Java technologies -- table of contents

Development

Administration

4.6: Java Technologies

The J2EE (Java TM 2 Platform Enterprise Edition) technologies providestandard architectures for defining and supporting a multi-tiered programming model.

The technologies support all application components, namely:

- Application clients
- Enterprise JavaBeans TM
- Servlets and JavaServer PagesTM
- Applets

The Java technologies are:

- JavaMail
- Java Naming and Directory Interface (JNDI)
- Java Message Service (JMS)

See the *Related information* links for information on other programming topics.

4.6.1: Using JavaMail

WebSphere Application Server supports JavaMail version 1.1.3 and the JavaBeans Activation Framework (JAF) version 1.0.1.

In WebSphere Application Server, JavaMail is supported in all Web applicationcomponents, namely:

- servlets
- JSP files
- enterprise beans
- application clients

The JavaMail APIs model a mail system. These APIs provide a platform and protocol independent framework to build Java based, e-mail clientapplications. The JavaMail APIs only provide general mail facilities for reading and sending mail. These APIs require service providers to implement the protocols.

In addition to service providers, JavaMail requires the JavaBeans Activation Framework or JAFto handle mail content that is not plain text as, for example, MIME (Multipurpose Internet Mail Extensions),URL (Uniform Resource Locator) pages, and file attachments.

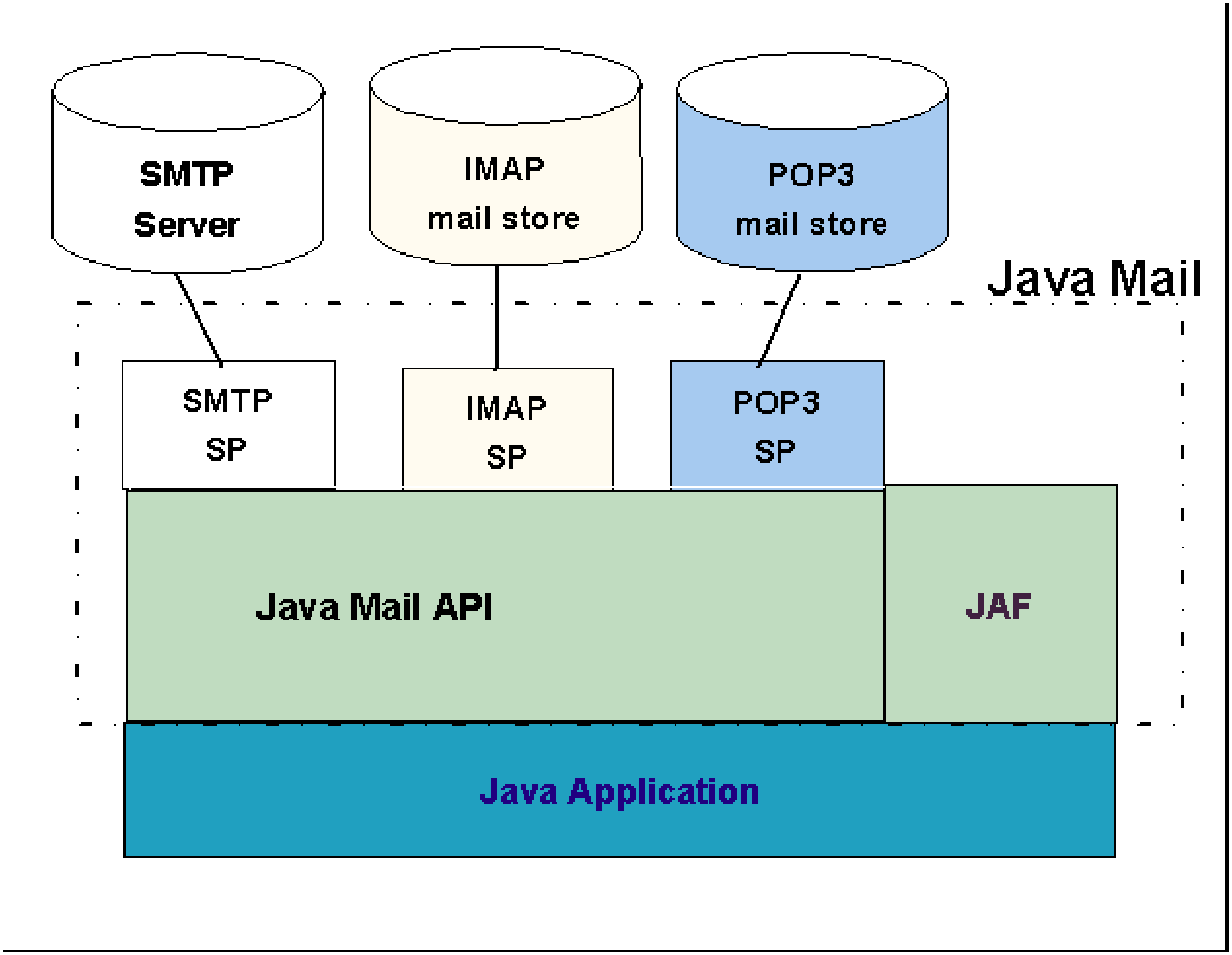
The service providers implement specific protocols. For example, SMTP (or SimpleMail Transfer Protocol), is a transport protocol for sending mail. POP3 or Post Office Protocol 3 is the standardprotocol for receiving mail. IMAP or Internet Message Access Protocol is an alternative protocol to POP3.

The following graphic illustrates the relationship among the different JavaMail components:

- JavaMail APIs
- JavaBeans Activation Framework
- Service providers
- Mail protocols

The dotted line around specific objects represents the grouping that comprises a working JavaMail installation. With the exception of POP3, all the components in the installation view are shipped as part of WebSphere Application Server using the following Sun licensed packages:

- **mail.jar -** contains JavaMail APIs, the SMTP service provider, and the IMAP service provider.
- **activation.jar -** contains the JavaBeans Activation Framework.


SMTP
Server
IMAP
mail store
POP3
mail store
Java Mail
SMTP
SP
IMAP
SP
POP3
SP
Java Mail API
JAF
Java Application

4.6.1.1: Writing JavaMail applications

According to the J2EE specifications, each *javax.mail.Session*instance must be treated as a resource factory. Therefore, to use JavaMail, do the following:

1. Declare mail resource references in your application component's deployment descriptors,as described in this example:
   ```
   <resource-ref><description>description</description><res-ref-name>mail/MailSession</res-ref-name><res-type>javax.mail.Session</res-type><res-auth>Container</res-auth></resource-ref>
   ```
2. Configure, during deployment, each referenced mail resource.See article, 4.6.1.2: Configuring JavaMail, for a description of the parameters requiredto configure a mail resource.
3. Locate in your application component, during runtime, each specific JavaMail session using JNDI lookup. An example of the code follows:
   ```
   Session session = (Session)ctx.lookup("java:comp/env/mail/MailSession");
   ```

   Your application component can now use *session* to create messages and get store access.

Coding example for sending and saving a message

The following code segment shows how an application component sends a messageand saves it to the mail account's ***Sent*** folder:

```
javax.naming.InitialContext ctx = new javax.naming.InitialContext();
mail_session = (javax.mail.Session) ctx.lookup("java:comp/env/mail/MailSession");

MimeMessage msg = new MimeMessage(mail_session);
msg.setRecipients(Message.RecipientType.TO, InternetAddress.parse("bob@coldmail.net"));

msg.setFrom(new InternetAddress("alice@mail.eedge.com"));
msg.setSubject("Important message from eEdge.com");
msg.setText(msg_text);
Transport.send(msg);

Store store = mail_session.getStore();
store.connect();
Folder f = store.getFolder("Sent");
if (!f.exists()) f.create(Folder.HOLDS_MESSAGES);
f.appendMessages(new Message[] {msg});
```

See the related information links for the JavaMail APIs.

4.6.1.2: Configuring JavaMail

A mail resource is configured using appropriate system management facilities,as for example, the administrative console.

Refer to article 6.6.37: Administering mail providers and mail sessionsfor detailed configuration instructions.

> ℹ The "mail originator" setting (the exact name in the administrative console varies depending on the product edition; see 6.6.37.0.1 for a description) can beoverridden for individual messages in your application, using method***Message.setFrom()***

See the related topics for more JavaMail documentation.

4.6.1.3: Debugging JavaMail

There will be times when you need to debug your JavaMail applications.One option is to turn on JavaMail's debugging feature. With this option on, JavaMail will print to ***stdout*** its interactions with the mail servers. These interactions are printed in detail, in a step-by-step format.

> ℹ With WebSphere Application Server,***stdout*** and ***stderr*** are usually redirected to files.The specific file paths can be set with an application server's *Properties* > *File* panel.For example, for the Default Server, ***stdout*** is redirected by defaultto the file:
> **`<WAS_HOME>\logs\default_server_stdout.log`**
> See the Problem determination section for moreinformation on ***stdout*** and ***stderr***.

Enable debugging programmatically, through the console, or through the command line.

- The easiest way to turn on debugging is to call methodsetDebug()on the mail session after *session* is obtained through a JNDI lookup, as shownbelow:

  ```
  javax.naming.InitialContext ctx = new javax.naming.InitialContext();
  mail_session = (javax.mail.Session) ctx.lookup("java:comp/env/mail/MailSession");
  mail_session.setDebug(true);
  ...
  ```

  This debugging approach requires re-compiling and, very likely, re-loading the applicationcomponent in which this code is embedded. This approach may be impractical at times.
- One alternative is to set the system property ***mail.debug*** through the console.If your JavaMail code is embedded in a servlet, a JSP, or an EJB, from the Administrative Console:
  1. Select the server that owns the component
  2. Open the ***Properties*** > ***JVM Settings*** panel
  3. Add an entry to ***System Properties*** with ***Name***=mail.debug and ***Value***=true.
  4. Restart the server.
- If your JavaMail code is in a Java client which is invoked from the commandline, add the ***-Dmail.debug=true*** flag to the *java* command, and the debugging output will be displayed in the command window.

  > ℹ Property ***mail.debug***,set with the last two approaches, is shared by all mail session instances within the sameJVM process. When debugging is enabled in this manner, JavaMail will print out step-by-step, mail-relatedinteractions to ***stdout*** for all these mail sessions.

The output in ***stdout*** looks like the following example:

```
...


DEBUG: getProvider() returning javax.mail.Provider[TRANSPORT,smtp]
DEBUG SMTP: useEhlo true, useAuth false

DEBUG: SMTPTransport trying to connect to host "smtp3.eedge.com", port 25

DEBUG SMTP RCVD: 220 relay14.eedge.com ESMTP Sendmail; Tue, 19 Dec 2000 15:08:42 -0700

DEBUG: SMTPTransport connected to host "smtp3.eedge.com", port: 25

DEBUG SMTP SENT: EHLO y2001
DEBUG SMTP RCVD: 250-relay14.eedge.com Hello testpc.eedge.com, pleased to meet you
250-8BITMIME
250-SIZE 20000000
250-DSN
250-ONEX
250-ETRN
250-XUSR
250 HELP


DEBUG SMTP SENT: MAIL FROM:<alice@mail.eedge.com>
DEBUG SMTP RCVD: 250 <alice@mail.eedge.com>... Sender ok


DEBUG SMTP SENT: RCPT TO:<bob@coldmail.net>
```

```
DEBUG SMTP RCVD: 250 <bob@coldmail.net>... Recipient ok


Verified Addresses
 bob@coldmail.net
DEBUG SMTP SENT: DATA
DEBUG SMTP RCVD: 354 Enter mail, end with "." on a line by itself


DEBUG SMTP SENT:
...
DEBUG SMTP RCVD: 250 PAA125654 Message accepted for delivery


DEBUG SMTP SENT: QUIT
```

4.6.1.4: Running the JavaMail sample

The JavaMail sample is packaged in an ear file called ***jmsample.ear*** that is locatedin directory:

`product_installation_root/installableApps`

The JavaMail sample contains three application components:

1. one stateful session bean
2. one servlet
3. one JSP file

Each component, when invoked, gathers data to compose a mail message,and then sends the message. Optionally, the sent message can also be saved in anIMAP account, in a folder named *Sent*. See article Writing JavaMail applicationsfor the coding example to send a message and create the *Sent* folder.

Complete the following tasks to use ***jmsample.ear***:

1. Set up mail servers
2. Configure a MailSession resource
3. Install the ***jmsample.ear*** components

---

1. Set up mail servers

To send out email messages, you need a mail transport server.Since SMTP is the most widely used transport protocol, such a mailserver is also known as a SMTP server. An alternative to installing and configuringyour own SMTP server, is to use an existing implementation.For example, if your Internet mail address is *john_smith@mycompany.com*,then *mycompany.com* could serve as your SMTP server. Ask your company's email administratorfor more information.

The components in the ***jmsample.ear*** file can optionally save a copy of thesent message into an email account. If you plan to try this capability, you will also need to set up an IMAP mail account.

2. Configure a MailSession resource

See article Properties related to JavaMail supportfor information on MailSession resource properties.

One property that requires your attention when you configure the MailSession resource for ***jmsample.ear*** is ***JNDI Name***.You can explicitly define this property or allow System Management to define it for you.

Since for all componentsin this sample the MailSession resource references have been pre-boundto the JNDI path **`mail/DefaultMailSession`**, enter this pathin the ***JNDI Name*** property.

If, at this time, you do not define the JNDI Name as shown above,you will have to bind the application's mail resource references to this mail sessionwhen you install ***jmsample.ear***.

Review the tutorial, Create a JavaMail session,for detailed information on configuring the mail session resource.

3. Install the *jmsample.ear* components

Install the ***jmsample.ear*** file as an enterprise application.See article Installing EJB modules with the JAVA administrative consolefor information on installing the `jmsample.ear`.

After the install, you can invoke the servlet or JSP by using one of the following URLs:

`http://localhost:9080/jmsample/servlet`
`http://localhost:9080/jmsample/Email.jsp`

To test the JavaMail servlet, do the following:

1. Open a browser window
2. Enter `http://localhost:9080/jmsample/servlet`
   (The servlet GUI should display.)
3. Enter the following information, replacing variable input as appropriate:
   - **To:** Your e-mail address as for example, *anybody@mycompany.com*
   - **Cc:** Optionally enter another e-mail address here.
   - **From:** *somebody@mycompany.com*
   - **Subject:** *JavaMail Servlet Test*
   - **Message to send:** Any text as for example, *This is a test message that isbeing sent from the JavaMail Mail Servlet*.
   - Check the **Save the sent message into mail store** box.
   - **Send Option:** As is
   - Click the **Send** button.
     (If the test number at the top of the page was incremented from **1** to **2**, the message was sent successfully.)
4. Log on to your e-mail account and verify that you received the mail

To test the JavaMail Java Server Page, do the following:

1. Open a browser window
2. Enter `http://localhost:9080/jmsample/Email.jsp`
   (The Java Server Page GUI should display.)
3. Enter the following information, replacing variable input as appropriate:
   - **To:** Your e-mail address as for example, *anybody@mycompany.com*
   - **Cc:** Optionally enter another e-mail address here.
   - **From:** *somebody@mycompany.com*
   - **Subject:** *JavaMail JSP Test*
   - **Message to send:** Any text as for example, *This is a test message that isbeing sent from the JavaMail Mail JSP*.
   - Check the **Save the sent message into mail store** box.
   - **Send Option:** As is
   - Click the **Send** button.
     (If the test number at the top of the page was incremented from **1** to **2**, the message was sent successfully.)
4. Log on to your e-mail account and verify that you received the mail

Use these instructions to invoke the EJB:

A. Locate file ***deplmtest.jar*** in the *product_installation_root*/InstalledApps/jmsample.ear directory.

B. Copy the ***deplmtest.jar*** file to the *product_installation_root*/classes directory.

The ***deplmtest.jar*** file contains all the artifacts for the EJB, including the proxies and a simple client.

C. The client uses the system properties as data input forthe message creation.The following example demonstrates how Java system properties are gathered for the EJB client on Windows NT:

The line breaks in this example were added to make the information more legible. This information really exists as one line of input.

Change the property values in this example to the ones you defined for your test.

For the Windows platform, specify the following:

```
product_installation_root\java\bin\java
-Djava.ext.dirs=product_installation_root\java\jre\lib\ext;product_installation_root\classes;product_installation_root\lib-Djava.naming.factory.initial=com.ibm.websphere.naming.WsnInitialContextFactory-Dmailtest.to=bob@mycompany.com
-Dmailtest.cc=alice@mycompany.com-Dmailtest.from=john@mycompany.com -Dmailtest.subj="Important message sent from an
EJB"-Dmailtest.message="As the subject line says, this is a very important message." -Dmailtest.save_msg=off
-Dmailtest.ejbhome=ejb/JMSampEJB  mailtest.MailClient
```

For UNIX platforms, specify the following:

ℹ Entering the following command as one, continuous lineof input at a command prompt might not work on some UNIX platforms. Use the back slashto indicate the command continues on the next line, or invoke the command from a shell script.

```
product_installation_root/java/bin/java
-Djava.ext.dirs=product_installation_root/java/jre/lib/ext:product_installation_root/classes:product_installation_root/lib-Djava.naming.factory.initial=com.ibm.websphere.naming.WsnInitialContextFactory-Dmailtest.to=bob@mycompany.com
-Dmailtest.cc=alice@mycompany.com-Dmailtest.from=john@mycompany.com -Dmailtest.subj="Important message sent from an
EJB"-Dmailtest.message="As the subject line says, this is a very important message." -Dmailtest.save_msg=off
-Dmailtest.ejbhome=ejb/JMSampEJB  mailtest.MailClient
```

See the related topics for links to Javadoc and the *Create a JavaMail session* tutorial.

4.6.2: JNDI (Java Naming and Directory Interface) overview

Distributed computing environments often employ namingand directory services to obtain shared components and resources.Naming and directory services associate names with locations, services, information, and resources.

Naming services provide name-to-object mappings. Directory services provideinformation on objects and the search tools required to locate those objects.There are many naming and directory service implementations, and theinterfaces to them vary.

Java Naming and Directory Interface or JNDI provides a common interface thatis used to access the various naming and directory services.See URL java.sun.com/products/jndi/serviceproviders.htmlfor a list of naming and directory service providers which supportaccess through the JNDI interface.

JNDI is an integral part of other Java programming models and technologies, such as:

- Enterprise JavaBeans (EJB)
- JavaMail
- Java Database Connection Service (JDBC)
- Java Message Service (JMS)

4.6.2.1: JNDI implementation in WebSphere Application Server

IBM WebSphere Application Server includes a name server to provideshared access to Java components, and an implementation of the`javax.naming` JNDI package which allows users to accessthe WebSphere name server through the JNDI naming interface.

WebSphere Application Server does ***not*** provide implementations for:

- `javax.naming.directory` or
- `javax.naming.ldap` packages

Also, WebSphere Application Server does ***not*** support interfaces defined in the

`javax.naming.event` package.

However, to provide access to LDAP servers, the JDK shipped with WebSphere Application Server supports Sun's implementation of:

- `javax.naming.ldap` and
- `com.sun.jndi.ldap.LdapCtxFactory`

WebSphere Application Server's JNDI implementation is based on version 1.2 of the JNDIinterface, and was tested with version 1.2.1 of Sun's JNDI SPI (Service Provider Interface).

The default behavior of this JNDI implementation should be adequatefor most users. However, users with specific requirements cancontrol certain aspects of the JNDI behavior. See the following section for information on modifying the JNDI behavior:

- JNDI caching - Description of the caching feature and properties, andthe effects of the different properties on caching behavior.

4.6.2.2: Using JNDI

Refer to these examples to learn how to use JNDI.

- Get an initial context
- Get an initial context using JNDI properties found in the current environment
- Get an initial context by explicitly setting JNDI properties
- Look up a home for an EJB
- Look up a JavaMail session

Get an initial context

In general, JNDI clients should assume the correct environment isalready configured so there is no need to explicitly set property values and passthem to the **`InitialContext`** constructor. However, a JNDI client may needto access a name space other than the one identified in its environment.In this event, it is necessary toexplicitly set one or more properties used by the **`InitialContext`** constructor. Anyproperty values passed in directly to the **`InitialContext`** constructor take precedenceover settings of those same properties found elsewhere in the environment.

View the following examples for information on passing property values to the **`InitialContext`** constructor:

- Get an initial context using JNDI properties found in the current environment:

  The current environment includes the Java system properties and properties defined inproperties files found in the JNDI client's CLASSPATH. See article Installing files and setting classpathsfor information on defining CLASSPATHs.

  ```
     ...   import javax.naming.Context;   import javax.naming.InitialContext;   ...   Context
  initialContext = new InitialContext();   ...
  ```

- Get an initial context by explicitly setting JNDI properties:

  ```
     ...   import java.util.Hashtable;   import javax.naming.Context;   import
  javax.naming.InitialContext;   ...   Hashtable env = new Hashtable();
  env.put(Context.INITIAL_CONTEXT_FACTORY, "com.ibm.websphere.naming.WsnInitialContextFactory");
  env.put(Context.PROVIDER_URL, "iiop://myhost.mycompany.com:900");   Context initialContext = new
  InitialContext(env);   ...
  ```

- Look up a home for an EJB

  The example below shows a lookup of an EJB home. The actual home lookup nameis determined by the application's deployment descriptors.

  ```
     // Get the initial context as shown in the previous example   ...   // Look up the home interface
  using the JNDI name   try {      java.lang.Object ejbHome =
  initialContext.lookup("java:comp/env/comp/mycompany/accounting");      accountHome =
  (AccountHome)javax.rmi.PortableRemoteObject.narrow(         (org.omg.CORBA.Object) ejbHome,
  AccountHome.class);   }   catch (NamingException e) { // Error getting the home interface      ...
  }
  ```

- Look up a JavaMail session:

  The example below shows a lookup of a JavaMail resource. The actual lookup nameis determined by the application's deployment descriptors.

  ```
     // Get the initial context as shown above   ...   Session session = (Session)
  initialContext.lookup("java:comp/env/mail/MailSession");
  ```

4.6.2.3: JNDI caching

To increase the performance of JNDI operations, WebSphere Application Server'sJNDI implementation employs caching to reduce the number of remote calls to thename server. For most cases, use the default cache setting.

JNDI context objects employ caching to increase the performanceof JNDI lookup operations. Objects bound and looked up are cached in orderto speed up subsequent lookups of those objects. Objects are cached asthey are bound or initially looked up. Normally, JNDI clients should beable to use the default cache behavior. The following sectionsdescribe in detail cache behavior, and how JNDI clients can override defaultcache behavior if necessary.

- Cache behavior
- Cache properties
- Coding examples

Cache behavior

A cache is associated withan initial context when a `javax.naming.InitialContext` object is instantiatedwith the `java.naming.factory.initial` property set to:

`com.ibm.websphere.naming.WsnInitialContextFactory`

`WsnInitialContextFactory` searches the environment properties for a cachename, defaulting to the provider URL. If no provider URL is defined, acache name of "iiop:///" is used. All instances of InitialContext whichuse a cache of a given name share the same cache instance.

After an associationbetween an InitialContext instance and cache is established, the associationdoes not change. A `javax.naming.Context` object returned from a lookup operationwill inherit the cache association of the Context object on which the lookupwas performed. Changing cache property values with the `Context.addToEnvironment()`or `Context.removeFromEnvironment()` method does not affect cache behavior.Properties affecting a given cache instance, however, may be changed witheach InitialContext instantiation.

A cache is restricted toa process and does not persist past the life of the process. A cached objectis returned from lookup operations until either the max cache life forthe cache is reached, or themax entry life for the object's cache entryis reached.

After this time, a lookup on the object will cause the cacheentry for the object to be refreshed. If a bind or rebind operation isexecuted on an object, the change will not be reflected in any caches otherthan the one associated with the context from which the bind or rebindwas issued. This "stale data" scenario is most likely to happen when multipleprocesses are involved, since different processes do not share the samecache, and Context objects in all threads in a process will typically sharethe same cache instance for a given name service provider.

Usually, cached objects are relatively static entities, and objects becoming staleshould not be a problem. However, timeout values can be set on cache entriesor on a cache itself so that cache contents are periodically refreshed.

Cache properties

JNDI clients can use several properties to control cache behavior.These properties can be set in theenvironment Hashtable passed to the InitialContext constructor.

You can set properties:

    - From the command line
    - In a properties file
    - Within a Java program

- To set properties through the command line,enter the actual string value as indicated in this example:

  ```
  java -Dcom.ibm.websphere.naming.jndicache.maxentrylife=1440
  ```

- To set properties in a file,create a text file listing the properties, as for example:

  ```
  ...   com.ibm.websphere.naming.jndicache.cacheobject=none   ...
  ```

- To set properties in a Java program,use the following **PROPS.JNDI_CACHE*** Java constants, defined in ***com.ibm.websphere.naming.PROPS***:

  ```
      public static final String JNDI_CACHE_OBJECT =
  "com.ibm.websphere.naming.jndicache.cacheobject";    public static final String
  JNDI_CACHE_OBJECT_NONE      = "none";    public static final String JNDI_CACHE_OBJECT_POPULATED =
  "populated";    public static final String JNDI_CACHE_OBJECT_CLEARED   = "cleared";    public static
  final String JNDI_CACHE_OBJECT_DEFAULT   = JNDI_CACHE_OBJECT_POPULATED;    public static final
  String JNDI_CACHE_NAME = "com.ibm.websphere.naming.jndicache.cachename";    public static final
  String JNDI_CACHE_NAME_DEFAULT = "providerURL";    public static final String JNDI_CACHE_MAX_LIFE =
  "com.ibm.websphere.naming.jndicache.maxcachelife";    public static final int
  JNDI_CACHE_MAX_LIFE_DEFAULT = 0;    public static final String JNDI_CACHE_MAX_ENTRY_LIFE =
  "com.ibm.websphere.naming.jndicache.maxentrylife";    public static final int
  JNDI_CACHE_MAX_ENTRY_LIFE_DEFAULT = 0;
  ```

  To set a property in your program, enter the following:

  ```
  env.put(PROPS.JNDI_CACHE_OBJECT, PROPS.JNDI_CACHE_OBJECT_NONE).  // Sets a property to a value
  ```

Cache properties are evaluated when an InitialContext instance iscreated. The resulting cache association, including"none", cannot bechanged. The "max life" cache properties affectthe individual cache's behavior. If the cache already exists, cache behavior will beupdated according to the new "max life" property settings. If no"max life" properties exist in the environment, the cachewill assume default "max life"settings, irrespective of the previous settings.The various cache properties are describedbelow. All property values must be string values.

- **com.ibm.websphere.naming.jndicache.cacheobject**

  Caching is turned on or off with this property. Additionally, an existingcache can be cleared.Listed below are the valid values for this property and the resulting cachebehavior:

  - "**populated**" (default): Use a cache with the specified name. If the cache already exists, leave existing cache entries in cache; otherwise, create a new cache.
  - "**cleared**": Use a cache with the specified name. If the cache already exists, clear all cache entries from cache; otherwise, create a new cache.
  - "**none**": Do not cache. If this option is specified, the cache name is irrelevant. Therefore, this option will not disable a cache that is already associated with other InitialContext instances. The InitialContext being instantiated will not be associated with any cache.

- **com.ibm.websphere.naming.jndicache.cachename**

  It is possible to createmultiple InitialContext instances, each operating on the namespace of adifferent name service provider. By default, objects from each serviceprovider are cached separately, since they each involve independent namespacesand name collisions could occur if they used the same cache. The providerURL specified when the initial context is created serves as the defaultcache name. With this property, a JNDI client can specify a cache nameother than the provider URL. Listed below are the valid options forcache names:

  - "**providerURL**" (default): Use the value for java.naming.provider.url property as the cache name. The default provider URL is "iiop:///". URLs are normalized by stripping off everything after the port. For example, "iiop://server1:900" and "iiop://server1:900/com/ibm/initCtx" are normalized to the same cache name.
  - ***Any string***: Use the specified string as the cache name. Any arbitrary string with a value other than "providerURL" can be used as a cache name.

- **com.ibm.websphere.naming.jndicache.maxcachelife**

  By default, cached objects remain in the cache for the life of the process oruntil cleared with the com.ibm.websphere.naming.jndicache.cacheobject propertyset to "cleared". This property enables a JNDI client to set the maximum lifeof a cache as follows:

  - "**0**" (default): Make the cache lifetime unlimited.
  - ***Positive integer***: Set the maximum lifetime of the cache, in minutes, to the specified value. When the maximum cache lifetime is reached, the cache is cleared before another cache operation is performed. The cache is repopulated as bind, rebind, and lookup operations are executed.

- **com.ibm.websphere.naming.jndicache.maxentrylife**

  By default, cached objects remain in the cache for the life of the processor until cleared with the com.ibm.websphere.naming.jndicache.cacheobjectproperty set to "cleared". This property enables a JNDI client to set themaximum lifetime of individual cache entries as follows:

  - "**0**" (default): Lifetime of cache entries is unlimited.
  - ***Positive integer***: Set the maximum lifetime of individual cache entries, in minutes, to the specified value. When the maximum lifetime for an entry is reached, the next attempt to read the entry from the cache will cause the entry to be refreshed.

Coding examples

```
import java.util.Hashtable;import javax.naming.InitialContext;import javax.naming.Context;/*****
Caching discussed in this section pertains to the WebSphere Application Server initial context
factory. Assume the property, java.naming.factory.initial, is set to
"com.ibm.ejs.ns.WsnInitialContextFactory" as a java.lang.System property.*****/Hashtable env;Context
ctx;// To clear a cache:env = new Hashtable();env.put(PROPS.JNDI_CACHE_OBJECT,
PROPS.JNDI_CACHE_OBJECT_CLEARED);ctx = new InitialContext(env);// To set a cache's maximum cache
lifetime to 60 minutes:env = new Hashtable();env.put(PROPS.JNDI_CACHE_MAX_LIFE, "60");ctx = new
InitialContext(env);// To turn caching off:env = new Hashtable();env.put(PROPS.JNDI_CACHE_OBJECT,
PROPS.JNDI_CACHE_OBJECT_NONE);ctx = new InitialContext(env);// To use caching and no caching:env =
new Hashtable();env.put(PROPS.JNDI_CACHE_OBJECT, PROPS.JNDI_CACHE_OBJECT_POPULATED);ctx = new
InitialContext(env);env.put(PROPS.JNDI_CACHE_OBJECT, PROPS.JNDI_CACHE_OBJECT_NONE);Context
noCacheCtx = new InitialContext(env);Object o;// Use caching to look up home, since the home should
rarely change.o = ctx.lookup("com/mycom/MyEJBHome");// Narrow, etc. ...// Do not use cache if data
is volatile.o = noCacheCtx.lookup("com/mycom/VolatileObject");// ...
```

4.6.2.4: JNDI helpers and utilities

Refer to the Sun JNDI specificationfor information on the base JNDI APIs. IBM WebSphere Application Server provides the following JNDI extension and utility to help you implement and debug JNDI.

View the specific files for details.

- JNDI helper class
- Name Space Dump utility

4.6.2.4.1: JNDI helper class

The class **`com.ibm.websphere.naming.JndiHelpers`** contains static methodsto simplify common tasks. Refer to theAPI documentationfor more information.

JNDI helper methods provide assistance with:

- *Recursively creating subcontexts.*

  ```
      [...]    import com.ibm.websphere.naming.JndiHelper;    [...]    try {       Context
  startingContext = new InitialContext();       startingContext =
  startingContext.lookup("com/mycompany");       // Creates each intermediate subcontext, if
  necessary, as well as leaf context.       // AlreadyBoundException is not thrown.
  JndiHelper.recursiveCreateSubcontext(startingContext, "apps/accounting");       }    catch
  (NamingException e)       // Handle  error.    }    [...]
  ```

- *Rebinding objects and creating intermediate contexts that do not already exist.*

  ```
      [...]    import com.ibm.websphere.naming.JndiHelper;    [...]    try {       Context
  startingContext = new InitialContext();       // Creates each intermediate subcontext, if necessary,
  and rebinds object.      JndiHelper.recursiveRebind(startingContext,
  "com/mycompany/apps/accounting", someObject);       }    catch (NamingException e)       // Handle
  error.    }    [...]
  ```

- *Binding objects and throwing a **NameAlreadyBoundException** if the object is already bound.*

  There are two versions of this JndiHelper method:

  ```
      public static void recursiveBind(Context startingContext, Name name, Object obj)    public
  static void recursiveBind(Context startingContext, String name, Object obj)
  ```

  ```
      [...]    import com.ibm.websphere.naming.JndiHelper;    [...]    try {       Context
  startingContext = new InitialContext();       // Creates each intermediate subcontext, if necessary,
  and binds object.      JndiHelper.recursiveBind(startingContext, "com/mycompany/apps/accounting",
  someObject);    }    catch (NamingException e)       // Handle error.    }          }    catch
  (Exception e)       // Handle other errors.    }    [...]
  ```

4.6.2.4.2: JNDI Name Space Dump utility

The name space stored by a given name server can be dumped withthe name space dump utility that is shipped with WebSphere Application Server.This utility can be invoked from the command line or from a Java program. Thenaming service for the WebSphere Application Server host must be active when this utilityis invoked.

To invoke this utility using the **`class com.ibm.websphere.naming.DumpNameSpace`** API, see the API documentation.

To invoke the utility through the command line, enter the following commandfrom the ***AppServer/bin*** directory:

UNIX: **dumpNameSpace.sh** [[-*keyword value*]...]

Windows NT: **dumpNameSpace** [[-*keyword value*]...]

The keywords and associated values for the **dumpNameSpace** utility are:

-host *myhost.austin.ibm.com*

Represents the bootstrap host or the WebSphere Application Server host whose name space youwant to dump. The value defaults to ***localhost***.

-port *nnn*

Represents the bootstrap port which, if not specified, defaults to **900**.

-factory *com.ibm.websphere.naming.WsnInitialContextFactory*

Indicates the initial context factory to be used to get the JNDI initialcontext. The value defaults to:
com.ibm.websphere.naming.WsnInitialContextFactory
The default value generally does not need to be changed.

-startAt *some/subcontext/in/the/tree*

Indicates the path from the bootstrap host's root context to the toplevel context where the dump should begin. The utility recursively dumps subcontextsbelow this point. It defaults to an empty string, that is, the bootstrap host root context.

-format {**jndi** | **ins**}

jndi Displays name components as atomic strings.

ℹ The default format is **jndi**.

ins Displays name components parsed per INS rules (id.kind).

-report {**short** | **long**}

short Dumps the binding name and bound object type. This output is also provided by JNDI Context.list().

ℹ The default report option is **short**.

long Dumps the binding name, bound object type, local object type, and stringrepresentation of the local object (that is, the IORs, string values, and other values that are printed).

For objects of user-defined classes to display correctly with the long report option, it may be necessary to add their containing directories to the list of directories searched. This can be done by setting the environment variable **WAS_USER_DIRS**.The value can include one or more directories, as for example:

UNIX:

```
WAS_USER_DIRS=/usr/classdir1:/usr/classdir2       export WAS_USER_DIRS
```

Windows NT:

```
set WAS_USER_DIRS=c:\classdir1;d:\classdir2
```

All zip, jar, and class files in the specified directories can then be resolved bythe class loader when running **dumpNameSpace**

-traceString *"some.package.name.to.trace.*=all=enabled"*

Represents the trace string with the same format as that generated by the servers. The outputis sent to file, **DumpNameSpaceTrace.out**.

-help

Provides a description of Name Space Dump utility and command line usage.

Examples of Name Space Dump utility usage and output

- Invoke the name space dump utility through a Java program.
- Invoke the name space dump utility through the command line.
- View the name space dump utility output.

- Invoke the name space dump utility by adding the following code to your Java program:

```
{   [...]   java.io.PrintStream filePrintStream = ...   Context ctx = new InitialContext();   ctx =
(Context) ctx.lookup("ejsadmin/node");   // Starting context for dump   DumpNameSpace dumpUtil = new
DumpNameSpace(filePrintStream, DumpNameSpace.SHORT);   dumpUtil.generateDump(ctx);   [...]}
```

- Invoke the name space dump utility from the command line by entering the following command:

```
dumpNameSpace -host myhost.mycompany.com -port 901
```

- The generated output will look like the following example, which isthe **SHORT** dump format:

```
Getting the initial contextGetting the starting
context====================================================================================Name
Space Dump   Provider URL: iiop://will:901   Context factory:
com.ibm.websphere.naming.WsnInitialContextFactory   Starting context: (top)=bootstrap host root
context   Formatting rules: jndi   Time of dump: Fri Mar 09 15:11:48 CST
2001====================================================================================
====================================================================================Beginning of
Name Space Dump====================================================================================
1 (top)                                                  2 (top)/jta
javax.naming.Context    3 (top)/jta/usertransaction
com.ibm.ejs.jts.jta.UserTransactionImpl    4 (top)/SecurityCurrent
com.ibm.ejs.security.util.SecurityCurrentRef    5 (top)/ContextHome
com.ibm.ejs.ns.CosNaming.EJSRemoteContextHome    6 (top)/PropertyHome
com.ibm.ejs.ns.CosNaming.EJSRemotePropertyHome    7 (top)/BindingHome
com.ibm.ejs.ns.CosNaming.EJSRemoteBindingHome    8 (top)/will
javax.naming.Context    9 (top)/will/resources                               javax.naming.Context
10 (top)/will/resources/sec                           javax.naming.Context   11
(top)/will/resources/sec/SecurityServer            com.ibm.WebSphereSecurityImpl.SecurityServerImpl
12 (top)/ejsadmin                                     javax.naming.Context   13 (top)/ejsadmin/node
javax.naming.Context   14 (top)/ejsadmin/node/will                           javax.naming.Context
15 (top)/ejsadmin/node/will/homes                     javax.naming.Context   16
(top)/ejsadmin/node/will/homes/DeployEJBHome       com.ibm.ejs.sm.tasks.EJSRemoteDeployEJBHome   17
(top)/ejsadmin/node/will/homes/ServletEngineHome
com.ibm.ejs.sm.beans.EJSRemoteServletEngineHome[etc.]
====================================================================================End of Name
Space Dump====================================================================================
```

4.6.3: Java Message Service (JMS) overview

IBM WebSphere Application Server supports messaging as a method of communication based on the Java MessageService programming interface.

Unlike JavaMail that enables communication initiated by people or by software components to people, Java Message Service (or JMS)only provides communication between software components and applications. Communication provided by JMS is *loosely coupled*, which meansthe sender and receiver do not have to be active or aware of each other. The communication is also *asynchronous*. This meansclients do not have to request messages from the JMS provider in order to receive them, and software components can send messages to other components without stopping their processes to wait for a response.

In this peer-to-peer communication system, each client connects to a messaging agent that provides the framework for sending and receiving messages.The client is required to know only the following:

- message format
- destination of the message

There are two approaches to messaging:

- Point-to-point
- Publish/subscribe

The *point-to-point* messaging approach uses such facilities as message queues, senders (or message producers), and receivers (or message consumers). Clients send messages that are destined for a specific receiver to a unique queue. When the receiving client extracts a message from the specific queue, it sends an acknowledgement indicating the message was processed. Queues hold all messages until the messages are received or until they expire.

The *publish/subscribe* messaging approach uses the concepts of publishers, subscribers, and topics. Clients send messages to a topic or a content hierarchy.In order to receive the message, the message consumers must subscribe to that topic. So, in this approach,the message producers are known as publishers and the message consumers are known assubscribers. The JMS provider distributes the messages sent from the multiple publishers to the topic, to the multiplesubscribers of that topic.

The MQSeries product is the default JMS provider for WebSphere Application Server. The MQSeries administration tool, **JMSAdmin**, is used to bind JMS objects (connection factories and destinations) into the namespace, and to set their properties.

WebSphere Application Server Enterprise Edition Version 4.0 also provides the *JMS Listener* function. Similar to an event listener, the JMS Listenerenables WebSphere Application Server to react to anonymous, incoming JMS messagesby invoking an appropriate enterprise java bean. The invoked enterprise bean is a stateless session bean with an`onMessage()` method.

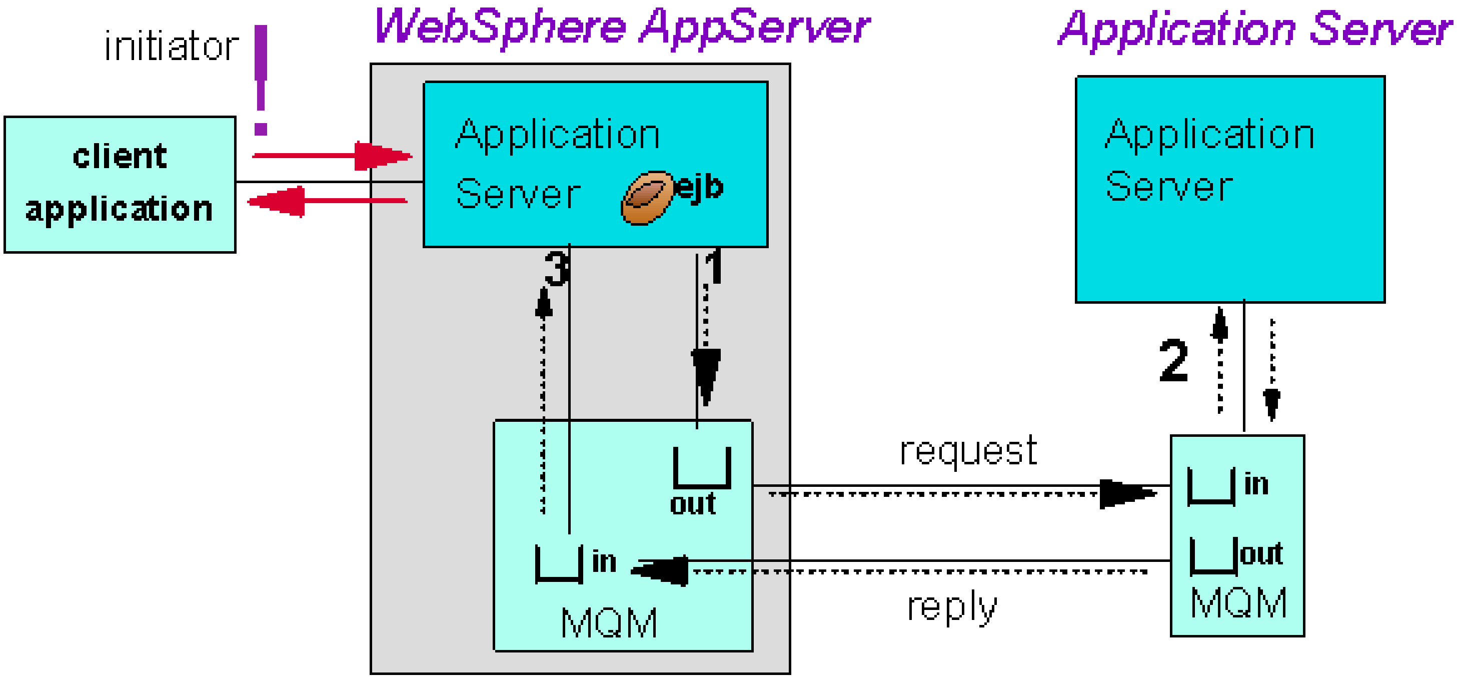
4.6.3.1: Using the JMS point-to-point messaging approach

This article describes the point-to-point messaging approach using WebSphere ApplicationServer's default JMS provider, MQSeries. The MQSeries messaging server implements *point-to-point* communication. To enable the MQSeries point-to-point messaging support, you need:

- MQSeries Version 5.2 or greater
- With MQSeries Version 5.2, you need MQSeries SupportPac MA88

MQSeries can now act as a resource manager in application transactions, and WebSphere Application Server can act as the transaction coordinator.For example, when a client application sends a request, WebSphere Application Server, using MQSeries, puts the message on an *out*queue and waits for a response to return to the *in* queue.In this scenario, there is no guarantee the message was sent, or that the receiver received the message. These types of messages are known as *non persistent*messages.

The *point-to-point* messaging approach in WebSphere Application Server and MQSeriesis illustrated in the following graphic:



Message delivery can be defined as:

- *Persistent* - this is the default mode of delivery. The "message send" is logged into stable storage.
- *Non persistent* - message delivery is not guaranteed. This mode of delivery improves performance and reduces storage overhead.

Message delivery properties can be set:

- On the queue within the queue manager
- On the queue object using the JMSAdmin tool
- On individual messages within your JMS application

To define a queue in the JNDI namespace and to set thepersistence properties for the queue,enter the following command in the MQSeries JMSAdmin tool:

```
InitCtx> DEFINE Q(TESTQ) PERSISTENCE(xxx)
```

Where **xxx** is one of the following:

APP	(Default) Persistence is defined by the application
QDEF	Persistence is defined by the queue default (Set in the queue manager)
PERS	Messages are persistent
NON	Messages are non-persistent

If your application sends a message and requires a reply, set a reasonable timeoutvalue in your application to handle a delayed or "no" reply situation. The followingapplication code waits for a maximum of 5000 milliseconds:

```
        Message inMessage = queueReceiver.receive(5000);
```

Set a similar timeout in your reply message.

Transactions to MQSeries are boundary transactions not end-to-end transactions.This means that only a *put* to a queue, or a *get* from a queue is part of the transaction. The flow to a remote application is notpart of the transaction. In order to guarantee the message is received bythe remote application, define that message in the JMSAdmin tool as a *persistent* message.

WebSphere enterprise applications can use the JMS Listener function to automaticallyreceive messages from input queues (JMS destinations) and to coordinate the processing of those messages. This enables automatic asynchronousdelivery of messages to an enterprise application, instead of the applicationhaving to explicitly poll for messages on the queue. For more information on the JMS Listener function, see An overview of the JMS Listener.

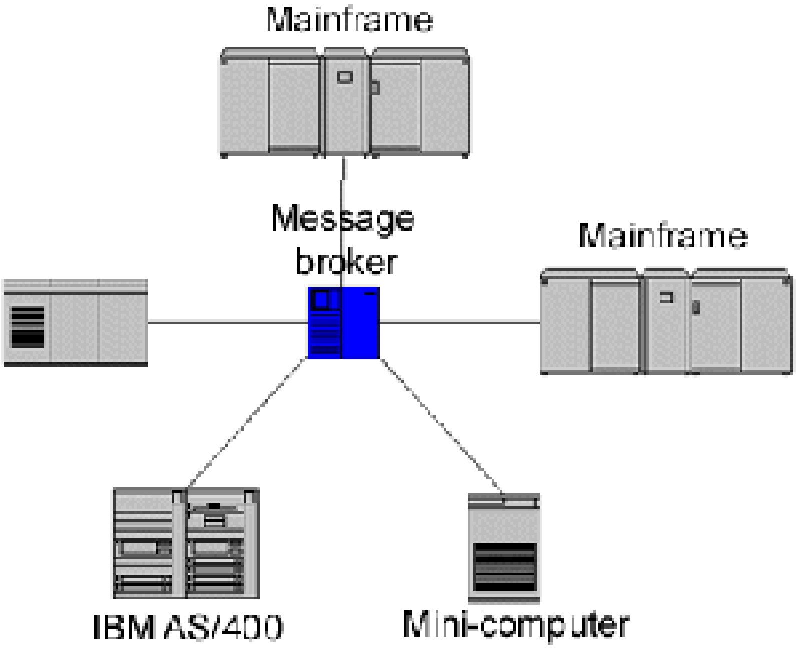
> The link to the JMS Listener documentation will not work unless the WebSphere Application Server Enterprise Edition product extensions are installed on your system.

See the Support of the MQSeries Java Message Service resources article for configuration information.

4.6.3.2: Using the JMS publish/subscribe messaging approach

This article describes the "publish/subscribe" messaging approach using WebSphere ApplicationServer's default JMS provider, MQSeries. You can implement the "publish/subscribe"messaging approach in MQSeries with the Pub-Sub SupportPac or with Integrator.

To alleviate the complexity of a multiple queue manager topology, MQSeries introduced the concept of Message Brokers with the MQSeries Integrator product. The following graphic illustrates five queue managers configured to use a Message Broker:



In addition to the Message Broker, the Integrator product also supportsa Message Repository Manager, and the *publish and subscribe* messaging approach.With this approach, the Message Broker matches a topic on a published message with a list of clients who have subscribed to that topic. Neither publisher nor subscriber is aware of each other. Publishers only know of the topics they describe for their messages, and subscribers only know of the topicsthey requested.

In this topology, WebSphere Application Server can be a publisher or subscriber, or both, but requires the configuration and resource support of the MQSeries Integrator product.

Visit the MQSeries Integratorsite for more information.

4.6.3.3: Support of Java Message Service resources

Unlike other J2EE resources that are typically objects that run in and are part of the application server, JMS resources are external to WebSphere Application Server.This means administrators must first use a JMS provider's administration tool to create the connection factories and destinations, and to assign these objects with correctconfiguration attributes. After this step is completed, administrators can then use WebSphere Application Servers' administrative client to create JMS resource objects to reference the external objects.

In WebSphere Application Server Version 4.0, the MQSeries product is defined as thedefault JMS provider. However, since MQSeries is not shipped with WebSphere Application Server, this JMS provider is not installed on any node.
Since this provider is predefined, after you install the MQSeries product, you only need to go to the **Nodes** tab of theproperties editor for the Provider to install it on the desired nodes.

The following steps describe how to implement JMS support in WebSphere Application Server:

1. Configure a JMS provider. By default this will be the MQSeries product.
2. Create the destination and connection factory with the JMS provider's admin tool.
   This will bind references to these objects in the JNDI namespace.
3. Create corresponding JMS resources in WebSphere Application Server, declaring the location where they were bound by the JMS admin tool as an attribute.
   The `RepositoryObject`implementation for the resource binds the resource into the WebSphere Application Servernamespace.
4. Deploy the application, which resolves the JMS resource references with the JMS objects.
5. Start the application server containing the application.

   > At this point, the `ResourceBinder` object in the application server binds theJMS resource objects into the namespace.
6. The application code performs a "lookup" on a JMS resource.
   The "lookup" finds the `IndirectJNDILookup` bound at the target WebSphere Application Server location, and uses it to perform a subsequent lookup of the actualresource in the provider's namespace.

4.6.3.4: Support for the use of MQSeries Java Message Service resources

WebSphere Application Server Enterprise JavaBeans support the transactional use of MQSeries Java Message Service (JMS) resources.

To use this feature, install MQSeries version 5.2 and the MQSeries classes for Java and JMS.Only MQSeries V5.2 provides this support; earlier versions will not work.

To configure JMS resources for use with WebSphere Application Server:

1. Download the MQSeries Java and JMS classes, or the pub-sub packagefrom one of the following URLs:
   - http://www.ibm.com/software/ts/mqseries/api/mqjava.html
   - http://www.ibm.com/software/ts/mqseries/txppacs/ma0c.html
2. Review the MQSeries Using Java book for a description of the parameters required for WebSphere Application Server.

   The instructions in this book refer to a WebSphere Application Server Version 3.5.3 environment, and are not valid for WebSphere Application Server Version 4.0. For example, the following content in the book is invalid for Version 4.0:
   - Names of jar files
   - Dependent classpath
   - Adminserver classpath
3. Do the following to configure WebSphere Application Server and MQSeries for JMS support:
   a. Modify the `JMSAdmin.bat` file to include the option `-java.ext.dirs=<WS AE>\lib` when running the MQSeries administration tool, JMSAdmin.

   b. Modify the `JMSAdmin.config` file by uncommenting the following lines:
   `INITIAL_CONTEXT_FACTORY=com.ibm.ejs.ns.jndi.CNInitialContextFactory`
   `PROVIDER_URL=iiop://localhost/` (or `iiop://host-name`)

   c. *Comment out* the following lines in the `JMSAdmin.config` file:
   `INITIAL_CONTEXT_FACTORY=com.ibm.sun.jndi.fscontext.RefFSContextFactory`
   `PROVIDER_URL=file:/C:/JNDI-Directory/`

   d. Add the following to the application server classpath:

      You can add these classes in the console'sdefault JVM settings, or by editing WebSphere Application Server's `admin.config` file.
      - `<MQ JMS>\lib` directory
      - `com.ibm.mq.jar` file
      - `com.ibm.mqjms.jar` file

      An alternative way to set up the configuration isto use the administrative console. In the Resources, JMS Providers folder, specifythe ContextFactory, the provider URL, and the path to the MQSeries JMS `.jar` files.See the article about administering JMS support resources for more information.
4. Bind the classes provided by the new function into the JNDI namespace using the MQSeries administration tool, JMSAdmin.The JMSAdmin tool provides for two, new WebSphere Application Server JMS connection factories:
   - **WSQCF -** a new type of queue connection factory
   - **WSCTF -** a new type of topic connection factory

WebSphere Application Server connection factory objects

The following calls can be used either in a global transactionor in an unspecified transaction context:

- QueueSender.send
- MessageConsumer.receive
- MessageConsumer.receiveNoWait
- TopicPublisher.publish

If another resource manager, as for example JDBC, is involved in a globaltransaction, the MQSeries JMS resources are involved in a 2-phase commit.The 1-phase commit occurs if only the JMS resources are involved in a global transaction. This is a feature of the Transaction Manager optimization.

In a global transaction, messages sent with `QueueSender.send` or published with `TopicPublisher.publish` do not become visible until thetransaction is committed. Messages received by `MessageConsumer.receive`or `MessageConsumer.receiveNoWait` are requeued if the transaction isrolled back. Both bean-managed transaction demarcation and container-managedtransaction demarcation are supported.

If no global transaction is active, then an "unspecified transaction context"situation occurs. The following circumstances cause an "unspecified transaction context:"

- EJB methods when a global transaction cannot occur (for example, `ejbCreate`)
- Bean Managed Transaction methods where the bean writer chose not to begin a transaction
- Container Managed Transaction `NOT_SUPPORTED` or `NEVER` methods
- Container Managed Transaction `SUPPORTS` methods when no transaction exists

In an unspecified transaction context, the transactional behavior is specified in thetransacted flag that is passed when the session is created. If the transacted flag is setto *false*, the messaging operations occur immediately. This is also known as the 0-phase commit.If the flag is set to *true*, the send, receive, and publish operations occur on the commit of the session,or also known as the 1-phase commit.

A summary of the transactional behavior for objects created on WSQCF or WSTCF is described in the following table:

	Global transaction context	**Unspecified transaction context**
transacted=**false**	2-phase commit	0-phase commit
transacted=**true**	2-phase commit	1-phase commit

To commit or to roll back the messaging work done on a transacted session, call method `session.commit()`or `session rollback()`.First check whether `session.getTransacted()` returns **true** before committing the session.`Session.getTransacted()` returns **true** if:

- The user passed in **true** as the transacted parameter when the session was created, and
- No global transaction is active at the moment of the call.

If both tests are met, you can commit the session. Trying to commit a session when a global transaction is active will result in the JMS exception, `IllegalStateException`, being thrown.

JMS XA support in WebSphere Application Server is integrated with local transactions.For container managed transactions, an "unspecified transaction context" causes WebSphereApplication Server to start a local transaction.In Version 4.0, the scope of the local transaction is the EJB method.The action taken at the end of the EJB method (commit or rollback of the local transaction)depends on the information contained in the deployment descriptor. This information is aWebSphere Application Server extension. The transaction manager will commit or rollback anyoutstanding, uncommitted work done within the local transaction without any user intervention. The default is to roll back.

Any work performed on a JMS session in an unspecified transaction context, will be rolled back or committed if the corresponding local transaction is rolled back or committed.

Requestors are only used with non-transacted sessions. Therefore, *QueueRequestor*and *TopicRequestor* cannot be used with sessions created by WebSphere ApplicationServer JMS connection factories.

Unsupported interfaces and methods

The following JMS interfaces are not designed for application use and, therefore,cannot be invoked:

Unsupported interfaces
javax.jms.ServerSession
javax.jms.ServerSessionPool
javax.jms.ConnectionConsumer
all the *javax.jms.XA* interfaces

The following JMS methods are inappropriatein this environment and interfere with connection management by the container. Therefore, these methods cannot be used:

Unsupported methods
javax.jms.Connection.setExceptionListener
javax.jms.Connection.stop
javax.jms.Connection.setClientID
javax.jms.Connection.setMessageListener
javax.jms.Session.getMesssageListener
javax.jms.QueueConnection.createConnectionConsumer
javax.jms.TopicConnection.createConnectionConsumer
javax.jms.TopicConnection.createDurableConnectionConsumer
javax.jms.MessageConsumer.setMessageListener

All the above methods throw the JMS exception, `IllegalStateException`, when invoked.

You cannot register a *MessageListener*with a *QueueReceiver* or *TopicSubscriber*.

The following methods throw the JMS exception, `IllegalStateException`, if used within a global transaction:

javax.jms.Session.commit
javax.jms.Session.rollback
javax.jms.Session.recover

With the Enterprise JavaBeans programming model, you must ensure all JMS resources are closed correctly.Since JMS resources never time-out, JMS resources that are not closed correctly will continue to consume MQSeries resources.The MQSeries resources also persist until the application server or MQSeries Queue manager is restarted.

Restrictions

The following restrictions exist regarding the use of JMS XA support in WebSphere Application Server:

- A subscriber can only be used in the same type of transactional context (for example, a global transaction or anunspecified transaction context) as the one that existed when the subscriber was created.

  If this restriction is not respected, the JMS exception, `subscriber restriction`, is thrown.

  If a global transaction is active at the creation of the subscriber, that subscriber can be used to receive messages in different global transactions, but not in an unspecified transaction context.

  If an unspecified transaction context is active when the subscriber is created, that subscriber cannotbe used with a global transaction.

- The use of JMS sessions across methods with different transactional attributes is restricted.

  If a session was used within a global transaction, it cannot be reused in a differentglobal transaction or in an unspecified context until the first transaction commits.Similarly, if there is work outstanding in a local transaction then the session cannot be used in a global transaction until the local transaction has finished.Session use, in this case, refers to the send, receive, and publish operations usingthe message producers or consumers that were created on the session.

6.6.37: Administering mail providers and mail sessions (overview)

WebSphere Application Server's implementation ofJavaMail does not provide mail servers. Even if your application components can communicatewith mail servers, you still must configure separate IMAP and SMTP serversto enable the mail functions.

Only the SMTP and IMAP service providers are shipped with WebSphere Application Server.To use other protocols, install the appropriate service providers for those protocols.

When you configure your mail servers, use fully qualified internet host names.

For information on how to install a service provider, see Chapter 5, *The Mail Session*, in theJava Mail API design specification.

The parameters used to configure a mail session resource can be dividedinto two groups:

1. mail send (transport) and
2. mail store access

If your enterprise application references a mail sessionresource, your application will use one of these functions:

- only mail-send
- both mail-send and mail-store access
- only mail-store access

> ℹ Use of the *only mail store-access* option is rare. As such, the store access part of the configuration is treated bySystem Management as optional while the *mail-send* function is treated as mandatory.

6.6.37.0: Properties of JavaMail providers

Key:

Applies to Java administrative console of Advanced Edition Version 4.0

Applies to Web administrative console of Advanced Single Server Edition Version 4.0

Applies to Application Client Resource Configuration Tool

Class Path

The path to the JAR file containing the implementation class for the JavaMail provider

Custom Properties

Name-value pairs for setting additional properties beyond those available in the administrative interface

Description

Optional description for your administrative records

Name

Administrative name of the JavaMail provider

6.6.37.0.1: Properties of JavaMail sessions

Key:

☕ Applies to Java administrative console of Advanced Edition Version 4.0

🌐 Applies to Web administrative console of Advanced Single Server Edition Version 4.0

C:\ Applies to Application Client Resource Configuration Tool

Category 🌐

Optional category for classifying this resource for your administrative records

Custom Properties C:\

Name-value pairs for setting additional properties

Confirm Password ☕ **or Re-Enter Password** C:\

Confirm the password that you entered in the preceding field

Description ☕ 🌐 C:\

Optional description for your administrative records

Enable Store 🌐

Enable JavaMail sessions

J2EE Resource Provider 🌐

The JavaMail provider with which this JavaMail session is associated

JNDI Binding Path ☕ **or JNDI Name** 🌐

The JNDI name for the resource, including any naming subcontexts. This name is used as the linkage between the platform's binding information for resources defined in the client applications deployment descriptor and actual resources bound into JNDI by the platform

JNDI Name 🌐 C:\

See JNDI Binding Path

Name ☕ 🌐 C:\

Administrative name of the JavaMail session object

Mail From 🌐 C:\ **or Outgoing Mail Originator** ☕

Value of replyTo field in mail messages sent through the mail transport host. The Internet email address that by default will bedisplayed in the received message as either the "From" or the "Reply-To" address. The recipient's reply will come to the specified address.

Mail Store Access Host ☕ **or Mail Store Host** 🌐 C:\

The server to which to connect when reading mail. This property combines with the mail store user ID and password to represent a valid mail account. For example, if the mail account is john_william@my.company.com, enter my.company.com.

Mail Store Access Password **or Mail Store Password**

The password to use when connecting to the mail store host. This property combines with the mail store user ID and password to represent a valid mail account. For example, if the mail account is john_william@my.company.com, enter the password corresponding to john_william.

Mail Store Access Protocol **or Mail Store Protocol**

The protocol to be used when reading mail. Should be IMAP.

Mail Store Access User Name **or Mail Store User**

The user ID to use when connecting to the mail store host.This property combines with the mail store user ID and password to represent a valid mail account. For example, if the mail account is john_william@my.company.com, enter john_william.

Mail Store Host

See Mail Store Access Host

Mail Store Password

See Mail Store Access Password

Mail Store Protocol

See Mail Store Access Protocol

Mail Store User

See Mail Store Access User

Mail Transport Host **or Outgoing Mail Server**

The server to which to connect when sending mail. Specify the fully qualified Internet host name of the mail server, also known as the SMTP server

Mail Transport Password **or Outgoing Mail Password**

The password to provide when connecting to the mail transport host. This property is rarely used for the default SMTP protocol. You can leave this field blank unless you use a transport protocol that requires a user ID and password.

Mail Transport Protocol **or Outgoing Mail Protocol**

The transport or protocol to use when sending mail, such as "POP3", "IMAP4", or "SMTP"

The default is set to SMTP, and usually you should not change it. To use a different protocol, first install the required service provider, and then enter the protocol name in this field.

Mail Transport User **or Outgoing Mail User Name**

The user ID to provide when connecting to the mail transport host. This property is rarely used for the default SMTP protocol. You can leave this field blank unless you use a transport protocol that requires a user ID and password.

Outgoing Mail Originator

See Mail From

Outgoing Mail Server

See Mail Transport Host

Outgoing Mail Password

See Mail Transport Password

Outgoing Mail Protocol

See Mail Transport Protocol

Outgoing Mail User Name

See Mail Transport User

Re-Enter Password

See Confirm Password

6.6.37.1: Administering JavaMail support resources with the Java console

Use the Java administrative console to administer JavaMail sessions.

Work with resources of this type by locating them in the tree view:

WebSphere Administrative Domain -> Resources -> JavaMail Sessions

The instances will be displayed in the details view.

6.6.37.1.1: Configuring new JavaMail support resources with the Java administrative console

1. Click **Console -> New -> JavaMail Session** from the console menu bar.
2. Specify JavaMail session properties.
3. Click **OK**.

Alternatively, use the **Console -> New** menu options pertaining to JavaMail resources.

6.6.37.9: Administering JavaMail providers and sessions with the Application Client Resource Configuration Tool

Use the Application Client Resource Configuration Tool to edit the configurations of JavaMail sessions to be used by your application clients.

Work with objects of this type by locating them in the tree that is displayed by the tool when you use it to openan EAR file. If file with which you are working contains JavaMail sessions, its tree will contain one or more of the following:

Resources -> ***application*.jar** -> **JavaMail Providers** -> **Mail Provider** (a default mail provider) ->**JavaMail Sessions**

Inside the **JavaMail Sessions** folder will be JavaMail sessioninstances.

6.6.37.9.1: Configuring new JavaMail sessions with the Application Client Resource Configuration Tool

During this task, you will configure new mail sessions for your application client. The mail sessions will be associated with the preconfigured default mail provider supplied by the product.

To configure a new JavaMail Session:

1. Start the tool and open the EAR file for which you want to configure the new JavaMail session. The EAR file contents will be displayed in a tree view.
2. From the tree, select the JAR file in which you wantto configure the new JavaMail session.
3. Expand the JAR file to view its contents.
4. Click **JavaMail Providers** -> **MailProvider** -> **JavaMail Sessions**. Do one of the following:
    - Right-click the **JavaMail Sessions** folder and select **New Factory**.
    - On the menu bar, click **Edit** -> **New**.
5. In the resulting property dialog, configure the JavaMail session properties.
6. When finished, click **OK**.
7. On the menu bar, click **File** -> **Save** to save your changes.

6.6.37.9.3: Removing JavaMail sessions with the Application Client Resource Configuration Tool

Please see "Removing objects from EAR files with theApplication Client Resource Configuration Tool", as this task is similar for all object types supported by the tool.

6.6.37.9.4: Updating JavaMail session configurations with the Application Client Resource Configuration Tool

During this task, you will modify (update) the configuration of an existing JavaMail session. Note, you cannot update the default JavaMail provider, but you can viewits properties by performing similar steps.

1. Start the tool and open the EAR file containing the JavaMail session. The EAR file contents will be displayed in a tree view.
2. From the tree, select the JAR file containing the JavaMail session that you want to update.
3. Expand the JAR file to view its contents.
4. Keep expanding the JAR file contents until you locate the particular JavaMail session that you want to update. When you find it, do one of the following:
    - Right-click the object and select **Properties**
    - On the menu bar, click **Edit** -> **Properties**
5. In the resulting property dialog, update the properties. For detailed field help, see:
    - JavaMail session properties
6. When finished, click **OK**.
7. On the menu bar, click **File** -> **Save** to save your changes.

6.6.38: Administering URL providers and URLs (overview)

A Default URL Provider is included in the initial product configuration. It utilizes the URL support provided by the JDK. Any URL resources with protocols supported by the JDK (such as HTTP, FTP, FILE) can use the Default URL Provider.

Please see the related information links for tasks and settings relatedto URL support.

6.6.38.0: Properties of URL providers

Key:

☕ Applies to Java administrative console of Advanced Edition Version 4.0

🌐 Applies to Web administrative console of Advanced Single Server Edition Version 4.0

C:\ Applies to Application Client Resource Configuration Tool

Classpath ☕ **or Class Path** C:\ **or Server Classpath** 🌐

The path to the JAR file containing the implementation classes for the URL provider

Custom Properties C:\

Name-value pairs for setting additional properties

Description ☕ 🌐 C:\

Optional description of the URL Provider, for your administrative records

Name ☕ 🌐 C:\

Administrative name for the URL Provider

Node ☕

The node with which the URL Provider is associated

If using the Java-based administrative console, use the buttons on the node panel to access dialogs for installing drivers on specific nodes, and for uninstalling drivers.

Protocol ☕ 🌐 C:\

The protocol supported by this stream handler.

For Advanced Single Server Edition: This field is required

For the Application Client Resource Configuration Tool: This refers to a *custom*protocol. If you are going to use a standardprotocol, such as "nntp," "smtp," or "ftp," then leave this field blank

Server Classpath 🌐

See Classpath

Stream Handler Class ☕ C:\ **or Stream Handler Class Name** 🌐

Fully qualified name of Java class that implements the stream handler for the protocol specified by the **Protocol** property

6.6.38.0.1: Properties of URLs

Key:

Applies to Java administrative console of Advanced Edition Version 4.0

Applies to Web administrative console of Advanced Single Server Edition Version 4.0

Applies to Application Client Resource Configuration Tool

Category

Administrative category for the URL

Custom Properties

Name-value pairs for setting additional properties

Description

Optional description of the URL, for your administrative records

JNDI Binding Path **or JNDI Name**

The JNDI name for the resource, including any naming subcontexts. This name is used to link the platform's binding information. The binding associates the resources defined in the deployment descriptor of the module to the actual (physical) resources bound into JNDI by the platform.

JNDI Name

See JNDI Binding Name

Name

Administrative name for the URL

Spec **or URL**

The string from which to form a URL

URL

See Spec

URL Provider

The URL Provider that implements the protocol for this URL

6.6.38.1: Administering URL resources with the Java administrative console

Use the Java administrative console to administer URL providers and URLs.

Work with resources of this type by locating them in the tree view:

WebSphere Administrative Domain -> Resources -> URL Providers -> *provider_name* -> URLs

The URL instances will be displayed in the details view.

6.6.38.1.1: Configuring URL resources with the Java administrative console

URL resources allow you to define specific locations for a resource on a network.

To create a URL resource, click **Console -> Wizards -> Create URL Resource** from the console menu bar. This leads to the URL Resource task wizard.

Alternatively, use the **Console -> New** menu options pertaining to URL resources.

Using the URL Resource wizard

You use a URL Resource wizard to define a resource that uses the Default URL Provider. Using the wizard sets basic properties pertaining to URL resources. To set other property values for URL resources, use the properties dialogs for URL resources.

- Naming the URL resource and specifying its URL string
- Completing the URL resource

Naming the URL resource and specifying its URL string

On the Specifying the URL Resource panel:

1. Name your resource.
2. Describe your resource.
3. Specify the string for the URL. A sample string is `http://www.ibm.com`.

   View URL resources properties help
4. Click **Next**.

Completing the URL resource

The final panel lists the URL resource name and the URL string.

If you do not want to change the values specified, click **Finish**. Thewizard will define a URL resource with the values you specified, and display a message indicating whether the resource was successfully created. Your specifiedURL will be created using the Default URL Provider. If the wizard encounters an error, a message will display explaining why a URL resource could not be defined.

To change the values specified, click **Back** to return to the appropriatepanel(s), make any needed changes, and then click **Finish** on this panel.

6.6.38.1.2: Installing and uninstalling URL providers on nodes, with the Java administrative console

To install a provider:

1. Locate a URL provider instance in the tree view.
2. Click the **Nodes** tabbed page.
3. Click the **Install New** button.
4. Configure the new provider:
    1. Select a node on which to install the provider.
    2. Specify or browse for the classpath directory (see URL provider properties).
    3. Click **Install**.
5. Click **Apply** on the URL provider properties view.

To uninstall a provider:

1. Locate a URL provider instance in the tree view.
2. Click the **Nodes** tabbed page.
3. Select a provider from the list of providers.
4. Click **Uninstall**.
5. Click **Apply** on the URL provider properties view.

You can also click a node instance in the tree view todisplay a similar configuration dialog.

6.6.38.9: Administering URL providers and URLs with the Application Client Resource Configuration Tool

Use the Application Client Resource Configuration Tool to edit the configurations of URL providers and URLs to be used by your application clients.

Work with objects of this type by locating them in the tree that is displayed by the tool when you use it to openan EAR file. If file with which you are working contains URL providers and URLs, its tree will contain one or more of the following:

Resources -> ***application*.jar** -> **URL Providers** -> ***url_provider_instance***

where ***url_provider_instance*** isa particular URL provider.

If you expand the tree further, you will also see the **URLs** folders containing the URL instances for each URL provider instance.

6.6.38.9.1: Configuring new URL providers and URLs with the Application Client Resource Configuration Tool

During this task, you will create URL providers and URLs for your client application. Note, in a separate administrative task, the Java code for the required URL provider must be installed on the client machine on which the client application resides.

To configure a new URL provider:

1. Start the tool and open the EAR file for which you want to configure the new URL provider. The EAR file contents will be displayed in a tree view.
2. From the tree, select the JAR file in which you wantto configure the new URL provider.
3. Expand the JAR file to view its contents.
4. Click the folder called **URL Providers**. Do one of the following:
   - Right-click the folder and select **New Provider**.
   - On the menu bar, click **Edit** -> **New**.
5. In the resulting property dialog, configure the URL provider properties.
6. When finished, click **OK**.
7. On the menu bar, click **File** -> **Save** to save your changes.

6.6.38.9.1.1: Configuring new URLs with the Application Client Resource Configuration Tool

During this task, you will create URLs for your client application.

1. In the tree, click the URL provider for which you want to create a URL.
   - Configure a new URL provider.
   - Or, click an existing URL provider.
2. Expand the URL provider to view its **URLs** folder.
3. Click the folder. Do one of the following:
   - Right-click the folder and select **New Factory**.
   - On the menu bar, click **Edit** -> **New**.
4. In the resulting property dialog, configure the URL properties.
5. When finished, click **OK**.
6. On the menu bar, click **File** -> **Save** to save your changes.

6.6.38.9.3: Removing URL providers and URLs with the Application Client Resource Configuration Tool

Please see "Removing objects from EAR files with theApplication Client Resource Configuration Tool", as this task is similar for all object types supported by the tool.

6.6.38.9.4: Updating URL and URL provider configurations with the Application Client Resource Configuration Tool

During this task, you will modify (update) the configuration of an existing URL or URL provider.

1. Start the tool and open the EAR file containing the URL or URL provider. The EAR file contents will be displayed in a tree view.
2. From the tree, select the JAR file containing the URL or URL providerthat you want to update.
3. Expand the JAR file to view its contents.
4. Keep expanding the JAR file contents until you locate the particular URL or URL provider that you want to update. When you find it, do one of the following:
   - Right-click the object and select **Properties**
   - On the menu bar, click **Edit** -> **Properties**
5. In the resulting property dialog, update the properties. For detailed field help, see:
   - URL provider properties
   - URL properties
6. When finished, click **OK**.
7. On the menu bar, click **File** -> **Save** to save your changes.

6.6.39: Administering messaging and JMS providers (overview)

Please see the related information links.

6.6.39.0: Properties of JMS providers

Key:

Applies to Java administrative console of Advanced Edition Version 4.0

Applies to Web administrative console of Advanced Single Server Edition Version 4.0

Applies to Application Client Resource Configuration Tool

Binding Classname or Binding Class or JNDI Binding Mechanism

Java classname to be used for namespace binding. This value is required only for providers with non-standard binding requirements.

Classpath or Class Path or Server Class Path

The class path that identifies the location of the driver classes (the JMS initial context factory)

Context Factory Class or Context Factory Classname

The Java class name of the JMS providers initial context factory

Custom Properties

Name-value pairs for setting additional properties

Description

Optional description for your administrative records

External Initial Context Factory

Java classname of the initial context factory of a provider

External Provider URL or Provider URL

JMS provider URL for external JNDI lookups

JNDI Binding Mechanism

See Binding Classname

Name

Administrative name for this provider

Nodes

The nodes with which this provider is associated. See also node properties.

If using the Java-based console, use the buttons on the node panel to access dialogs for installing providers on specific nodes, and for uninstalling providers.

Provider URL

See External Provider URL

Server Class Path

See Classpath

6.6.39.0.1: Properties of JMS connection factories

Key:

☕ Applies to Java administrative console of Advanced Edition Version 4.0

🌐 Applies to Web administrative console of Advanced Single Server Edition Version 4.0

C:\ Applies to Application Client Resource Configuration Tool

Category 🌐

Optional category for your administrative records

Connection Type 🌐 C:\

Whether the JMS Destination is a queue or topic. Queues are used for point-to-point messaging. Topics are used for publish-and-subscribe messaging.

Custom Properties C:\

Name-value pairs for setting additional properties

Description ☕ 🌐 C:\

Optional description for your administrative records

External JNDI Name C:\ 🌐 **or External JNDI Path** ☕

Namespace location of JMS created connection factory

External JNDI Path ☕

See External JNDI Name

JMS Provider ☕

The JMS provider with which this connection factory is associated

JNDI Name C:\ 🌐

Namespace location of JMS created connection factory, including any naming subcontexts.

The name is used to link the platform binding information. The binding associates the resources defined the deployment descriptor of the module to the actual (physical) resources bound into JNDI by the platform.

Name ☕ 🌐 C:\

Administrative name for this JMS connection factory

6.6.39.0.2: Properties of JMS destinations

Key:

☕ Applies to Java administrative console of Advanced Edition Version 4.0

🌐 Applies to Web administrative console of Advanced Single Server Edition Version 4.0

C:\ Applies to Application Client Resource Configuration Tool

Category 🌐

Optional category for your administrative records

Custom Properties C:\

Name-value pairs for setting additional properties

Description ☕ 🌐 C:\

Optional description for your administrative records

Destination Type C:\ 🌐

Whether the JMS Destination is a queue or topic. Queues are used for point-to-point messaging. Topics are used for publish-and-subscribe messaging.

External JNDI Name C:\ 🌐 **or External JNDI Path** ☕

Namespace location of JMS created destination

External JNDI Path ☕

See External JNDI Name

JMS Provider ☕

The JMS provider with which this connection factory is associated

JNDI Name C:\ 🌐

Namespace location of JMS created connection factory, including any naming subcontexts.

The name is used to link the platform binding information. The binding associates the resources defined the deployment descriptor of the module to the actual (physical) resources bound into JNDI by the platform.

Name ☕ 🌐 C:\

Administrative name for this JMS destination

6.6.39.1: Administering JMS support resources with the Java administrative console

Use the Java administrative console to administer JMS providers, JMS connectionfactories, and JMS destinations.

Work with resources of this type by locating them in the tree view:

WebSphere Administrative Domain -> Resources -> JMS Providers -> ***provider_name***

Expand the tree further to see the **JMS Connection Factories** and **JMS Destinations**folders. The corresponding instances will be displayed in the details view.

6.6.39.1.1: Configuring JMS resources with the Java administrative console

To create a JMS connection factory or destination, click **Console -> Wizards -> Create JMS Resource** from the console menu bar. This leads to the JMS Resource task wizard.

Alternatively, use the **Console -> New** menu options pertaining to JMS resources.

Using the JMS Resource wizard

You can use a JMS Resource wizard to define a connection factory or destination. Using the wizard sets basic properties pertaining to JMS resources. To set other property values, use the properties dialogs for JMS resources.

- Selecting to create a new connection factory or destination
- Naming the JMS resource and specifying its JNDI path
- Specifying a JMS provider
- Creating a new JMS provider
- Completing the connection factory or destination

Selecting to create a new connection factory or destination

On the Specifying a JMS Resource panel, select to create a new JMS connectionfactory or a new destination, and then click **Next**.

Naming the JMS resource and specifying its JNDI path

On the Creating a JMS Connection Factory or Creating a JMS Destination panel:

1. Specify the JNDI path where the JMS administrative tool has bound this resource.

   If a connection factory is being created, it is recommended that within the name you identify the JMS resource to which the factory will provide connections. For example, enter `QueueConnectionFactory`.
2. Describe your resource.
3. Specify the external JNDI path where your JMS resource should be stored. A sample path is `jms/QueueConnectionFactory`.

   View JMS resources properties help
4. Click **Next**.

Specifying a JMS provider

On the Specifying a JMS Provider panel, select an existing JMS provider or opt to create a new JMS provider, and then click **Next**.

A JMS provider represents a library that implements the JMS interfaces fora specific vendor.

Creating a new JMS provider

On the Creating a JMS Provider panel:

1. Name your JMS provider. Further, specify a context factory class name and a URL for the provider. Optionally, describe your JMS provider and specify a binding class name.

   View JMS resources properties help

2. Click **Next**.

Completing the connection factory or destination

The final panel lists the JMS resource name, the external JNDI path where the resource will reside, and the provider name. If a new provider is being created, the panel also lists the context factory class name and the provider URL.

If you do not want to change the values specified, click **Finish**. Thewizard will define a connection factory or destination with the values you specified, and display a message indicating whether the resource was successfully created. If the wizard encounters an error, a message will display explaining why a connection factoryor destination could not be defined.

To change the values specified, click **Back** to return to the appropriatepanel(s), make any needed changes, and then click **Finish** on this panel.

6.6.39.1.2: Installing and uninstalling JMS providers on nodes, with the Java administrative console

To install a provider:

1. Locate a JMS provider instance in the tree view.
2. Click the **Nodes** tabbed page.
3. Click the **Install New** button.
4. Configure the new provider:
    - Select a node on which to install the provider.
    - Specify or browse for the classpath directory (see JMS provider properties).
    - Click **Install**.
5. Click **Apply** on the JMS provider properties view.

To uninstall a provider:

1. Locate a JMS provider instance in the tree view.
2. Click the **Nodes** tabbed page.
3. Select a provider from the list of providers.
4. Click **Uninstall**.
5. Click **Apply** on the JMS provider properties view.

You can also click a node instance in the tree view todisplay a similar configuration dialog.

6.6.39.9: Administering JMS providers, connection factories, and destinations with the Application Client Resource Configuration Tool

Use the Application Client Resource Configuration Tool to edit the configurations of JMS providers, JMS connectionfactories, and JMS destinations to be used by your application clients.

Work with objects of this type by locating them in the tree that is displayed by the tool when you use it to openan EAR file. If file with which you are working contains JMS providers, JMS connectionfactories, and JMS destinations, its tree will contain one or more of the following:

Resources -> ***application*.jar** -> **JMS Providers** -> ***jms_provider_instance***

where ***jms_provider_instance*** isa particular JMS provider.

If you expand the tree further, you will also see the **JMS Connection Factories** and **JMS Destinations** folders containing the connection factory and destination instances for each JMS provider instance.

6.6.39.9.1: Configuring new JMS providers with the Application Client Resource Configuration Tool

During this task, you will create new JMS provider configurations for your applicationclient. The client application can make use of a messaging service through the Java Message Service APIs. A JMS provider provides two kinds of J2EE factories. One is a JMS Connection factory, and the other is aJMS destination factory.

Note, in a separate administrative task, the JMS client must be installed on the client machine where the client application resides. The messaging product vendor must provide an implementation of the JMS client. For more information, see your messaging product documentation.

To configure a new JMS provider:

1. Start the tool and open the EAR file for which you want to configure the new JMS provider. The EAR file contents will be displayed in a tree view.
2. From the tree, select the JAR file in which you wantto configure the new JMS provider.
3. Expand the JAR file to view its contents.
4. Click the folder called **JMS Providers**. Do one of the following:
   - Right-click the folder and select **New Provider**.
   - On the menu bar, click **Edit** -> **New**.
5. In the resulting property dialog, configure the JMS provider properties.
6. When finished, click **OK**.
7. On the menu bar, click **File** -> **Save** to save your changes.

6.6.39.9.1.1: Configuring new connection factories with the Application Client Resource Configuration Tool

During this task, you will create a new JMS connection factory configuration for your applicationclient.

To configure a new connection factory:

1. In the tree, click the JMS provider for which you want to create a connection factory.
    - Configure a new JMS provider.
    - Or, click an existing JMS provider.
2. Expand the JMS provider to view its **JMS Connection Factories** folder.
3. Click the folder. Do one of the following:
    - Right-click the folder and select **New Factory**.
    - On the menu bar, click **Edit** -> **New**.
4. In the resulting property dialog, configure the JMS connection factory properties.
5. When finished, click **OK**.
6. On the menu bar, click **File** -> **Save** to save your changes.

6.6.39.9.1.2: Configuring new JMS destinations with the Application Client Resource Configuration Tool

During this task, you will create new JMS destination configuration for your applicationclient.

To configure a new destination:

1. In the tree, click the JMS provider for which you want to create a destination.
    - Configure a new JMS provider.
    - Or, click an existing JMS provider.
2. Expand the JMS provider to view its **JMS Destinations** folder.
3. Click the folder. Do one of the following:
    - Right-click the folder and select **New Factory**.
    - On the menu bar, click **Edit** -> **New**.
4. In the resulting property dialog, configure the JMS destination properties.
5. When finished, click **OK**.
6. On the menu bar, click **File** -> **Save** to save your changes.

6.6.39.9.3: Removing JMS providers, connection factories, and destinations with the Application Client Resource Configuration Tool

Please see "Removing objects from EAR files with theApplication Client Resource Configuration Tool", as this task is similar for all object types supported by the tool.

6.6.39.9.4: Updating JMS provider, connection factory, and destination configurations with the Application Client Resource Configuration Tool

During this task, you will modify (update) the configuration of an existing JMS provider, connection factory, or destination.

1. Start the tool and open the EAR file containing the JMS provider, connection factory, or destination.
   The EAR file contents will be displayed in a tree view.
2. From the tree, select the JAR file containing the JMS provider, connection factory, or destinationthat you want to update.
3. Expand the JAR file to view its contents.
4. Keep expanding the JAR file contents until you locate the particular JMS provider, connection factory, or destination that you want to update. When you find it, do one of the following:
   - Right-click the object and select **Properties**
   - On the menu bar, click **Edit** -> **Properties**
5. In the resulting property dialog, update the properties. For detailed field help, see:
   - JMS provider properties
   - JMS connection factory properties
   - JMS destination properties
6. When finished, click **OK**.
7. On the menu bar, click **File** -> **Save** to save your changes.