ODBC 3.x)
- dBASE II
- dBASE III+
- dBASE IV
- dBASE V (IDAPI)
- DIALOG Biographical
- DIALOG Business and News
- DIALOG Commerce Business Daily
- DIALOG Company Directory
- DIALOG Federal Register

- DIALOG Market Research
- DIALOG Patents
- DIALOG Sci_Tech and Bio_Med
- DIALOG Trade Names and Products
- DIALOG Trademarks
- dialog.djx
- DIF
- EDI (X12)
- edi4.djx
- EDIFACT
- Enable
- Excel 2000
- Excel 95
- Excel 97
- Excel v2
- Excel v3
- Excel v4
- Excel v5
- Excel XP
- eXcelon 2.x
- eXcelon XIS 3.0
- Extractor
- FIX
- FIXML
- Folio Flat File
- Foxbase+*
- FoxPro*
- Fujitsu Cobol
- GoldMine* Import File (dbf)
- GoldMine
- Great Plains DOS (Btrieve)
- Great Plains Unix_Mac (C-tree)
- HCFA1500 - NSFA
- HIPAA
- Hitachi HiRDB (ODBC)
- HTML
- IBM* DB2 7.2 Universal Database Multimode
- IBM DB2 Loader
- IBM DB2 UDB Mass Insert
- IBM DB2 Universal Database Multimode
- IDAPI
- Informix* (ODBC 3.x)r
- Informix DB Loader
- Informix-Online DS Multimode
- Informix_SE
- Ingres (ODBC 3.x)
- Interbase (IDAPI)
- Join engine

- LDAP
- LDIF
- Lotus* 123 r1A
- Lotus 123 r2
- Lotus 123 r3
- Lotus 123 r4
- Lotus Note*s 5
- Lotus Notes Structured Text
- Lotus Notes Structured Text
- Lotus Notes
- Macola Acct (Btrieve)
- Magic PC
- MAILER'S+4 (dBase)
- Micro Focus* COBOL
- Microsoft* COBOL
- Microsoft IIS Extended Logfile Format
- MUMPS (ODBC)
- Navision Financials (ODBC 3.x)
- NonStop SQL_MX (ODBC)
- ODBC 3.5 MultiMode
- ODBC 3.5
- ODBC 3.5
- ODBC 3.x Mass Insert
- ODBC 3.x MultiMode
- ODBC 3.x
- ODBC 3.x
- Oracle 7.x
- Oracle 7.x
- Oracle 8.x Multimode
- Oracle 8.x
- Oracle 8.x
- Oracle 9i Multimode
- Oracle 9i
- Oracle 9i
- Paradox* v5 (IDAPI)
- PayChex* (DJF) Import
- Personal Librarian
- Pervasive* SQL
- Platinum Acct (Btrieve)
- PostgreSQL*
- Progress* (ODBC 3.x)
- Quattro Pro* Windows v5
- RBase (ODBC 3.x)
- RealWorld Acct (MFCOBOL)
- Red Brick
- Remedy* ARS
- Report Reader
- Rich Text Format

- salesforce.com
- SAP (IDoc)
- SAS Transport Format
- SBT Acct (FoxPro)
- Scalable SQL
- Sequential Binary
- SGML
- Solomon Acct (Btrieve)
- SPLUS
- SPSS
- SQL Script
- SQL Server 2000
- SQL Server 6.x
- SQL Server 7
- SQL Server BCP
- SQL Server Mass Insert
- SQLBase* 6.X
- SQLBase
- Statistica
- SWIFT
- Sybase* Adaptive Server* 11.x
- Sybase Adaptive Server 12.x
- Sybase BCP
- Sybase SQL Anywhere 6
- Sybase SQL Anywhere
- Sybase SQL Server Mass Insert
- Sybase SQL Server System 11 Multimode
- SYSTAT
- Tape Drive Sequential
- Teradata (Fastload)
- Text (Delimited - EDI)
- Text (Delimited - EDI)
- Text (Delimited - EDIFACT)
- Text (Delimited - EDIFACT)
- Text (Delimited - HL7)
- TRADACOMS
- UB92 - NSF
- Unicode (Delimited)
- Unicode (Fixed)
- USMARC
- Variable Sequential (MVS)
- Velocis (ODBC 3.x)
- Visual dBASE* 5.5
- Visual FoxPro
- WATCOM* SQL v5
- WordPerfect* 6.0 (Mail Merge)
- XDB (ODBC 3.x)
- XML