XML -- table of contents

Development

Administration

4.2.3: Incorporating XML

IBM WebSphere Application Server provides XML Document Structure Services, which consist of a document parser, a document validator, and a document generator for server-side XML processing.

See article 4.1.1.2 for all of the details about XML supportin the product.If you are just becoming familiar with XML, start with article 0.33, a primer on XML concepts, vocabulary, and uses.

Other related information provides guidance on the following topics:

- Structure -- defining and obeying the syntax for an XML tag set
- Content -- determining the mechanism for filling XML tags with data
- Presentation -- determining the mechanism for formatting and displaying XML content

In addition, some special topics are covered, including DOM objects andmanipulation of Channel Definition Format (CDF) files as illustrated bythe SiteOutliner example.

When you install IBM WebSphere Application Server, the core XML APIs are automatically added to the appropriate class path, enabling you to serve static XML documents as soon as the product is installed.

To serve XML documents that are dynamically generated, use the core APIs to develop servlets or Web applications that generate XML documents (for example, the applications might read the document content from a database) and then deploy those components on your application server.

4.2.3.2: Specifying XML document structure

The structure of an XML document is governed by syntax rules for its tag set. Those tags are defined formally in an XML-based grammar, such as a Document Type Definition (DTD). At the time of this publication, DTD is the most widely-implemented grammar. Therefore, this article discusses options for using DTDs.

Options for XML document structure include:

Do not use a DTD. Not using a DTD enables maximum flexibility in evolving XML document structure, but this flexibility limits the ability to share the documents among users and applications. An XML document can be parsed without a DTD. If the parser does not find an inline DTD or a reference to an external DTD, the parser proceeds using the actual structure of the tags within the document as an implied DTD. The processor evaluates the document to determine whether it meets the rules for well-formedness.

Use a public DTD. Various industry and other interest groups are developing DTDs for categories of documents, such as chemical data and archival documents. Many of these DTDs are in the public domain and are available over the Internet. Using an industry standard DTD maximizes sharing documents among applications that act on the grammar. If the standard DTD does not accomodate the schema the applications need, flexibility is limited.

Several industry and interest groups have developed and proposed DTD grammars for the types of documents they produce and exchange. To make it easier for you to use those grammars, local copies are installed with the product. Use the grammars as examples in developing your own grammars as well as for creating and validating XML documents of those types. The library is located atproduct_installation_root\web\xml\grammar\

Develop a DTD. If none of the public DTDs meet an enterprise's needs and enforcing document validity is a requirement, the XML implementers can develop a DTD. Developing a DTD requires careful analysis of the information (data) that the documents will contain.

For DTD updates,visit the XML Industry Portal.For details about the DTD specifications and sample DTDs, refer toIBM's developerWorkssite for education and other DTD resources.

4.2.3.3: Providing XML document content

The content of an XML document is the actual data that appears within the document tags. XML implementers must determine the source and the mechanism for putting the data into the document tags. The options include:

Static content. XML document content is created and stored on the Web server as static files. The XML document author composes the document to include valid XML tags and data in a manner similar to how HTML authors compose static HTML files. This approach works well for data that is not expected to change or that will change infrequently. Examples are journal articles, glossaries, and literature.

Dynamically generated content. XML document content can be dynamically generated from a database and user input. In this scenario, XML-capable servlets, Java beans, and even inline Java code within a JavaServer Page (JSP) file can be used to generate the XML document content.

A hybrid of static and dynamically generated content. This scenario involves a prudent combination of static and dynamically generated content.

You can also use XSL to add to or remove information from existing XML content.For details, see the Related information.

4.2.3.4: Rendering XML documents

Options for presenting XML documents include:

Present the XML document in an XML-enabled browser. An XML-enabled browser can parse a document, apply its XSL stylesheet, and present the document to the user. Searching and enabling users to modify an XML document are other possible functions of XML-enabled browsers.

Present the XML document to a browser that converts XML to HTML. Until XML-enabled browsers are readily available, presenting XML documents to users will involve converting the XML document to HTML. That conversion can be handled by conversion-capable browsers. Another option is to use JavaScript or ActiveX controls embedded within the XML document. Microsoft Internet Explorer Version 5 is an XML-to-HTML converter. HTML is not the only format to which XML documents can be converted. It's just the easiest to implement given the commerically available browsers and user agents.

Send an HTML file to the browser. If the users do not have XML-capable browsers, the XML document must be converted at the server before being transmitted to the browser. The server-side XML application that handles the conversion could also determine the capability of the browser before converting the document to HTML, to avoid unnecessary processing if the browser is XML-capable. The XSL processor included with this product supports such server-side functions.

Using XSL to convert XML documents to other formats

IBM WebSphere Application Server includes the Lotus XSL processor and its open-sourceversion, Xalan, for formatting and converting XML documents. Processing can be done at the server or at the browser, to HTML or to other XML-compliant markup languages.For sample code, see the Xalan documentation.

Converting XML documents at the server

One option for presenting an XML document is for the server to convert the XML document to HTML and return the HTML document to the client. On the server side, this typically requires the creation of a servletto handle the processing of one data stream (the XML document)with another (the XSL document).The output of that process is then forwarded back to the browser.

Server-side processing often requires the passing in of parameters through theXSL processor to customize the output.For an example, see the Xalan documentation.

4.2.3.6: Using DOM to incorporate XML documents into applications

The Document Object Model (DOM) is an API for representing XML and HTML documentsas objects that can be accessed by object-oriented programs, such as business logic, for the purposes of creating, navigating, manipulating, and modifying the documents.

Article 0.33.3 introduces DOM concepts and vocabulary. Article 4.1.1.2 tells youwhere to find the DOM specification and org.w3c.dom package.

Article 4.2.3.6.1 providesa quick reference so that you can jump right into DOM development, referring to thepackage and specification as needed.

4.2.3.6.1: Quick reference to DOM object interfaces

This section highlights a few of the object interfaces. Refer to theDOM Specification for details (see article 4.1.1.2).

Node methods

Node methods include:

Method	**Description**
hasChildNodes	Returns a boolean to indicate whether a node has children
appendNode	Appends a new child node to the end of the list of children for a parent node
insertBefore	Inserts a child node before the existing child node
removeChild	Removes the specified child node from the node list and returns the node
replaceChild	Replaces the specified child node with the specified new node and returns the new node

Document methods

The Document object represents the entire XML document. Document methods include:

Method	**Description**
createElement	Creates and returns an Element (tag) of the type specified. If the document will be validated against a DTD, that DTD must contain an Element declaration for the created element.
createTextNode	Creates a Text node that contains the specified string
createComment	Creates a Comment node with the specified content (enclosed within `<!--` and `-->` tags)
createAttribute	Creates an Attribute node of the specified name. Use the setAttribute method of Element to set the value of the Attribute. If the document will be validated against a DTD, that DTD must contain an Attribute declaration for the created attribute.
createProcessingInstruction	Creates a Processing Instruction with the specified name and data (enclosed within <? and ?> tags). A processing instruction is an instruction to the application (such as an XML document formatter) that receives the XML document.

Element methods

Element node methods include:

Method	**Description**
getAttribute	Returns the value of the specified attribute or empty string
setAttribute	Adds a new attribute-value pair to the element

removeAttribute	Removes the specified attribute from the element
getElementsByTagName	Returns a list of the element descendants that have the specified tag name

A Text node can be a child of an Element or Attribute node and contains the textual content (character data) for the parent node. If the content does not include markup, all of the content is placed within a single Text node. If the content includes markup, that markup is placed in one or more Text nodes that are siblings of the Text node that contains the non-markup content.

The Text node extends the CharacterData interface, which has methods for setting, getting, replacing, inserting, and making other modifications to a Text node. In addition to those methods, the Text node adds a method:

Method	**Description**
splitText	Splits the Text node at the specified offset. Returns a new Text node, which contains the original content starting at the offset. The original Text node contains the content from the beginning to the offset.

4.2.3.7: SiteOutliner sample

The SiteOutliner servlet illustrates how to use the XML Document Structure Services to generate and view a Channel Definition Format (CDF) file for a target directory on the servlet's Web server. Use Lotus Notes 5 (the Headlines page), Microsoft Internet Explorer 4 Channel Bar, PointCast, Netscape Navigator 4.06, orother CDF-capable viewers to view and manipulate the CDF file.

SiteOutliner is part of the WebSphere Samples Gallery. When you open the gallery,follow the links to SiteOutliner to run it on your local machine.

6.6.0.2: Command line administration

The following command line administrative tools are available for interacting with theWebSphere administrative server.

- Use **wscp** for operational and configuration tasks such as starting, stopping, and configuring application servers and other object types.
- Use **XMLConfig** for configuration tasks. This tool provides a way to work with the administrative domain as represented in WebSphere XML.

6.6.0.2.1: XMLConfig command line interface for XML configuration

Use the *XMLConfig* tool to import and export configuration data to and from the WebSphere Application Server administration repository. This XML-based approach complements the administration you can perform through the administrative console.

You can use this tool to perform multiple changes to the WebSphere Application Server administration repository at a single time without having to go through the repetitive routines on the administration console. You may also use this tool to extract the repository information from one server and import it onto a cloned server.

The XMLConfig tool provides three fundamental features:

- **full export**

  Generates an XML document describing the configuration of the entire administrative domain. In effect, the full export takes a "snapshot" of the administrative repository contents. Click here for an example of a full export, including a discussion of the various parts of the resulting XML file.

- **partial export**

  Provides an XML document specifying administrative objects to export from the administrative domain. The objects and their children are exported to an XML document that you specify. Click here for an example of a partial export.

- **import**

  Imports a newly created XML document or a document you previously exported and modified. In the XML document, you can add, modify, or remove administrative objects such as servlets and virtual hosts. Your XML document can replace the administrative repository contents partially or entirely.

6.6.0.2.1.1: XMLConfig - Command syntax

This section describes the command line syntax for the XMLConfig tool.

Because setting the class path appropriately is vital to the tool's success, the product contains an XMLConfig.bat (Windows NT) or XMLConfig.sh (*IX) file for starting the tool. The file is located in the bin directory of the product installation root, uses the com.ibm.websphere.xmlconfig.XMLConfig class, and has the following command-line syntax:

```
{ ( -import xml_data_file ) ||     [ ( -export xml_output_file [-partial xml_data_file] )
-adminNodeName primary_node_name       [-nameServiceHost host_name [ -nameServicePort port_number ]]
[-traceString trace_spec [-traceFile file_name]]       [-generatePluginCfg true | false]
[-substitute "key1=value1[;key2=value2;[...]]"]}
```

Supported arguments include:

-adminNodeName

Required argument that specifies the node containing the administrative server to which you are connecting. The value of this argument must match the node name given in the topology tree on the Topology tab of the WebSphere Administrative Console.

-import || -export || -export -partial

Required argument that specifies the operation to perform: an import or export. Unless you also specify the parameter -partial, the export is treated as a full export.

-nameServiceHost, -nameServicePort

Optional arguments that specify the host name of the machine that contains the naming service, and the port through which to communicate with the naming service. The default value of -nameServicePort is 900.

-traceString

Optional argument that specifies the internal code to trace. For more information, see the traceString section of the trace help.

-generatePluginCfg

Generate the plug-in configuration if necessary.

-substitute

Optional argument that specifies the variables to be substituted (for example, `-substitute "NODE_NAME=admin_node;APP_SERVER=default_server"`).

In the input XML file, each key should appear as key for substitution. This argument substitutes any occurrence of `$NODE_NAME$` with admin_node and `$APP_SERVER$` with default_server in the input XML file.

If the substitution string contains semicolons,use `$semiColon$` to separate it from the ";" delimiter.

The following examples demonstrate correct syntax. Node1 is the name by which the node that contains the administrative server is administered.

Import operation:

```
XMLConfig -adminNodeName Node1 -import import.xml
```

Full export operation:

```
XMLConfig -adminNodeName Node1 -export export.xml
```

Partial export operation:

```
XMLConfig -adminNodeName Node1 -export export.xml -partial imput.xml
```

6.6.0.2.1.1.1: XMLConfig - Example of a full export

The following example shows the XML elements for each object type in the WebSphere administrative domain. It is a full export of the administrative repository, featuring the default administrative configuration.

To produce a similar export, you would issue the command:

```
XMLConfig -adminNodeName host_name -export export.xml
```

where host_name is the host name of the machine that contains the administrative serverand export.xml is the name of the output file.

When you perform an export, the XML output file does not contain blank lines or gaps. In contrast, the following export has been broken into segments, each of which is briefly discussed.

Also, this example was obtained from a system that used the default configuration, and might vary slightly from actual output due to changes on your particular system. It is recommended that you try the export command yourself to see exactly what output is produced.

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE websphere-sa-config   SYSTEM
"file:///$XMLConfigDTDLocation$$dsep$xmlconfig.dtd"><websphere-sa-config>
```

These tags mark the beginning of the export. The following part of the export contains a tag for the default virtual host, *default_host*, in the administrative domain.

The default host recognizes several MIME types, which are listed as part of a MIME table in the virtual-host tag. In fact, there are so many MIME types that many are omitted from the example. These are followed by the aliases for the virtual host.

Skip ahead

```
<virtual-host action="update" name="default_host">    <mime-table>        <mime
type="application/vnd.lotus-wordpro">            <ext>lwp</ext>            <ext>sam</ext>
</mime>        <mime type="text/tab-separated-values">           <ext>tsv</ext>        </mime>
<mime type="application/x-troff-me">            <ext>me</ext>        </mime>        <mime
type="image/x-portable-anymap">            <ext>pnm</ext>        </mime>        <mime
type="text/x-ssi-html">            <ext>htmls</ext>            <ext>shtml</ext>        </mime>
<mime type="application/vnd.lotus-screencam">            <ext>scm</ext>        </mime>        <mime
type="application/xml">            <ext>xsl</ext>        </mime>        ...    </mime-table>
<alias-list>        <alias>*:80</alias>        <alias>*:9080</alias>    </alias-list></virtual-host>
```

Next is the Java Messaging Service (JMS) configuration:

Skip ahead

```
<jms-provider action="update" name="IBM MQSeries">    <description>(OPTIONAL) Description of JMS
provider.</description>    <external-initial-context-factory>
com.ibm.websphere.naming.WsnInitialContextFactory    </external-initial-context-factory>
<external-provider-url>iiop://localhost</external-provider-url>    <jndi-binding-mechanism/>
<jms-connection-factory action="update" name="Test JMSConnFactory">        <description>Test
factory</description>        <jndi-name>jms/Test JMSConnFactory</jndi-name>
<external-jndi-name>TestJMSFactory</external-jndi-name>    </jms-connection-factory>
<jms-destination action="update" name="JMSTestDest">        <description>JMS destination
</description>        <external-jndi-name>JMSDest</external-jndi-name>
<jndi-name>jms/JMSTestDest</jndi-name>    </jms-destination></jms-provider>
```

The following configuration describesa resource adaptor for CICS:

Skip ahead

```
<j2c-resource-adapter action="update" name="TestJ2cAdapter">
<description>TestJ2cAdapter</description>
<archive-file>D:\WebSphere\AppServer\bin\cicseci.rar</archive-file>    <j2c-adapter-install-info>
<node-name>subodh</node-name>        <resource-archive-file>
D:\WebSphere\AppServer\installedConnectors\cicseci.rar        </resource-archive-file>
</j2c-adapter-install-info>    <j2c-connection-factory action="update" name="TestFactory">
<jndi-name>TestJ2cFactory</jndi-name>        <description>Test factory</description>
<connection-timeout>0</connection-timeout>        <maximum-connections>0</maximum-connections>
<minimum-connections>0</minimum-connections>        <reap-time>0</reap-time>
<unused-timeout>0</unused-timeout>        <pool-name>POOLNAME</pool-name>
<subpool-name>SUBPOOLNAME</subpool-name>        <config-property name="TraceLevel" value="1">
<description>(OPTIONAL)The level of trace to be output                to the Server Trace Log. Range
0-3. 0=off,                1=exceptions, 2=1+entry/exit, 3=2+debug</description>
<type>java.lang.Integer</type>        </config-property>        <config-property
name="ServerSecurity" value="">            <description>(OPTIONAL)Fully Qualified Class
implementing ServerSecurity for connections to the                Gateway (use on conjunction with
ClientSecurity            </description>            <type>java.lang.String</type>
</config-property>        <config-property name="PortNumber" value="2006">
<description>The port number the gateway is listening on</description>
<type>java.lang.String</type>        </config-property>        <config-property
```

```
name="KeyRingPassword" value="">            <description>The Password for the KeyRing
Class</description>            <type>java.lang.String</type>        </config-property>
<config-property name="ConnectionURL" value="">            <description>The URL of the CICS
Transaction Gateway</description>            <type>java.lang.String</type>        </config-property>
<config-property name="ClientSecurity" value="">            <description>(OPTIONAL)Fully Qualified
Class                implementing ClientSecurity for connections to the                Gateway (use
on conjunction with ServerSecurity</description>            <type>java.lang.String</type>
</config-property>        <config-property name="TranName" value="">            <description>The
Transaction name for programs to run under.            </description>
<type>java.lang.String</type>        </config-property>        <config-property name="TPNName"
value="">            <description>The TPN id for programs to run under. This                takes
precedence over TranName.</description>            <type>java.lang.String</type>
</config-property>        <config-property name="Password" value="">            <description>A
Password for the UserName</description>            <type>java.lang.String</type>
</config-property>        <config-property name="ServerName" value="">            <description>The
CICS Server as defined in the CICS                Transaction Gateway</description>
<type>java.lang.String</type>        </config-property>        <config-property name="UserName"
value="">            <description>A user Name to access CICS Resources</description>
<type>java.lang.String</type>        </config-property>        <config-property name="KeyRingClass"
value="">            <description>Fully Qualified Class containing the SSL                Keyrings.
Required only for SSL protocol</description>            <type>java.lang.String</type>
</config-property>    </j2c-connection-factory></j2c-resource-adapter>
```

Next is the configuration for the default URL provider:

Skip ahead

```
<url-provider action="update" name="Default URL Provider">    <description>This is the URL Provider
for the protocols        supported by the JDK (e.g., http, file, ftp, etc.).  The        protocol
and stream handler class name properties are not        used for this provider.</description>
<protocol>unused</protocol>    <stream-handler-class-name>unused</stream-handler-class-name>    <url
action="create" name="TestURL">        <description>test ur</description>
<specification>testurl.com</specification>        <jndi-name>testurl</jndi-name>    </url>
<install-info>        <node-name>subodh</node-name>
<jar-file-location>unused</jar-file-location>    </install-info></url-provider>
```

The following configuration is for JavaMail:

Skip ahead

```
<mail-session action="update" name="TestMailSession">    <description>Test mail sesson</description>
<jndi-name>TestMail</jndi-name>    <mail-transport-protocol>smtp</mail-transport-protocol>
<mail-transport-host>testServer</mail-transport-host>
<mail-transport-user>testuser</mail-transport-user>
<mail-transport-password>test</mail-transport-password>    <mail-from>testoriginator</mail-from>
<mail-store-protocol>imap</mail-store-protocol>    <mail-store-host>testhost</mail-store-host>
<mail-store-user>testuser</mail-store-user>
<mail-store-password>test</mail-store-password></mail-session>
```

The following section contains information aboutthe JDBC database driver and datasources. The configuration contains multiple data-source elements,only one of which is shown:

Skip ahead

```
<jdbc-driver action="update" name="Sample DB Driver">    <implementation-class>
COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource    </implementation-class>    <description/>
<data-source action="update" name="SampleDataSource">        <database-name>was</database-name>
<minimum-pool-size>1</minimum-pool-size>        <maximum-pool-size>10</maximum-pool-size>
<connection-timeout>180</connection-timeout>        <idle-timeout>1800</idle-timeout>
<orphan-timeout>1800</orphan-timeout>        <statement-cache-size>100</statement-cache-size>
<default-user>subodhv</default-user>        <default-password>{xor}MzIsZg==</default-password>
<disable-auto-connectioncleanup>false</disable-auto-connectioncleanup>        <description>Sample
Data Source</description>        <jndi-name>jdbc/SampleDataSource</jndi-name>
<config-properties>            <property name="serverName" value=""/>            <property
name="portNumber" value=""/>            <property name="URL" value=""/>        </config-properties>
</data-source>    ...    <install-info>        <node-name>subodh</node-name>
<jdbc-zipfile-location>            K:/PROGRA~1/SQLLIB/java/db2java.zip
</jdbc-zipfile-location>    </install-info></jdbc-driver>
```

The following section contains the node, or physical machine, in the administrative domain. Its host name is subodh:

```
<node action="update" name="subodh">
```

Next is an application server. In this case, it is the default application server, Default Server:

Skip ahead

```
    <application-server action="update" name="Default Server">        <executable>java</executable>
```

```
<command-line-arguments/>        <environment/>        <user-id/>        <group-id/>
<working-directory/>        <umask>18</umask>        <stdin/>
<stdout>D:\WebSphere\AppServer/logs/Default_Server_stdout.log</stdout>
<stderr>D:\WebSphere\AppServer/logs/Default_Server_stderr.log</stderr>
<process-priority>20</process-priority>
<maximum-startup-attempts>2</maximum-startup-attempts>        <ping-interval>60</ping-interval>
<ping-timeout>200</ping-timeout>        <ping-initial-timeout>300</ping-initial-timeout>
<selection-policy>roundrobinpreferlocal</selection-policy>        <trace-specification/>
<trace-output/>        <transaction-log-file/>        <olt-enabled>false</olt-enabled>
<system-properties/>        <debug-enabled>false</debug-enabled>
<transaction-timeout>120</transaction-timeout>
<transaction-inactivity-timeout>60000</transaction-inactivity-timeout>
<thread-pool-size>20</thread-pool-size>        <security-enabled>false</security-enabled>
<admin-agent-ior/>        <cache-config>            <cache-size>2047</cache-size>
<cache-sweep-interval>1000</cache-sweep-interval>            <passivation-directory/>
</cache-config>        <log-file-spec/>        <performance-monitor-spec>
pmi=none:beanModule=maximum:connectionPoolModule=medium:
j2cModule=medium:jvmRuntimeModule=none:jvmpiModule=low:
servletSessionsModule=none:threadPoolModule=high:
transactionModule=none:webAppModule=maximum        </performance-monitor-spec>
<olt-server-host>localhost</olt-server-host>        <olt-server-port>2102</olt-server-port>
<selection-policy>roundrobinpreferlocal</selection-policy>        <source-path/>
<wlm-template-ior/>        <thread-pool-size>20</thread-pool-size>        <jvm-config>
<initial-heap-size>64</initial-heap-size>            <max-heap-size>64</max-heap-size>
<generated-command-line-arguments>                -Xms64m -Xmx64m
</generated-command-line-arguments>            <system-properties/>
<additional-command-line-arguments/>            <debug-mode>false</debug-mode>
<debug-string/>            <run-hprof>false</run-hprof>            <hprof-args/>
<disable-jit>false</disable-jit>            <verbose-class-loading>false</verbose-class-loading>
<verbose-jni>false</verbose-jni>            <verbose-gc>false</verbose-gc>
<boot-classpath-replace/>            <boot-classpath-append/>            <boot-classpath-prepend/>
</jvm-config>
```

Within the application-server configurationare the attributes of the Web container, as follows:

Skip ahead

```
        <web-container>            <dynamic-cache-config>                <enabled>false</enabled>
<cache-size>0</cache-size>                <default-priority>0</default-priority>
</dynamic-cache-config>            <transport name="http">
<transport-host>*</transport-host>                <transport-port>9080</transport-port>
<http-transport>                    <connection-timeout>5</connection-timeout>
<backlog-connections>50</backlog-connections>
<keep-alive-timeout>5</keep-alive-timeout>
<maximum-keep-alive>25</maximum-keep-alive>
<maximum-req-keep-alive>100</maximum-req-keep-alive>
<ssl-enabled>false</ssl-enabled>                </http-transport>            </transport>
<thread-maximum-size>50</thread-maximum-size>
<thread-minimum-size>25</thread-minimum-size>
<thread-inactivity-timeout>10</thread-inactivity-timeout>
<thread-is-growable>true</thread-is-growable>
```

A session manager configuration follows.The tag for the Web container is still open.

Skip ahead

```
            <session-manager>
<enable-security-integration>false</enable-security-integration>
<enable-ssl-tracking>false</enable-ssl-tracking>
<invalidation-schedule-first-hour>0</invalidation-schedule-first-hour>
<invalidation-schedule-second-hour>0                </invalidation-schedule-second-hour>
<persistent-db2-row-size>0</persistent-db2-row-size>
<maximum-inmemory-session-count>1000</maximum-inmemory-session-count>
<write-contents>0</write-contents>                <write-frequency>0</write-frequency>
<write-interval>0</write-interval>                <enable-sessions>false</enable-sessions>
<enable-url-rewriting>false</enable-url-rewriting>
<enable-cookies>true</enable-cookies>                <enable-protocol-switch-rewriting>false
</enable-protocol-switch-rewriting>                <cookie name="JSESSIONID">
<domain/>                <maximum>-1</maximum>                    <path>/</path>
<secure>false</secure>                </cookie>
<tuning-invalidation-time>30</tuning-invalidation-time>
<persistent-sessions>false</persistent-sessions>                <data-source name="">
<default-user/>                    <default-password/>                </data-source>
<using-multi-row>false</using-multi-row>                <allow-overflow>true</allow-overflow>
</session-manager>
```

Configuration for the Web container ends:

```
        </web-container>
```

Next is an ORB configuration:

Skip ahead

```
        <orb-config>            <bootstrap-host-name/>
<bootstrap-port>900</bootstrap-port>            <comm-trace-enabled>false</comm-trace-enabled>
<connection-cache-maximum>240</connection-cache-maximum>
<connection-cache-minimum>100</connection-cache-minimum>            <external-config-url/>
<force-tunnel>whenrequired</force-tunnel>            <listener-port>0</listener-port>
<locate-request-timeout>180</locate-request-timeout>            <local-host-name/>
<lsd-host-name/>            <no-local-copies>false</no-local-copies>
<request-retries-count>1</request-retries-count>
<request-retries-delay>0</request-retries-delay>            <request-timeout>180</request-timeout>
<thread-pool-size>20</thread-pool-size>            <tunnel-agent-url/>
<rmi-remote-code-base/>            <ssl-listener-port>0</ssl-listener-port>        </orb-config>
```

The WebSphere plug-in configuration comes next:

Skip ahead

```
        <custom-service-config-list>            <custom-service-config
name="Automatic Generation of Plugin Configuration">                <description>If enabled, the
plugin configuration                    files will be regenerated when the application
server is started</description>                <classname>
com.ibm.websphere.plugincfg.initializers.AEPluginCfgService                </classname>
<classpath/>                <external-config-url/>                <enabled>false</enabled>
</custom-service-config>        </custom-service-config-list>
```

Configuration for the application server ends:

```
    </application-server>
```

Then the end tag for the node is placed, indicating that all the elements pertaining to the node have been addressed:

```
</node>
```

The security configuration comes next:

Skip ahead

```
<security-config security-cache-timeout="600" security-enabled="false">    <app-security-defaults>
<realm-name>Default</realm-name>        <challenge-type ssl-enabled="false">
<basic-challenge/>        </challenge-type>    </app-security-defaults>    <auth-mechanism>
<localos>            <user-id/>            <password>{xor}</password>        </localos>
</auth-mechanism></security-config>
```

Then comes the server group configuration:

Skip ahead

```
<server-group action="update" name="Test Server group">    <server-group-attributes>
<environment/>        <user-id/>        <group-id/>        <working-directory/>
<umask>18</umask>        <stdin/>        <stdout>stdout.txt</stdout>
<stderr>stderr.txt</stderr>        <process-priority>20</process-priority>
<maximum-startup-attempts>2</maximum-startup-attempts>        <ping-interval>60</ping-interval>
<ping-timeout>200</ping-timeout>        <ping-initial-timeout>300</ping-initial-timeout>
<selection-policy>roundrobinpreferlocal</selection-policy>
<debug-enabled>false</debug-enabled>        <transaction-timeout>120</transaction-timeout>
<transaction-inactivity-timeout>60000</transaction-inactivity-timeout>
<selection-policy>roundrobinpreferlocal</selection-policy>        <source-path/>        <jvm-config>
<initial-heap-size>64</initial-heap-size>            <max-heap-size>64</max-heap-size>
<generated-command-line-arguments>                -Xms64m -Xmx256m
</generated-command-line-arguments>            <system-properties/>
<additional-command-line-arguments/>            <debug-mode>false</debug-mode>
<debug-string/>            <run-hprof>false</run-hprof>            <hprof-args/>
<disable-jit>false</disable-jit>            <verbose-class-loading>false</verbose-class-loading>
<verbose-jni>false</verbose-jni>            <verbose-gc>false</verbose-gc>
<boot-classpath-replace/>            <boot-classpath-append/>            <boot-classpath-prepend/>
</jvm-config>        <custom-service-config-list/>    </server-group-attributes>    <clone
name="TestCLone">        <parent-node>subodh</parent-node>    </clone></server-group>
```

Several enterprise application entries come next;only one is shown here, with one EJB module and one Web module:

Skip ahead

```
<enterprise-application action="create" name="subodh/sampleApp">
<source-node>subodh</source-node>     <ear-file-name>
```

```
D:\WebSphere\AppServer\installableApps\sampleApp.ear     </ear-file-name>
<enterprise-app-install-info>         <node-name>subodh</node-name>         <ear-install-directory>
D:\WebSphere\AppServer/installedApps/sampleApp.ear         </ear-install-directory>
</enterprise-app-install-info>     <application-binding>         <authorization-table>
<role name="All Role">                <description>                    All Authenticated users in
the enterprise.                 </description>                 <all-authenticated-users
name="AllAuthenticatedUsers"/>             </role>             <role name="Everyone Role">
<description>Everyone in the enterprise.</description>                <everyone name="Everyone"/>
</role>         </authorization-table>         <run-as-map/>     </application-binding>
<ejb-module name="Increment">         <jar-file>Increment.jar</jar-file>
<module-install-info>             <application-server-full-name>
/NodeHome:subodh/EJBServerHome:Default Server/             </application-server-full-name>
</module-install-info>         <ejb-module-binding>             <data-source>
<jndi-name>jdbc/SampleDataSource</jndi-name>                 <default-user/>
<default-password/>             </data-source>             <enterprise-bean-binding
name="EnterpriseBeanBinding_1">                <jndi-name>IncBean</jndi-name>
</enterprise-bean-binding>         </ejb-module-binding>     </ejb-module>     ...     <web-module
name="default_app">         <war-file>default_app.war</war-file>
<context-root>/</context-root>         <module-install-info>
<application-server-full-name>                /NodeHome:subodh/EJBServerHome:Default Server/
</application-server-full-name>         </module-install-info>         <web-module-binding>
<virtual-host-name>default_host</virtual-host-name>         </web-module-binding>     </web-module>
... </enterprise-application>
```

The last item is the end tag for the export itself:

```
</websphere-sa-config>
```

6.6.0.2.1.1.2: XMLConfig - Example of a partial export

To do a partial export of your Websphere Administrative Domain configuration into an XML file, you need to create an XML file specifying the resources you would like to export. This partial file is then used as an input parameter in the XMLConfig export command line.

The partial XML file always begins with the following header lines:

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE websphere-sa-config   SYSTEM
"file:///$XMLConfigDTDLocation$$dsep$xmlconfig.dtd">
```

The contents of the Websphere administrative domain that you wish to extract into an XML file start with <websphere-sa-config> and end with </websphere-sa-config>. What goes in between these tags depends on what you want to export. This is an example of a partial XML file you would create to export the entire contents of an application server on a node named mynode:

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE websphere-sa-config   SYSTEM
"file:///$XMLConfigDTDLocation$$dsep$xmlconfig.dtd"><websphere-sa-config>    <node name="mynode"
action="locate">        <application-server name="Default Server" action="export">
</application-server>    </node></websphere-sa-config>
```

As an example of how you would do a partial configuration export into an XML file, we can name this file *PartialFile.xml*, place it in the appserver/bin directory, and run the following command from that directory:

```
XMLConfig -export NewExport.xml -adminNodeName mynode -partial PartialFile.xml
```

An XML file named *NewExport.xml* is then created in the appserver/bin directory with the following output:

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE websphere-sa-config   SYSTEM
"file:///$XMLConfigDTDLocation$$dsep$xmlconfig.dtd"><websphere-sa-config>    <node name="mynode"
action="locate">        <application-server action="update" name="Default Server">
<executable>java</executable>            <command-line-arguments/>            <environment/>
<user-id/>            <group-id/>            <working-directory/>            <umask>18</umask>
<stdin/>            <stdout>D:\WebSphere\AppServer/logs/Default_Server_stdout.log</stdout>
<stderr>D:\WebSphere\AppServer/logs/Default_Server_stderr.log</stderr>
<process-priority>20</process-priority>
<maximum-startup-attempts>2</maximum-startup-attempts>            <ping-interval>60</ping-interval>
<ping-timeout>200</ping-timeout>            <ping-initial-timeout>300</ping-initial-timeout>
<selection-policy>roundrobinpreferlocal</selection-policy>            <trace-specification/>
<trace-output/>            <transaction-log-file/>            <olt-enabled>false</olt-enabled>
<system-properties/>            <debug-enabled>false</debug-enabled>
<transaction-timeout>120</transaction-timeout>
<transaction-inactivity-timeout>60000</transaction-inactivity-timeout>
<thread-pool-size>20</thread-pool-size>            <security-enabled>false</security-enabled>
<admin-agent-ior/>            <cache-config>                <cache-size>2047</cache-size>
<cache-sweep-interval>1000</cache-sweep-interval>                <passivation-directory/>
</cache-config>            <log-file-spec/>            <performance-monitor-spec>
pmi=none:beanModule=maximum:connectionPoolModule=medium:
j2cModule=medium:jvmRuntimeModule=none:jvmpiModule=low:
servletSessionsModule=none:threadPoolModule=high:
transactionModule=none:webAppModule=maximum            </performance-monitor-spec>
<olt-server-host>localhost</olt-server-host>            <olt-server-port>2102</olt-server-port>
<selection-policy>roundrobinpreferlocal</selection-policy>            <source-path/>
<wlm-template-ior/>            <thread-pool-size>20</thread-pool-size>            <jvm-config>
<initial-heap-size>64</initial-heap-size>                <max-heap-size>64</max-heap-size>
<generated-command-line-arguments>-Xms64m -Xmx64m </generated-command-line-arguments>
<system-properties/>                <additional-command-line-arguments/>
<debug-mode>false</debug-mode>                <debug-string/>
<run-hprof>false</run-hprof>                <hprof-args/>
<disable-jit>false</disable-jit>                <verbose-class-loading>false</verbose-class-loading>
<verbose-jni>false</verbose-jni>                <verbose-gc>false</verbose-gc>
<boot-classpath-replace/>                <boot-classpath-append/>
<boot-classpath-prepend/>            </jvm-config>            <web-container>
<dynamic-cache-config>                    <enabled>false</enabled>
<cache-size>0</cache-size>                    <default-priority>0</default-priority>
</dynamic-cache-config>                <transport name="http">
<transport-host>*</transport-host>                    <transport-port>9080</transport-port>
<http-transport>                        <connection-timeout>5</connection-timeout>
<backlog-connections>50</backlog-connections>
<keep-alive-timeout>5</keep-alive-timeout>
<maximum-keep-alive>25</maximum-keep-alive>
<maximum-req-keep-alive>100</maximum-req-keep-alive>
<ssl-enabled>false</ssl-enabled>                    </http-transport>                </transport>
<thread-maximum-size>50</thread-maximum-size>
<thread-minimum-size>25</thread-minimum-size>
<thread-inactivity-timeout>10</thread-inactivity-timeout>
<thread-is-growable>true</thread-is-growable>                <session-manager>
```

```
<enable-security-integration>false</enable-security-integration>
<enable-ssl-tracking>false</enable-ssl-tracking>
<invalidation-schedule-first-hour>0</invalidation-schedule-first-hour>
<invalidation-schedule-second-hour>0</invalidation-schedule-second-hour>
<persistent-db2-row-size>0</persistent-db2-row-size>
<maximum-inmemory-session-count>1000</maximum-inmemory-session-count>
<write-contents>0</write-contents>                    <write-frequency>0</write-frequency>
<write-interval>0</write-interval>                    <enable-sessions>false</enable-sessions>
<enable-url-rewriting>false</enable-url-rewriting>
<enable-cookies>true</enable-cookies>
<enable-protocol-switch-rewriting>false</enable-protocol-switch-rewriting>
<cookie name="JSESSIONID">                        <domain/>
<maximum>-1</maximum>                        <path>/</path>
<secure>false</secure>                    </cookie>
<tuning-invalidation-time>30</tuning-invalidation-time>
<persistent-sessions>false</persistent-sessions>                    <data-source name="">
<default-user/>                        <default-password/>                    </data-source>
<using-multi-row>false</using-multi-row>                    <allow-overflow>true</allow-overflow>
</session-manager>            </web-container>            <orb-config>
<bootstrap-host-name/>                <bootstrap-port>900</bootstrap-port>
<comm-trace-enabled>false</comm-trace-enabled>
<connection-cache-maximum>240</connection-cache-maximum>
<connection-cache-minimum>100</connection-cache-minimum>                <external-config-url/>
<force-tunnel>whenrequired</force-tunnel>                <listener-port>0</listener-port>
<locate-request-timeout>180</locate-request-timeout>                <local-host-name/>
<lsd-host-name/>                <no-local-copies>false</no-local-copies>
<request-retries-count>1</request-retries-count>
<request-retries-delay>0</request-retries-delay>
<request-timeout>180</request-timeout>                <thread-pool-size>20</thread-pool-size>
<tunnel-agent-url/>                <rmi-remote-code-base/>
<ssl-listener-port>0</ssl-listener-port>            </orb-config>
<custom-service-config-list>                <custom-service-config name="Automatic Generation of
Plugin Configuration">                    <description>If enabled, the plugin configuration
files will be regenerated when the application                        server is
started</description>
<classname>com.ibm.websphere.plugincfg.initializers.AEPluginCfgService</classname>
<classpath/>                    <external-config-url/>                    <enabled>false</enabled>
</custom-service-config>            </custom-service-config-list>        </application-server>
</node></websphere-sa-config>
```

6.6.0.2.1.2: XMLConfig grammar

This section discusses the general structure of an XML element. For detailed information about each object, see the package summary file.See also the full export example topic.

Each object tag in the XML document contains:

- An object type, such as `virtual-host`
- A name attribute that identifies the particular resource on which to perform the action
- An action attribute that controls the behavior of the import or partial export operation

For example, an application server named MyAppServer has the following object tag:

```
<application-server name="MyAppServer" action="update">
```

In this case, the action is update. The following list describes all of the available actions.Unless otherwise stated, actions apply only to the import operation.

create

Adds the specified resource to the administrative domain. If the resource already exists, the create action is treated as an update action.When using this action, you must specify all required attributes for the object type.

update

Updates the properties of a specified resource. You need only specify the properties that you want to update.

However, if the resource does not exist, the update action becomes a create. In such a case, if some of the properties are not specified, an error occurs.

delete

Removes the specified resource and its children (recursive delete).

locate

Locates the specified resource. Use this action to provide the containment path to specific child resources. Applies to the partial export operation as well as the import operation.

export

Exports the configuration of the specified resource and its children to the output XML document. Applies only to the partial export operation.

start

Starts the specified resource (if applicable).

stop

Stops the specified resource, (if applicable).

stopforrestart

Stops the specified resource, and restarts it.(if applicable). Node only.

restart

Stops the specified resource and starts it again (if applicable).

enable

Makes the specified resource available for user requests (currently applies only to servlets and Web applications).

disable

Makes the specified resource unavailable for user requests (currently applies only to servlets and Web applications).

ℹ Some operational actions, such as start and stop, do not apply to all object types. In general, if the operation is supported in the WebSphere Administrative Console, it is supported in the XML import operation.

6.6.0.2.1.3: XMLConfig - Using the tool programmatically

The XMLConfig class is structured so that you can use it programmatically to retrieve information as Document or Element. The import/export facility can thus be included in a Java program, as well as being operated from a command line.

Creating platform-neutral configurations

For import and partial export operations, a variable substitution operation is performed on the input XML document, allowing you to create platform-neutral XML documents. The following variables are available:

$server_root$

Replace with the product installation directory, such as `C:\WebSphere\AppServer` on Windows NT. (This variable is not available for use on iSeries.)

$psep$

Replace with the path separator as specified in the operating system JDK.

- On Windows NT, it is **;** (semicolon)
- On the UNIX platforms and Linux, it is **:** (colon)

$dsep$

Replace with the directory separator as specified in the operating system JDK.

- On Windows NT, it is \ (backward slash)
- On the UNIX platforms and Linux, it is / (forward slash)

Security XML configurations

In Version 4.0, you can enter your own*custom user registry* entries (key/value pairs).All user-defined keysmust be delimited with a special prefix, `Custom_`.When you use the product GUI to add custom entries,the product adds the prefix for you.However, if you want to configure custom entries programmatically,you must add the prefix yourself.

The following example sets these properties:

```
groupsFile = c:\temp\groups.props    db_URL = jdbc:db2:customDB
```

A complete stanza for security configuration follows.The markup that corresponds to the needed property settings is shown in bold.

```
<security-config security-cache-timeout="150" security-enabled="true">   <app-security-defaults>
<realm-name>Default</realm-name>      <challenge-type ssl-enabled="false">
<basic-challenge/>      </challenge-type>   </app-security-defaults>   <auth-mechanism>
<ltpa-config>         <ltpa-password>{xor}B7rj9Lrj77rj7Q==</ltpa-password>
<ltpa-timeout>1200000</ltpa-timeout>         <custom-ur-config>            <user-id>bob</user-id>
<password>{xor}PTA9bg==</password>            <attribute name="Custom_groupsFile"
value="c:/temp/groups.props"/>            <attribute name="Custom_db_URL"
value="jdbc:db2:customDB"/>         </custom-ur-config>      </ltpa-config>
</auth-mechanism></security-config>
```

Javadoc for the tool

It is recommended that you refer to the Javadoc for the latest programmatic use of XMLConfig, and refer to the exported XML for the sample markup for repository objects.

Javadoc for the com.ibm.websphere.xmlconfig class and all of the related object classes resides in the apidocs directory:

```
installation_root\web\apidocs\package_and_class_name
```

See the package summary file for a list of class names, such as ApplicationServerConfig. The Javadoc is labeled by the class name preceded by the package name, com.ibm.websphere.

6.6.0.2.1.5: Troubleshooting XMLConfig

This article describes what to do if XMLConfig fails and displays the following message at its command line:

```
MLC0137E Make sure that adminserver is up and running.Additionally check -nameServiceHost and
-nameServicePort pair, if using remote admin server.
```

One or more of the following conditions is likely to have caused the problem:

- The admin server did not start properly.
- XMLConfig is being run remotely from the machine on which the product administrative serveris installed, and the -nameServiceHost parameter has not been set to the host name of the machine that contains the administrative server.
- The naming service port was changed from the default value of 900 on the application server and the -nameServicePort parameter was not set to the changed port value.
- The -nameServicePort parameter was defined, but the -nameServiceHost parameter was not.
- If the fully qualified host name (host name and domain) is specified for the -adminNodeName parameter, instead of the node name (short host name), expect this error:

  ```
  001.553 22f45b XMLConfig     X Unabled to export Virtual Host Data: {0}
  javax.naming.NameNotFoundException:
  ```

To resolve the problem:

1. Check whether the administrative server started successfully, as describedin the documentation for starting the administrative server.
2. If the XMLConfig tool is being run remotely with respect to the machine on whichthe administrative server is running, ensure that the -nameServiceHost and -adminNodeName parameters are set to the host name of the remote machine.
3. If the naming service port has been changed from the default value on the machineon which the administrative server is running, look in the administrative server configuration file for the parameter:

   ```
   com.ibm.ejs.sm.adminServer.bootstrapPort
   ```

   Set the XMLConfig -nameServicePort parameter to this port value.
4. If you use the -nameServicePort parameter, you must also use the -nameServiceHost parameter, even if you are running XMLConfig on the same machine asthe administrative server.
5. Modify the -adminNodeName parameter to use the node name, which can be found as your host name in your TCP/IP networking configuration and also in the WebSphere Administrative Console under **WebSphere Administrative Domain**.

6.6.0.2.2: WebSphere Control Program (wscp)

The WebSphere Control Program (wscp) is a command line client for the administrativeserver. It can be used to administer application servers, enterprise applications, andother types of WebSphere Application Server objects.

- WebSphere Control Program (wscp) overview introduces **wscp**, describes the relationship between the administrative console and **wscp**, and discusses the benefits and drawbacks of using **wscp**. It also lists the object types and services that **wscp** supports.
- Command syntax and usage covers basic and detailed syntax for the **wscp** interface, describes how to use **wscp** in command line and interactive modes and run Tcl scripts, and gives examples of how to use the **wscp** online help facility.
- Advanced usage of wscp goes into depth on how to use various **wscp** features, including abbreviations of **wscp** commands, lists, Tcl commands, operating system commands, error information, and tracing. It also describes how to use **wscp** to manipulate the Java Naming and Directory Interface (JNDI) context of objects, monitor performance, set security defaults, administer role-based security, and connect to remote servers.
- Example wscp commands, procedures and scripts contains examples of how to use various **wscp** commands, Tcl procedures, and Tcl scripts. Note that not all **wscp** operations are described in this section.
- Sample Tcl procedures and scripts lists the example Tcl scripts that are used in the **wscp** documentation.
- Migrating wscp scripts from version 3.5.x to version 4.0 summarizes the **wscp** changes that are most likely to affect existing **wscp** scripts.
- Using the WscpCommand interface describes how to use the WscpCommand class, which enables you to execute **wscp** operations from within Java applications.

6.6.0.2.2.1: WebSphere Control Program (wscp) overview

The WebSphere Control Program (**wscp**) is a command-line andscripting interface for administering resources in WebSphere ApplicationServer Advanced Edition. It is based on Tcl (tool commandlanguage). Tcl is a portable command language that provides programmingfacilities, such as variables, procedures, conditionals, list-processingfunctions, and looping constructs. The **wscp** interfaceextends Tcl by providing a set of commands for manipulating WebSphereobjects.

This section contains the following topics:

- The Administrative Console and wscp
- Benefits of using wscp
- Limitations of using wscp
- Supported object types
- Supported services and commands

The Administrative Console and wscp

The administrative server manages the contents and activities of a domainby maintaining a repository. A repository is the database ofinformation about all resources in a domain. The repository allowsadministration of a domain from any machine--all information is stored ina central location. The repository contains descriptive informationabout the applications that are configured to run in the domain. Forexample, it contains the names of all application servers, nodes, servergroups, and J2EE resources (such as JDBC providers), and their current state(running, defined, or stopped).

All administration takes place through the manipulation of objects in therepository. Each resource in a domain corresponds to an object in therepository. For example, when you create an application server, acorresponding application server object is created in the repository.

Both the administrative console and the **wscp** interface can beused to administer the resources in a domain. They both modify therepository in response to user commands, and they both reflect any changes tothe configuration and status of the domain. Console users access andmodify the repository via a graphical user interface; **wscp**users manipulate objects in the repository by executing commands orscripts. The console and the **wscp** interface arecompatible. The results of actions performed with **wscp** arereflected in the console interface, and vice versa.

Both the console and **wscp** can be used to do the following:

- Define, configure, and manage application servers, cloned applicationservers, and other application server resources from any node in thenetwork.
- Install applications (by using wizards in the console and install commandsand scripts in **wscp**).
- Perform daily administrative operations, such as starting and stoppingapplication servers and making changes to their configuration.
- Replicate application servers to improve performance or availability or tosimplify administration tasks (by defining and managing server groups).
- Track the occurrence of specific events by setting and enablingtracing.

Benefits of using wscp

The **wscp** interface provides a high-level command-lineadministrative tool with programming capabilities. With**wscp**, you can do the following:

- Use Tcl (tool command language) to extend and automate administrativeoperations through the use of scripts. For example, you can write ascript to start and stop servers or applications as a group. You canalso use scripts to create a basic configuration or (by invoking the XMLConfiguration Management Tool, XMLConfig) to store an existingconfiguration. The stored configuration can be used as a backup or tore-create a configuration on another machine.
- Use Tcl to create custom procedures for specialized administrative tasks,such as making identical modifications to a particular attribute throughoutthe domain, or displaying the values of select attributes that need to befrequently monitored.
- Issue UNIX or Windows NT commands from within a **wscp** sessionand incorporate them into scripts.

Limitations of using wscp

Although **wscp** offers many of the same administration tasks asthe WebSphere Advanced Administrative Console, the following differencesapply:

- New configurations or changes to existing configurations made with**wscp** are not immediately reflected in the console. You mustactively poll the administrative server repository to see the changes byselecting all or a portion of the Topology view in the console and then usingthe **Refresh** button.

  Similarly, if a resource is created or deleted in the console, the changewill not be immediately reflected in **wscp**. You mustexplicitly refresh all object references in the repository cache by issuing a**wscp** list operation for the instance's object type.(Alternatively, exiting and reinvoking **wscp** also refreshes thecache.)

  To avoid problems due to inconsistent cached information, it is best toavoid issuing concurrent operations on the same object from withinwscp and the console.

- All administrative tasks that can be done with the console can be donewith **wscp**, with the exception of higher-level aggregate tasks suchas those in the wizards--for example, creating an application.Instead, **wscp** scripts can be created to provide the samefunctionality as provided by the wizards.

Supported object types

In **wscp**, a resource is represented as an object type. Forexample, the object type ApplicationServer represents an application server(enterprise bean server) and the object type DataSource represents adatabase. Each object type has attributes (called *properties*in the console) that describe the characteristics of the object. Forexample, application server object attributes include Executable,CurrentState, and WorkingDirectory. Data source object attributesinclude DatabaseName, MinPoolSize, and MaxPoolSize. An object type canbe thought of as a template object that defines the characteristics of allobjects of that type. Instances of the object type represent specificobjects in the domain.

The **wscp** interface manipulates objects in the repository byperforming operations on them. Examples of operations are create,start, show, modify, and stop. The following is a list of object typessupported by **wscp**.

Note:

> Any object that represents a live entity outside of the repository (forexample, application servers), can be started and stopped with**wscp**. The attributes of several object types cannot bemodified. The message `EditorNotDefinedForThisProperty` isdisplayed when you attempt to access these attributes.

- ApplicationServer
- DataSource
- EnterpriseApp
- GenericServer
- J2CConnectionFactory
- J2CResourceAdapter
- JDBCDriver
- JMSConnectionFactory
- JMSDestination
- JMSProvider
- MailSession
- Module
- Node
- ServerGroup
- URL
- URLProvider
- VirtualHost

As the console does, **wscp** automatically provides defaultattributes and values for an object when the object is created. Youneed to specify values only for those attributes that are required and lackdefaults. If needed, you can override the default value for anyattribute when creating an object by specifying a value for thatattribute.

In the repository, objects are represented as attribute lists. Anattribute list is a collection of attribute-value pairs. The followingoutput shows how **wscp** displays the attributes of a DataSourceobject named ds1:

```
{Name ds1} {FullName /JDBCDriver:connDrv/DataSource:ds1/}{Description null} {ConfigProperties
{{URLjdbc:oracle:thin:@wssol2:1521:oraejs}}} {ConnTimeout 180}{DatabaseName WAS} {DefaultPassword
null} {DefaultUser null}{DisableAutoConnectionCleanup False} {IdleTimeout 1800}
{JNDINamejdbc/carddb} {MaxPoolSize 10} {MinPoolSize 1} {OrphanTimeout 1800}{StatementCacheSize 100}
```

See 6.6.0.2.2.3.2: Specifying lists in wscp commands for details on attribute lists.

In general, objects are created by specifying the object type, theoperation to be performed, the name of the object instance, and one or moreoptions. For example, to create a DataSource object, you specify theobject type (DataSource), the operation (create), the name of the objectinstance to create, and if attributes are required, the -attributeoption. The argument to the -attribute option is a list ofattributes and their values. For example, the following **wscp**command (in interactive mode) creates a DataSource object named ds1 that is aresource for the WAS database. The database driver must also bespecified:

```
wscp> DataSource create /JDBCDriver:DB2Driver/DataSource:ds1/-attribute {{DatabaseName WAS}}
```

In the following example, the ApplicationServer start command is used tostart an application server named myServer:

```
wscp> ApplicationServer start /Node:dev-pc/ApplicationServer:myServer/
```

See 6.6.0.2.2.2: Command syntax and usage for more example commands, detailed syntax, and anexplanation of the convention for object names.

The following example uses the Tcl **foreach** command to iteratethrough all application servers in a domain and stop them:

```
wscp> foreach ejbserver [ApplicationServer  list] {ApplicationServer stop $ejbserver}
```

See 6.6.0.2.2.3.3: Example use of wscp and Tcl and 6.6.0.2.2.4: Example wscp commands, procedures, and scripts.

Supported services and commands

In addition to objects, **wscp** supports several services andcommands that perform various administration tasks, such as configuringWebSphere Application Server, enabling global security, enabling tracing anddata collection, manipulating Java Naming and Directory Interface (JNDI)contexts, and displaying help on **wscp** commands. Theseservices are as follows:

- Context
- DrAdmin
- Help
- PmiService
- Remote
- SecurityConfig
- SecurityRoleAssignment
- XMLConfig

6.6.0.2.2.2: Command syntax and usage

This section contains basic and detailed syntax for the **wscp**interface, and describes how to use the **wscp** online helpfacility. The topics include:

- 6.6.0.2.2.2.1: Basic syntax
- 6.6.0.2.2.2.2: Invoking and terminating wscp
- 6.6.0.2.2.2.3: Authenticating to the administrative server
- 6.6.0.2.2.2.4: Connecting to local and remote nodes
- 6.6.0.2.2.2.5: Using command-line mode
- 6.6.0.2.2.2.6: Using interactive mode
- 6.6.0.2.2.2.7: Running scripts
- 6.6.0.2.2.2.8: Detailed syntax
- 6.6.0.2.2.2.9: Specifying object names
- 6.6.0.2.2.2.10: Using the wscp help facility

6.6.0.2.2.2.1: Basic syntax

The syntax for the **wscp** command is as follows:

```
wscp [ -h  ]  [ -c command ] [ -f  Tcl_file_name] [ -p  properties_file_name] [ -x  extension_class]
[ [ -- ] options ] [-node node_name]
```

The command options and arguments are as follows:

- The -h option displays help for the command.
- The -c option indicates command-line mode. The commandargument specifies a single command to be executed by **wscp**.This option can be repeated multiple times on the command line. See 6.6.0.2.2.2.8: Detailed syntax for more information on the commandargument.
- The -f option evaluates the specified file (script) of Tclcommands. Scripts can have arguments, which are specified following thedouble hyphen (- -). This option can be repeated multiple times on thecommand line. See 6.6.0.2.2.2.7: Running scripts.
- The -p option loads the specified properties file. This option canbe repeated multiple times on the command line.
- The -x option loads the specified Tcl extension class. This optioncan be repeated multiple times on the command line.
- The options following the double hyphen (- -) are used to setTcl argc and argv variables as specified. The double hyphen isnecessary only if an option can otherwise be mistaken for a **wscp**shell option.

For example, to invoke **wscp** and load theinit.tcl script, issue the following command:

```
wscp -f init.tcl
```

If no command-line options or files are specified, an interactive shell(Tcl interpreter) is invoked, which is terminated by the **exit**command. Command-line options not supported by **wscp**, orspecified after the double hyphen (- -) on the command line, are used to setthe Tcl argc or argv variables. These variables can be interpreted byTcl extensions or other commands.

The **wscp** shell evaluates Tcl commands in the order specified onthe command line, so any extensions must be loaded prior to invoking commandsdependent on those extensions. The following extensions areautomatically loaded:

- com.ibm.ejs.sm.ejscp.EjscpExtension,the class for the **wscp** commands
- com.ibm. ejs.sm.ejscp.ContextExtension,the class for manipulating Java Naming and Directory Interface (JNDI) contexts
- com.ibm.ejs.sm.ejscp.DrAdminExtension,the class for tracing the **wscp** client, WebSphere applicationservers, and the WebSphere administrative server
- com.ibm.ejs.sm.ejscp.RemoteExtension,the class for the **wscp** Remote extension.
- com.ibm.ejs.sm.ejscp.PmiServiceExtension,the class for monitoring application server performance.
- com.ibm.ejs.sm.ejscp.SecurityConfigExtension,the class for setting basic security defaults.
- com.ibm.ejs.sm.ejscp.SecurityRoleAssignmentExtension,the class for manipulating J2EE security roles.

After a command or script is executed, control is returned to theshell.

The **wscp** commands can be run as individual **wscp**invocations from the operating system prompt (command-line mode), as scripts,or interactively in a **wscp** session (interactive mode).

6.6.0.2.2.2.2: Invoking and terminating wscp

To invoke **wscp**, use one of the following procedures.

- To run **wscp** with its default extensions loaded, enter one ofthe following at the command prompt:
  - On Unix systems:

    ```
    > wscp.sh
    ```
  - On Window systems:

    ```
    C:\ wscp.bat
    ```
- To run **wscp** with different extensions loaded, do thefollowing:
  1. Set up the CLASSPATH environment variable, including the appropriate JARfiles.
  2. Run Java, invoking the following:
     - The **wscp** Tcl shell (thecom.ibm.ejs.sm.ejscp.WscpShell class)
     - Desired extensions (the -x option). Extensions can also be loadedinteractively.
  3. Optionally, set properties to enable tracing.

When **wscp** is invoked without specifying the -p option, theproperty file homeDirectory/.wscprc is loaded if it exists,where homeDirectory is the value of the Java propertyuser.home. The .wscprc file can be used to automaticallyload settings for various properties, amending or replacing system propertiesalready defined. For example, you can modify the values of thewscp.hostName and wscp.hostPort properties to connect to aremote node. See 6.6.0.2.2.2.4: Connecting to local and remote nodes for details.

Note that the value of the Java user.home property can differdepending on the SDK version level. On Windows NT systems, the SDK1.2.2 used by WebSphere Application Server sets theuser.home property based on the value of the USERPROFILE environmentvariable.

The following example .wscprc file specifies values for variousproperties.

```
## primaryNode is used with qualifyHomeNames (default=hostName property)## wscp.primaryNode=pc-dev1## hostName is used for
com.ibm.CORBA.BootstrapHost#wscp.hostName=pc-dev1## hostPort is used for com.ibm.CORBA.BootstrapPort (default=900)##
wscp.hostPort=900## wscp.remotePasswordFile (default : uses random number for password)# wscp.remoteConnectionTimeout (default =
300 seconds)# wscp.remoteConnectionsAllowed (default = false)# wscp.remoteHostListAccept (default = null, colon-separated
string)# wscp.remoteHostListReject (default = null, colon-separated
string)#wscp.remotePasswordFile=C:/TEMP/passwordwscp.remoteConnectionTimeout=300wscp.remoteConnectionsAllowed=truewscp.remoteHostListAccept=abc.def.com:abc.comwscp.remoteHostListReject=dhcp-198-7.abc.def.com##
set traceString to enable tracing## wscp.traceString=com.ibm.ejs.sm.client.ui.desc.*=all=enabled#
wscp.traceString=com.ibm.ejs.sm.ejscp.*=all=enabled### Typically, these need to be set if only ContextExtension is loaded#
(otherwise, EjscpExtension sets them as noted above)## com.ibm.CORBA.BootstrapHost=pc-dev1# com.ibm.CORBA.BootstrapPort=900#
java.naming.factory.initial=com.ibm.ejs.ns.jndi.CNInitialContextFactory
```

The **exit** command terminates an **wscp** interactivesession.

6.6.0.2.2.2.3: Authenticating to the administrative server

If security is enabled for your administrative server, you must prepare the**wscp** client so that it can authenticate to the server. To dothis, perform the following steps:

- Make a copy of the properties file sas.client. props.This file is located in the subdirectory named properties, in the directorywhere you installed the product--for example,C:\WebSphere\AppServer\properties. Rename the file--forexample, you can name the file sas.wscp.props--and edit oradd the following property-value pairs:

  ```
  com.ibm.CORBA.loginSource=propertiescom.ibm.CORBA.loginUserid=your_user_idcom.ibm.CORBA.principalName=NT_domain/user_idcom.ibm.CORBA.loginPassword=your_password
  ```

  The value `properties` is the only option for thecom.ibm.CORBA.loginSource property. The value`prompt` (prompting for a user ID) is not supported for**wscp**.
- If you are using digital certificates, you must also enableaccess to the server key stores. A dummy key store file is provided as aWebSphere Application Server installation option. The dummy key storeis not intended for use in a production environment. See 5.5: Certificate-based authentication, for moreinformation on using certificates in WebSphere Application Server.
- Add the following line to your .wscprc file:

  ```
  com.ibm.CORBA.ConfigURL=URL of properties file
  ```

  An example URL isfile:///C:/WebSphere/AppServer/properties/sas.wscp.props.Alternatively, you can add the following option to the Java command line(preceding the application class name):

  ```
  -Dcom.ibm.CORBA.ConfigURL=URL of properties file
  ```

6.6.0.2.2.2.4: Connecting to local and remote nodes

When you invoke **wscp**, you are automatically connected to theadministrative server running on the local machine. (The host name isdefined in the default .wscprc property file.) To connect to aremote node (that is, to specify the host name of a remote administrativeserver), set the property wscp.hostName to the name that a domain namesystem (DNS) server can resolve. For example, the following propertyspecifies that the administrative server running on myHost is to beused:

```
wscp.hostName=myHost
```

This property can be set in the default .wscprc file. You can alsouse the -p option of the **wscp** command to specify a differentproperty file to use. See 6.6.0.2.2.2.2: Invoking and terminating wscp for information on the .wscprc property file.

6.6.0.2.2.2.5: Using command-line mode

To execute an individual **wscp** command in command-line mode, usethe -c option followed by the command. The following examples execute a**wscp** command on the local node. On Windows NT systems,enclose the command in double quotation marks ("). On UNIX systems,enclose the command in single quotation marks (').

```
C:\> wscp -c "ApplicationServer list"
```

```
% wscp.sh -c 'ApplicationServer list' -c 'Node list'
```

The -c option can be repeated multiple times on the command line.

6.6.0.2.2.2.6: Using interactive mode

To invoke an interactive **wscp** session, type **wscp** andpress Return. The following prompt signals that you have enteredinteractive mode:

```
wscp>
```

At the interactive prompt, enter a **wscp** command;**wscp** executes the command, displays the result, and is ready toaccept another command. Use the **exit** command to terminate a**wscp** interactive session.

In an interactive **wscp** session, you can break the line of a**wscp** command after a left brace ({) or after typing a backslash (\ ). The **wscp** session displays a question mark (?) prompt,and you can continue typing the command.

Note:

> The WscpCommand class allows you to embed interactive **wscp**operations in a Java application. See 6.6.0.2.2.7: Using the wscpCommand interface for more information on using this class.

6.6.0.2.2.2.7: Running scripts

To execute a **wscp** script from the command line, specify the -foption and the name of the executable file that contains the script.For example, the following command executes the script in themodEnv.tcl file.

```
C:\> wscp -f  modEnv.tcl
```

To execute a script from within an interactive **wscp** session, usethe Tcl **source** command to source the script. The followingcommand runs the script in the file myScript.tcl:

```
wscp> source myScript.tcl
```

6.6.0.2.2.2.8: Detailed syntax

The detailed syntax for the **wscp** command argument is asfollows:

```
object_type operation [object_name] [ -option  [value] | -attribute  attribute_list]
```

A command argument can be issued immediately following the -coption of the **wscp** command, or at the **wscp** interactiveprompt, as shown in the following examples.

```
C:\> wscp -c  "ApplicationServer list"wscp> ApplicationServer list
```

When using the -c option, enclose the command in double quotation marks (")(Windows NT systems) or in single quotation marks (') (UNIXsystems).

Note:

All object type names, object instance names, and attributes are casesensitive.

The arguments are as follows:

- The object_type argument specifies the name of an objecttype. Examples are application servers (the object typeApplicationServer) and JDBC providers (the object type JDBCDriver).Object type names can be abbreviated to the shortest unique string (ininteractive mode only).
- The operation argument specifies the action to be performed onthe object. Examples of **wscp** operations are create, modify,show, and start.
- The object_name argument specifies the name of an object on whichthe action is to be performed. An object_name indicates aparticular instance of an object type--for example, ejbserver1.You must use fully qualified object names. See 6.6.0.2.2.2.9: Specifying object names for details on object naming.
- The option argument specifies a qualifier that controls theprecise behavior of an **wscp** operation. For example, usingthe -force option with a stop operation performs a risky operation orcompletes an operation without confirmation. Some options take values,which further define the operation's behavior. For example, the-constraint option of list operations takes an attribute list as itsvalue. The use of this option causes **wscp** to display onlythose object instances that contain the attribute-value pair specified.
- The -attribute option specifies one or more properties of anobject. Examples of attributes are the DatabaseName and IdleTimeoutattributes of a DataSource object. Properties are specified in a listof attribute-value pairs. For example, the following is an attributelist for a DataSource instance:

  ```
  {Name ds1} {FullName /JDBCDriver:j1/DataSource:ds1/}{ConfigProperties {}} {Conn Timeout 180}
  {DatabaseName WAS}{DefaultPassword {}} {DefaultUser {}} {Description
  {}}{DisableAutoConnectionCleanup False} {IdleTimeout 1800} {JNDINamejdbc/ds 1} {MaxPoolSize 10}
  {MinPoolSize 1} {OrphanTimeout 1800}{StatementCacheSize 100 }
  ```

  See 6.6.0.2.2.3.2: Specifying lists in wscp commands for details.

6.6.0.2.2.2.9: Specifying object names

In the console, simple names such as myAppServer or dataSource1 are used tospecify object names. However, in the repository, object names arestored as fully qualified names. Fully qualified names are requiredwhen using **wscp**. Fully qualified names reflect anobject's containment hierarchy. Containment determines howresources are related to one another.

For example, a node can contain application servers. An enterpriseapplication contains one or more modules.

Containment represents a hierarchical naming structure. Thisstructure prevents name clashes between objects that have the same name butbelong to different containment hierarchies. A resource's positionin the containment structure is used to generate the resource's full,unique name within the repository. Examples of containment hierarchiesare:

- Node/ApplicationServer
- JDBCDriver/DataSource
- URLProvider/URL

Containment is also used for efficient operations: starting acontained resource starts the resources above it in the hierarchy.Containment is also used to impose restrictions on relationships.

Fully qualified names have the form/objectType**:**objectInstance/.. . /. For example, if the application servernamed AppServ1 runs on the node named Node1, the repository name of theapplication server reflects this hierarchy. The name of the applicationserver, /Node:Node1/ApplicationServer:AppServ1/, is a fullyqualified name.

Some objects, for example Node objects, do not belong to a containmenthierarchy. These objects are known as *root types*. Anexample of a fully qualified name for the Node object named myNode is/Node:myNode/.

Note:

> You must use fully qualified names in **wscp** commands andscripts. Due to the syntax of fully qualified names, an object namecannot contain a colon (:). Fully qualified names can containspaces; if spaces are used, the name must be enclosed in braces or doublequotation marks (") so that Tcl parses the name as a single argument.

To reduce typing and for readability of scripts, use variables to store afull or partial name of an object. For example, the following commandsset variables for the name of a node, server, container, and bean.These variables can then be used in building other object names:

```
set node  "Node:my-Pc"set serv "/${node}/ApplicationServer:myServer/"
```

The braces are required so that the remaining part of the object name isnot considered part of the variable name.

Elsewhere, the variable names can be used as shown in the followingexample:

```
wscp> ApplicationServer create $serv
```

The containment operation lists the containment hierarchy for a specifiedobject type. This operation is useful for determining the requiredelements in a fully qualified name. The syntax is as follows:

```
object_type containment
```

The following example command displays the containment hierarchy for theModule object type:

```
wscp> Module containmentEntepriseApp Module
```

6.6.0.2.2.2.10: Using the wscp help facility

You can access **wscp** online help in varying levels of detail.Either of the following commands displays a summary of syntax for the helpfacility.

```
C:\> wscp -c "Help"
wscp> Help
```

The following topics contain examples of how to obtain help:

- Help on object types
- Help on operations
- Help on attributes

Help on object types

To obtain help on an object type, use the following syntax:

```
object_type help  [operation [ -verbose ] ]
```

When no arguments are specified, this command displays a brief descriptionof all valid operations for the object type. Depending on the argumentspecified, you can access help on the following:

- The operation argument displays valid options for the specifiedoperation.
- The -verbose option provides details on the options for anoperation.

For example, the following command requests help on the ApplicationServerobject. The command displays the names and descriptions of all validoperations for that object type.

```
wscp> ApplicationServer helpThe following actions are available for ApplicationServerattributes
Display the attributes of the objectcontainment         Display the containment hierarchy for the
objectcreate                 Create the specified objectdefaults     Display or set attribute
defaultshelp                  Display this help messagelist                   Display all the
instances of this typemodify                    Modify the attributes of the specified
objectoperations         List all the actions available on the object typeremove
Remove the specified objectshow                 Display the attributes of specified objectstart
Start the specified objectstop                  Stop the specified object
```

The following command displays help for the list operation ofApplicationServer object types:

```
wscp> ApplicationServer help listApplicationServer list [-constraint <attribute list>] [-recursive]
```

The following command displays verbose help for the list operation:

```
wscp> ApplicationServer help list -verboseApplicationServer listThe following options are available
for list[-constraint <attribute list>]      Display only objects which satisfy the
constraints[-recursive]                  Display subtype objects
```

The following command displays the valid options for the clone operation ofServerGroup objects:

```
wscp> ServerGroup help cloneServerGroup clone <name> -cloneAttrs <attribute list> -node <name>
```

The following command displays verbose help for the clone operation:

```
wscp> ServerGroup help clone -verboseServerGroup clone <name> -cloneAttrs <attribute list> -node
<name>The following options are available for clone -cloneAttrs <attribute list>    Attribute list
for the clone-node <name>                    Node name
```

Help on operations

To obtain help on an object type's operations, use the followingsyntax:

```
object_type operations
```

This command displays a list of all valid operations for the given objecttype.

The following examples display a list of all valid operations for theServerGroup and EnterpriseApp object types:

```
wscp> ServerGroup operationsattributes clone containment create help list listClones modify
operations remove removeClone show showAttrs start stopwscp> EnterpriseApp operationsattributes
containment create defaults help list modify operations remove show start stop install listmodules
listnodes showdeploymentinfo
```

Help on attributes

To obtain help on an object type's attributes, use the followingsyntax:

```
object_type attributes [ -cloneOnly ]   [ -modelOnly ]  [ -optional ][ -readOnly ]  [ -readWrite ]
[ -repository ]  [ -required ]  [ -runtime ]   [ -server ]  [ -startup ]
```

When no options are specified, this command displays a list of all validattributes for the given object type. Various options allow you todisplay a subset of attributes. The options are as follows:

- -cloneOnly. Displays only those attributes associated withclones of the object type.
- -optional. Displays only those attributes that do not require avalue.
- -readOnly. Displays only those attributes that can be read but notchanged directly. An example of a read-only attribute isCurrentState.
- -readWrite. Displays only those attributes that can be viewed andmodified. A read/write attribute can be changed at any time, and thechanges take effect immediately.
- -repository. Displays only those attributes that are stored in therepository. (The values of some attributes are short-lived andtherefore are not stored permanently, for example, a process ID.)
- -required. Displays only those attributes for which valuesmust be specified in order for an object to be created.
- -runtime. Displays only those attributes that come intoexistence after an object has been started. Run-time attributes areread-only attributes not stored in the repository. Examples of run-timeattributes are ProcessID and StartTime.
- -server. Displays only those attributes that reside in a server(run-time attributes of a server, for example, the ProcessIDattribute).
- -startup. Displays only those attributes that can be changedwhile a server is running but do not take effect until the next time theobject is started. Examples of startup attributes are Executable andEnvironment.

The following commands display all valid attributes of the Node andGenericServer object types:

```
wscp> Node attributesName FullName CurrentState DesiredState StartTime HostName HostSystemType
ProcessId InstallRoot PathMapwscp> GenericServer attributesName FullName CurrentState DesiredState
StartTime ProcessId Environment SelectionPolicy Executable ExecutableActive CommandLineArgs
CommandLineArgsActive EnvironmentActive UserId UserIdActive GroupId GroupIdActive WorkingDirectory
WorkingDirectoryActive Umask UmaskActive Stdin StdinActive Stdout StdoutActive Stderr StderrActive
MaxStartupAttempts ProcessPriority ProcessPriorityActive PingInterval PingTimeout PingInitialTimeout
```

The following command displays only required attributes of the DataSourceand JDBCDriver object types:

```
wscp> DataSource attributes -requiredName wscp> JDBCDriver attributes -requiredName ImplClass
```

The following command displays only those attributes of ApplicationServerobjects that apply to server groups and clones.

```
wscp> ApplicationServer attributes -modelOnlySelectionPolicywscp> ApplicationServer attributes
-cloneOnlyName FullName
```

See Printing an object's attributes for a custom Tcl procedure that provides a shortcut methodof displaying various combinations of attributes for one or moreobjects.

6.6.0.2.2.3: Advanced usage of wscp

This section contains the following topics on using **wscp** andTcl:

- 6.6.0.2.2.3.1: Using abbreviations in wscp commands
- 6.6.0.2.2.3.2: Specifying lists in wscp commands
- 6.6.0.2.2.3.3: Example use of wscp and Tcl
- 6.6.0.2.2.3.4: Using wscp and operating system commands
- 6.6.0.2.2.3.5: Obtaining status and error information
- 6.6.0.2.2.3.6: Use of qualified home names in the administrative server
- 6.6.0.2.2.3.7: Tracing the administrative server, application servers, and the wscp client
- 6.6.0.2.2.3.8: Enabling tracing with DrAdmin
- 6.6.0.2.2.3.9: Manipulating the JNDI context of objects
- 6.6.0.2.2.3.10: Monitoring performance
- 6.6.0.2.2.3.11: Setting global security defaults
- 6.6.0.2.2.3.12: Managing security roles
- 6.6.0.2.2.3.13: Connecting to remote servers
- 6.6.0.2.2.3.14: Where to find more information about Tcl

6.6.0.2.2.3.1: Using abbreviations in wscp commands

In interactive mode, the **wscp** interface allows abbreviations forthe first word of a command only (that is, for object types). All otheroptions must be fully specified. If an abbreviation is ambiguous,**wscp** returns an error that includes a list of all matches.Entering a question mark (?) at the `wscp>` prompt lists allsupported commands and object types (including Tcl commands).

In the following example, the word JMS is ambiguous:

```
wscp> JMS listAmbiguous command name  "JMS":  JMSConnectionFactory, JMSDestination, JMSProvider
```

To resolve the ambiguity, you must provide as many characters as needed touniquely identify the desired object type--for example, JMSD.

6.6.0.2.2.3.2: Specifying lists in wscp commands

Some **wscp** options take Tcl lists as values (notably the -attributeoption of create and modify operations). A Tcl list is an orderedcollection of elements, where each element can be a string, a number, oranother list. Lists can be delimited by either double quotation marks(" ") or braces ({ }). The following example sets the variable x to athree-element list whose first element is the string "account", whose secondelement is "term", and whose third element is itself a list containing the twostrings "maturity" and "date".

```
set x {account term {maturity date}}
```

In **wscp**, an attribute list is a list of attribute-value pairs (aTcl list of lists). The following command modifies the PingInterval andPingTimeout attributes of an application server. The argument to the-attribute option is a list containing two elements. Bothelements are themselves lists.

```
wscp> ApplicationServer modify /Node:dev-pc/ApplicationServer:myServer/  -attribute {{PingInterval
120}  {PingTimeout 240}}
```

Note that Tcl parsers sometimes require a space between list items.

An attribute list containing only one attribute-value pair must also beenclosed in a list. For example, in the following command, theattribute-value pair {PingTimeout 240} must be nested within braces:

```
wscp> ApplicationServer  modify /Node:dev-pc/ApplicationServer:myServer/  \-attribute {{PingTimeout
240}}
```

In addition to the modify operation, the show operation has an -attributeoption that expects a Tcl list as its argument. If you specify only oneattribute to be displayed, you can use either a string or a Tcl list.Tcl interprets both the string and the Tcl list of one element asequivalent--for example, the arguments JarFile and {JarFile} areequivalent.

As in **wscp**, Tcl commands consist of one or more words separatedby spaces. A Tcl word that contains spaces must be enclosed in eitherbraces ({ }) or double quotation marks (" "). The use of braces anddouble quotation marks is similar. Both braces and double quotationmarks can be placed around a word that contains embedded spaces.However, using braces and double quotation marks differs in tworespects. First, unlike double quotation marks, braces can nest.Also, no substitutions occur inside braces, as they do inside double quotationmarks. All characters between braces are passed as an argument to acommand or procedure, without any special processing. Substitutions canoccur later if the argument is evaluated again.

Object names in **wscp** can contain spaces and therefore must beenclosed in either braces or double quotation marks. If object namesare enclosed in braces, no evaluation takes place and everything in braces ispassed to the command as an argument. If substitution must take place,for example, when the object name being passed as an argument contains avariable that must be expanded to form the name, then the object name must beenclosed in double quotation marks. The following example commandsdemonstrate the use of braces and double quotation marks in each case:

```
# Braces used because object name contains spaces# Object name is passed to the command as an
argument# and no substitution takes place wscp> ApplicationServer create
{/Node:dev-pc/ApplicationServer:My Server/} wscp> ApplicationServer
list{/Node:dev-pc/ApplicationServer:Default Server/} {/Node:dev-pc/ApplicationServer:Appl EJB
Server/} {/Node:dev-pc/ApplicationServer:Model EJB Server/} {/Node:dev-pc/ApplicationServer:Model
EJB Server2/} {/Node:dev-pc/ApplicationServer:Bean EJBServer/} {/Node:dev-pc/ApplicationServer:My
Server/}  # Double quotation marks used because substitution must take place# Assumes the NODE
constant is defined as /Node:dev-pc/ wscp> ApplicationServer create "${NODE}ApplicationServer:Test
Server/" wscp> ApplicationServer list{/Node:dev-pc/ApplicationServer:Default Server/}
{/Node:dev-pc/ApplicationServer:Appl EJB Server/} {/Node:dev-pc/ApplicationServer:Model EJB Server/}
{/Node:dev-pc/ApplicationServer:Model EJB Server2/} {/Node:dev-pc/ApplicationServer:Bean EJBServer/}
{/Node:dev-pc/ApplicationServer:My Server/} {/Node:dev-pc/ApplicationServer:Test Server/}
```

Lists are a special case of quoted strings, and Tcl quoting conventions canbe nonintuitive. Be particularly careful when specifying strings thatcontain spaces. For example, the Environment attribute of applicationservers is a list of strings of the formname=value. If the value contains spaces,braces are required. The following command modifies the existing valueof an Environment attribute:

```
wscp> ApplicationServer modify /Node:dev-pc/ApplicationServer:Server1/ -attribute \{{Environment
{{VARIABLE=word1 word2}}}}
```

The following command creates an application server and specifies twovalues for the Environment attribute. Note that the PATH environmentvariable does not require braces because it contains no spaces.

```
wscp> ApplicationServer create /Node:dev-pc/ApplicationServer:Server2/ -attribute \{{Environment
{PATH=/myPath {OTHERVARIABLE=word1 word2}}}}
```

The following commands display the Environment attributes of both serversin the above example:

```
wscp> ApplicationServer show /Node:dev-pc/ApplicationServer:Server1/ -attribute
Environment{Environment {{VARIABLE=word1 word2}}}wscp> ApplicationServer show
/Node:dev-pc/ApplicationServer:Server2/ -attribute Environment{Environment {PATH=/myPath
{OTHERVARIABLE=word1 word2}}}
```

Note that when you modify an attribute, you are replacing the existingvalue with a new value. If you want to add to or replace part of anexisting value (for example, change a path name), you must use a customprocedure. See Modifying an Environment attribute (modEnv procedure) for an example procedure named modEnv. The procedurecan be used to modify the Environment attribute of one or more servers in adomain. It modifies one element of the list and retains the values ofthe other elements.

Other attributes that take a list of strings as their arguments are theCommandLineArgs and SystemProperties attributes of application serverobjects.

6.6.0.2.2.3.3: Example use of wscp and Tcl

The **wscp** interface consists of **wscp** operations andbuilt-in Tcl commands. Tcl provides a portable method of controllingand extending **wscp** administrative operations. You can usenative Tcl commands for creating and executing new commands (the**proc** and **eval** commands), conditionalizing andcontrolling the flow of execution (if statements and loops), and handlingerrors and exceptions (the **catch** command).

The Tcl **foreach** looping command iterates over all elements in alist. In the following example, the **foreach** command is usedto iterate over all instances of ApplicationServer objects in a domain andthen stop each server object. The square brackets ([ ]) invoke commandsubstitution--the result of the ApplicationServer list operation (a listof server names) is used as the list argument to the **foreach**command. In turn, each ApplicationServer server name is substituted forthe variable $ejbserver.

```
wscp> foreach ejbserver [ApplicationServer  list] \ {puts "stopping $ejbserver...";
ApplicationServer stop $ejbserver}
```

As part of many server administration tasks, you often need to monitor thevalues of one or more server attributes. To do so, you can create andrun a procedure that displays the attributes of interest. The followingis an example procedure called showServerStatus, which displays the currentstate for each application server in a domain. As written, theshowServerStatus procedure displays the Name and CurrentState attributes ofall application servers. You can modify the procedure to displayadditional or different attributes.

```
proc showServerStatus {} {      puts "\nStatus of servers in the domain:\n"          foreach
ejbserver [ApplicationServer list] {          set serverInfo($ejbserver) [ApplicationServer show
$ejbserver -attribute \          {Name CurrentState}]          puts $serverInfo($ejbserver)
}}
```

The following example demonstrates output of the showServerStatusprocedure:

```
wscp> showServerStatusStatus of servers in the domain:{Name {Default Server}} {CurrentState
{Initialization Failed}}{Name {Appl EJB Server}} {CurrentState Running}{Name {Model EJB Server}}
{CurrentState Stopped}{Name {Model EJB Server2}} {CurrentState Stopped}{Name {Bean EJB Server}}
{CurrentState Running}{Name {My Server}} {CurrentState Stopped}{Name {Test Server}} {CurrentState
Stopped}
```

The following procedure displays the attributes of a specified objectinstance, formatting them so that the attributes are displayed one per line,without enclosing braces.

```
proc display {type name} {      set attrs [$type show $name]    foreach attr $attrs {           puts
$attr      }}
```

The following example demonstrates output from the displayprocedure. The procedure is used display the attributes of theDataSource object named testDataSource:

```
wscp> display DataSource /JDBCDriver:OracleJDBC/DataSource:testDataSource/Name
testDataSourceFullName /JDBCDriver:OracleJDBC/DataSource:testDataSource/ConfigProperties {URL
jdbc:oracle:thin:@wssol:1521:oraejs}ConnTimeout 300DatabaseName WASDisableAutoConnectionCleanup
FalseIdleTimeout 1800JNDIname testDataSourceMaxPoolSize 30MinPoolSize 3OrphanTimeout
1800StatementCacheSize 100
```

6.6.0.2.2.3.4: Using wscp and operating system commands

All UNIX and Windows NT commands can be issued from within a **wscp**session. In interactive mode, Tcl executes operating system commandswith an explicit **exec** command. The **exec** commandcreates one or more subprocesses and waits until they complete beforereturning. If you wish to disable this behavior and instead have Tclsearch your UNIX or Windows NT PATH environment variable for command namesbefore checking whether they are abbreviations, enter the following command ininteractive mode:

```
wscp> unset auto_noexec
```

To check the definition of the variable and to reset it to true, enter thefollowing in interactive mode:

```
wscp> set auto_noexec wscp> set auto_noexec true
```

The Tcl **info globals** command returns the names of all globalvariables currently defined.

6.6.0.2.2.3.5: Obtaining status and error information

Successful **wscp** commands return a result, typically either alist of information or an empty string. When commands are runinteractively, the Tcl result is displayed by the **wscp**shell. Failed **wscp** commands raise a TclException, which,unless there is a **catch** clause, stops the execution of theenclosing procedure.

When an exception is caught by **wscp**, the stack trace is appendedto the Tcl variable errorInfo. You can view the stack trace by issuingeither of the following commands:

```
wscp> puts $errorInfowscp> set errorInfo
```

The Tcl variable errorCode is set when **wscp** commands areexecuted. A nonzero value represents an error returned by thecommand.

Note:

> Interactive users and writers of Tcl scripts must check this variable todetermine whether a command succeeded or failed.

The errorCode variable is set to an integer (0 indicates success).The static statusToString method ofcom.ibm.ejs.sm.ejscp.WscpStatus can be usedto translate an errorCode value. For example, the following Tclprocedure takes an errorCode value and translates it:

```
# Converts a WscpStatus to its corresponding string translation#proc statusToString {{status -1}} {
global errorCode    if {$status == -1 && $errorCode != "NONE"} {set status $errorCode}    java::call
com.ibm.ejs.sm.ejscp.WscpStatus statusToString $status}
```

Note that this Tcl procedure can be used to convert any error code; ifan error code argument is not provided, the procedure converts the currentvalue of the global errorCode variable.

The sample Tcl script init.tcl contains the statusToString procedureand other useful procedures for debugging. See 6.6.0.2.2.4.1: Initialization and general-purpose procedures.

6.6.0.2.2.3.6: Use of qualified home names in the administrative server

The following property is used by the administrative server to set up theJava Naming and Directory Interface (JNDI) namespace. It specifieswhether enterprise bean homes are looked up in the initial context or in aspecified subcontext. If this property is set to true, bean homes arelooked up in the specified subcontext.

```
com.ibm.ejs.sm.adminServer.qualifyHomeName
```

Both the administrative server and **wscp**, by default, have thisproperty set to true. If the administrative server is running withqualified home names, **wscp** must also run with qualified homenames. The following property controls the use of qualified home namesin **wscp**:

```
wscp.qualifyHomeName
```

Note:

> This property is not related to the fully qualified names used for objectinstances.

You can disable the use of qualified home names in an administrative serverby setting thecom.ibm.ejs.sm.adminServer.qualifyHomeNameproperty to false in the admin.config file. This file is locatedin the directory where Advanced Application Server was installed (typically,on Windows NT systems,installation_drive\Websphere\Appserver\bin). If this propertyis set to false, the corresponding property in **wscp** must also beset to false.

6.6.0.2.2.3.7: Tracing the administrative server, application servers, and the wscp client

Use either of the following methods to enable tracing for a **wscp**client:

- Set trace properties (on the Java command line or in the .wscprcfile).
- Load thecom.ibm.ejs.sm.ejscp.DrAdminExtensionextension and then set trace properties by using the -setTrace option of theDrAdmin local operation. See 6.6.0.2.2.3.8: Enabling tracing with DrAdmin for details.

The following example lines set trace properties. The first lineenables all tracing for all of the wscp classes; the second line enablestracing for only the wscp.commands classes.

```
wscp.traceString=com.ibm.ejs.sm.ejscp.*=all=enabledwscp.traceString=com.ibm.ejs.sm.ejscp.commands.*=all=enabled
```

You can also use the DrAdmin extension to enable and set tracing for theadministrative server and for application (enterprise bean) servers.

If you are not using the DrAdmin extension, you must use the console toenable tracing for an application server. You can, however, use**wscp** to set trace for these servers by modifying the TraceSpecattribute of the server object.

See Administering the product messages,logs, and traces for additional information on tracing.

6.6.0.2.2.3.8: Enabling tracing with DrAdmin

In addition to the EjscpExtension class, WebSphere Advanced ApplicationServer includes the **wscp** DrAdmin operations for tracing. TheDrAdmin operations can be used to trace an administrative server or anyapplication server in a domain.

- The DrAdmin local operation traces the **wscp** clientitself.
- The DrAdmin remote operation traces the administrative server or anapplication server (the server can be local, running on the same machine as**wscp**, or it can be running on a remote machine).

The syntax for both commands is as follows:

```
DrAdmin local  [-setTrace trace_spec] [-setRingBufferSize size] [-dumpRingBuffer file_name]
[-dumpState file_name]DrAdmin remote  server_port [-serverHost host_name [-setTrace trace_spec]
[-dumpRingBuffer file_name] [-dumpState file_name[-setRingBufferSize string] [-stopServer]
[-stopNode] [-dumpThreads]
```

The arguments and options are as follows:

- server port. A port number is displayed as an Auditmessage in the Console Messages window when each server starts and when theadministrative server starts. These audit messages are written to thefile named tracefile in the logs subdirectory of the WebSphere homeinstallation directory.
- -serverHost. Specifies the name of the machine where theadministrative server or application server is running.
- -setTrace. Specifies a trace specification.
- -setRingBufferSize. Specifies the size of the ring buffer inkilobytes. The default is 8 KB.
- -dumpRingBuffer. Specifies a file name where the ring buffer is tobe written. By default, the ring buffer is written to the currentworking directory. If the administrative server is started as a WindowsNT service, the ring buffer is written to the system default directoryC:\WINNT\system32.
- -dumpState. Specifies a file name where state information for theserver is to be written.
- -stopServer. Stops the server being traced.
- -stopNode. Stops the node.
- -dumpThreads. Specifies a file name where thread history and errorinformation is to be written.

In the following example, the DrAdmin extension is used to trace theadministrative server running on port 1078. By default, the trace ringbuffer is written to the system default directory. The second examplespecifies a file name where the trace is to be written. (For Windows NTpathnames, you can use forward slashes (/). If backslashes are used,they must be prefaced with the backslash (\) character so that the backslashesare treated as ordinary characters.) The third example sets a tracespecification that enables tracing for all container classes.

```
wscp> DrAdmin remote 1078Server trace ring buffer dumped into file JmonDump52701921622wscp> DrAdmin
remote 1078 -dumpRingBuffer e:\\wscp\\dradmin.dumpServer trace ring buffer dumped into file
e:\wscp\dradmin.dumpwscp> DrAdmin remote 1078 -setTrace com.ibm.ejs.container.*=all=enabledServer
trace set to com.ibm.ejs.container.*=all=enabledServer trace ring buffer dumped into file
e:\wscp\dradmin.dump
```

6.6.0.2.2.3.9: Manipulating the JNDI context of objects

The **wscp** Context operations allow you to manipulate the JavaNaming and Directory Interface (JNDI) context of objects that use JNDI namebindings, such as J2CConnectionFactory, DataSource, URL and MailSessionobjects.

The following example creates an initial JNDI context:

```
wscp> Context init
```

The following command binds a MailSession object to the JNDI naming contextejsadmin:

```
wscp> Context bind /MailSession:UserMail/  ejsadmin
```

The following command unbinds a DataSource object from itsJNDI namingcontext:

```
wscp> Context unbind /JDBCDriver:DB2drv/DataSource:appData/
```

For more information on JNDI, see JNDI (Java Naming andDirectory Interface) overview.

6.6.0.2.2.3.10: Monitoring performance

Use the **wscp** PmiService operations to monitor application serverperformance. The PmiService operations can be used to enable or disabletracing, check the values of performance counters, and monitor otherperformance measures. The full range of WebSphere performancemonitoring functions is available through Resource Analyzer.

The following command example turns on tracing for the specifiedapplication server:

```
wscp> PmiService enableData /Node:Appserv1/ApplicationServer:sampleServ/ -dd {JVMRuntimeModule}
```

The following command example turns off tracing for the specifiedapplication server:

```
wscp> PmiService disableData /Node:Appserv1/ApplicationServer:sampleServ/ -dd {JVMRuntimeModule}
```

The following command example displays configuration information for thespecified server:

```
wscp> PmiService getConfigs /Node:Appserv1/ApplicationServer:sampleServ/ [output too long to
display]
```

The following command example lists the modules for which performancestatistics are collected:

```
wscp > PmiService listMembers /Node:Appserv1/ApplicationServer:sampleServ/jvmRuntimeModule
threadPoolModule transactionModule
```

The following command example displays performance information for thespecified server:

```
wscp> PmiService get /Node:Appserv1/ApplicationServer:sampleServ/ [output too long to display]
```

The following command example displays performance information for thespecified module:

```
wscp> PmiService gets /Node:Appserv1/ApplicationServer:sampleServ/ -dd jvmRuntimeModule{Description
jvmRuntimeModule.desc} {Descriptor {NamejvmRuntimeModule} {Type 13}
{FullNameroot/wssol2/ApplicationServer:BrokerAppSrv/jvmRuntimeModule} {NodeNamewssol2} {ServerName
ApplicationServer:BrokerAppSrv} {ModuleNamejvmRuntimeModule} {MaxPathLength 3} {DataDescriptor
{NamejvmRuntimeModule} {Type 13} {ModuleName jvmRuntimeModule} {DataId -1}{Path jvmRuntimeModule}}
{Path wssol2 ApplicationServer:BrokerAppSrvjvmRuntimeModule}}
```

The following command example displays the performance level descriptorsfor the specified application server:

```
wscp> PmiService getLevel /Node:Appserv1/ApplicationServer:sampleServ/{pmi/beanModule 0}
{pmi/threadPoolModule 0} {pmi/connectionPoolModule0} {pmi/jvmRuntimeModule 0} {pmi/transactionModule
0}{pmi/webAppModule 0} {pmi/servletSessionsModule 0} {pmi/jvmpiModule 0}
```

For more information on performance monitoring, see TheWebSphere Resource Analyzer.

6.6.0.2.2.3.11: Setting global security defaults

The **wscp** SecurityConfig operations can be used to do thefollowing:

- Enable and disable WebSphere security on a global basis.
- Specify the default authentication mechanism.
- Configure secure sockets layer (SSL) communication.

They cannot be used to configure security for individual applications orcomponents.

Modifying security properties

Before enabling security, you must make the following modifications to theproduct_installation_root/properties/sas.client.propsfile:

```
com.ibm.CORBA.loginSource=propertiescom.ibm.CORBA.loginUserId=userIDcom.ibm.CORBA.loginPassword=password
```

where *userID* is a valid userID and *password* is thecorresponding password.

The sas.client.props file specifies login information forboth the administrative console and **wscp**. However, you canset up **wscp** to use a different login mechanism than theadministrative console --for instance, you can set up **wscp** fora programmatic login and the console for an interactive login. Do thefollowing:

1. Copy the sas.client.props file to another directory.
2. Change the security properties for **wscp** that were describedearlier in this article.
3. In the file setupCmdLine.bat (Windows) or setupCmdLine.sh(Unix), set the value of the WSCPCLIENTSAS variable to the location of thecopied file. For example:

   ```
   WSCPCLIENTSAS=new_directory/sas.client.props
   ```

   where *new_directory* is the directory where the modified copyof the sas.client.props file resides.

See 6.6.0.2.2.2.3: Authenticating to the administrative server for more information about enabling security.

Security configuration examples

The following example command checks whether security is enabled:

```
wscp> SecurityConfig isSecurityEnabled
```

Return values are:

- 1 (true)--Security is enabled.
- 0 (false)--Security is disabled.

The following example command enables security for all applications:

```
wscp> SecurityConfig enableSecurity
```

The following example command disables security for all applications:

```
wscp> SecurityConfig disableSecurity
```

The following example command returns the current authentication mechanismfor security:

```
wscp> SecurityConfig getAuthenticationMechanism
```

Possible return values are:

- LOCALOS -- The underlying operating system's authenticationmechanism. The local operating system supports basic authenticationsuch as checking a user ID and password.
- LTPA -- Lightweight Third Party Authentication (LTPA). LTPAauthenticates users with a Lightweight Directory Access Protocol (LDAP)directory service and supports certificate-based authentication.

Note:

LTPA cannot be directly configured from the **wscp** command linebecause the configuration settings are too complex. However, you canuse the **wscp** XMLConfig operation to import LTPA configurations thathave been stored in XML files. See 6.6.0.2.2.4.6: Importing and exporting a configuration by using XMLConfig for instructions on how to use this command.

The following example command returns the user ID that can be used in localoperating system (LOCALOS) authentication:

```
wscp> SecurityConfig  getUserid{tym}
```

The following example command sets the authentication method to that of thelocal operating system and authenticates to the user tym:

```
wscp> SecurityConfig  setAuthenticationMechanism  LOCALOS -userid {{tym} {tympwd}}
```

The system uses the operating system's existing securityrepository. The administrative server must be restarted for the changeto take effect.

The following example command displays information about how SSL isconfigured in WebSphere Application Server:

```
wscp> SecurityConfig getSSLConfig{{TrustFileName
${WAS_HOME}/etc/ServerTrustFile.jks}{TrustFilePassword WebAS}
{KeyFileName${WAS_HOME}/etc/ServerKeyFile.jks} {KeyFilePassword WebAS}{KeyFileFormat 0}
{TrustFileFormat 0} {ClientAuthentication false}{UseGlobalDefaults true} {SecurityLevel 0}
{CryptoHardwareEnabledfalse} {CryptoTokenType {}} {CryptoLibraryFile {}} {CryptoPassword
{}}{SSLProperties {}}}
```

The following example command sets various SSL configurationparameters:

```
wscp> SecurityConfig setSSLConfig {{ClientAuthentication true}
{KeyFileName${WAS_HOME}/etc/NewKey.jks} {KeyFilePassword serverAS}}
```

6.6.0.2.2.3.12: Managing security roles

The **wscp** SecurityRoleAssignment operations allow you to managesecurity roles for J2EE applications.

Role-based security enables declarative, customized authentication forapplications. When a J2EE application is assembled, permission toexecute methods is granted to one or more roles, which represent abstractgroups of users. When the application is deployed, actual users orgroups of users are assigned to these roles. When the application isrun, WebSphere Application Server authorizes client requests based on theuser's identification information and what roles the user is assignedto. For a more detailed description of how role-based authentication isimplemented, see article 5.1.3.

The **wscp** SecurityRoleAssignment operations can be used toperform the following tasks:

- List the roles that are defined for an enterprise application.
- List the users and groups that are assigned to each role.
- Add users and groups to a role.
- Delete users and groups from a role.
- Specify the identity under which enterprise bean methods areexecuted.

In addition to individual users and groups, the special groups all usersand all authenticated users can be assigned to security roles.

Note that the SecurityRoleAssignment operations only work with existingsecurity roles; they cannot be used to define new roles. However,you can define and assign security roles when you install an enterpriseapplication with the **wscp** EnterpriseApp install command or installa module with the **wscp** Module install command. See 6.6.0.2.2.4.8: Creating an enterprise application for more information.

The **wscp** SecurityRoleAssignment examples in this section makeuse of the user-to-role mapping in the following table.

Banking enterprise application		**Roles**			
		Teller	**Clerk**	**Supervisor**	**WebTeller**
Users and groups	TellerGroup	Yes			Yes
	Bob	Yes	Yes		Yes
	Mary			Yes	
	ClerkGroup		Yes		
	Supervisor			Yes	
	SupervisorGroup			Yes	

The following example command lists the roles defined for the Bankingenterprise application:

```
wscp> SecurityRoleAssignment listRoles /EnterpriseApp:Banking/Teller Clerk Supervisor WebTeller
```

The following example command lists the roles defined for the Bankingapplication and the users assigned to each role:

```
wscp> SecurityRoleAssignment getUserRoleMapping /EnterpriseApp:Banking/ {Teller {Bob}} {Clerk {}}
{Supervisor {Supervisor}} {WebTeller {Bob}}
```

The following example command lists the users assigned to the Teller rolefor the Banking application:

```
wscp> SecurityRoleAssignment getUserRoleMapping /EnterpriseApp:Banking/ -roles {Teller}{Teller
{Bob}}
```

The following example command lists the roles to which the user Bob isassigned:

```
wscp> SecurityRoleAssignment getUserRoleMapping /EnterpriseApp:Banking/  -users {Bob} {Teller {Bob}}
{WebTeller {Bob}}
```

The following example command lists the roles defined for the Bankingapplication and the groups assigned to each role:

```
wscp> SecurityRoleAssignment getGroupRoleMapping /EnterpriseApp:Banking/ {Teller {TellerGroup}}
{Clerk {ClerkGroup}} {Supervisor {}} {WebTeller {TellerGroup}}
```

The following example command lists the groups assigned to the WebTellerrole for the Banking application:

```
wscp> SecurityRoleAssignment getGroupRoleMapping /EnterpriseApp:Banking/  -roles
{WebTeller}{WebTeller {TellerGroup}}
```

The following example command lists the special role mappings for theBanking application (that is, whether the role has been assigned to all usersor all authenticated users):

```
wscp> SecurityRoleAssignment getSpecialRoleMapping /EnterpriseApp:Banking/
```

Return values are:

- `Everyone`--All users
- `AllAuthenticatedUsers`--All authenticated users

The following example command adds the user Mary to the Teller andWebTeller roles:

```
wscp> SecurityRoleAssignment addUserRoleMapping /EnterpriseApp:Banking/ -userroles {{Teller Mary}
{WebTeller Mary}}
```

Use the -userroles option to specify which users are added to whichroles. Enter either a role-user pair (such as {Teller Mary}) or a listof role-user pairs (such as {{Teller Mary} {WebTeller Mary}}).

The following example command adds the group ClerkGroup to the WebTellerrole:

```
wscp> SecurityRoleAssignment addGroupRoleMapping /EnterpriseApp:Banking/ -grouproles {WebTeller
ClerkGroup}
```

Use the -grouproles option to specify which groups are added to whichsecurity roles. Enter either a role-group pair (such as {WebTellerClerkGroup}) or a list of role-group pairs (such as {{WebTeller ClerkGroup}{WebTeller SupervisorGroup}}).

The following example command adds the special group AllAuthenticatedUsersto the Clerk role:

```
wscp> SecurityRoleAssignment addSpecialRoleMapping /EnterpriseApp:Banking/-specialroles {{Clerk
AllAuthenticatedUsers} {Teller Everyone}}
```

Use the -specialroles option to specify which special groups are assignedto which security roles. Enter either a role-special group pair(such as {Clerk AllAuthenticatedUsers}) or a list of role-special group pairs(such as {{Clerk AllAuthenticatedUsers} {Teller Everyone}}).

The following example command deletes the user Bob from the Teller andWebTeller roles:

```
wscp> SecurityRoleAssignment deleteUserRoleMapping /EnterpriseApp:Banking/-userroles {{Teller Bob}
{WebTeller Bob}}
```

Use the -userroles option to specify which users are deleted from whichroles. Enter either a role-user pair (such as {Teller Bob}) or a listof role-user pairs (such as {{Teller Bob} {WebTeller Bob}}).

The following example command deletes the group ClerkGroup from theWebTeller role:

```
wscp> SecurityRoleAssignment deleteGroupRoleMapping /EnterpriseApp:Banking/-grouproles {WebTeller
ClerkGroup}
```

Use the -grouproles option to specify which groups are deleted from whichsecurity roles. Enter either a role-group pair (such as {WebTellerClerkGroup}) or a list of role-group pairs (such as {{WebTeller ClerkGroup}{WebTeller SupervisorGroup}}).

The following example command deletes the special groups AllUsers andAllAuthenticatedUsers from the Clerk role:

```
wscp> SecurityRoleAssignment deleteSpecialRoleMapping /EnterpriseApp:Banking/ -specialroles {{Clerk
AllAuthenticatedUsers} {Clerk Everyone}}
```

Use the -specialroles option to specify which special groups are deletedfrom which security roles. Enter either a role-special group pair(such as {Clerk AllAuthenticatedUsers}) or a list of role-special group pairs(such as {{Clerk AllAuthenticatedUsers} {Clerk Everyone}}).

The following example command lists the execution identities and roles thatenterprise bean methods run under in the Banking application:

```
wscp> SecurityRoleAssignment getRunAsToUser /EnterpriseApp:Banking/{Supervisor {Supervisor Bob}}
```

The following example command assigns the execution identity Mary forenterprise bean methods that run under the Supervisor role. A passwordmust also be specified; in this case, the password is marypwd.

```
wscp> SecurityRoleAssignment setRunAsToUser /EnterpriseApp:Banking/ -runasroles {Supervisor Mary
marypwd}
```

6.6.0.2.2.3.13: Connecting to remote servers

The **wscp** Remote operations enable **wscp** to administerremote servers as follows:

- Connect to remote servers by using the Remote attach command.
- Disconnect from remote servers by usng the Remote detach command.
- Start a server listener process by using the Remote listen command.

The following example command shows how to create a connection to a remoteserver:

```
wscp> Remote attach greenland.ibm.com:1000:123898
```

where greenland.ibm.com is the server name, 1000 is the portnumber, and 123898 is the key number.

The following example command closes all remote connections:

```
wscp> Remote detach
```

The following example commands opens up a server listener process for thespecified server, port number, and key:

```
wscp> Remote listen greenland.ibm.com:1000:123898Server listening at greenland.ibm.com:1000:123898
```

6.6.0.2.2.3.14: Where to find more information about Tcl

The following are some useful Tcl commands for writing **wscp**scripts. For additional information on Tcl, refer to *Tcl and theTK Toolkit* by John K. Ousterhout (Addison Wesley), or to the Tcldeveloper Web site at http://www.scriptics.com.

- **set**. Creates, reads, and modifies variables.

  ```
  set serv "/Node:dev-pc/ApplicationServer:myAppServer/"
  ```
- **eval**. Accepts any number of arguments, concatenates themwith separator spaces, then executes the result as a Tcl script. Oneuse of **eval** is for generating commands, saving them in variables,and then later evaluating the variables as Tcl scripts.

  ```
  set cmd {ApplicationServer stop /Node:MyNode/ApplicationServer:MyServer/}. . .eval $cmd
  ```
- **exec**. Creates one or more new processes and waits untilthey are complete before returning. Looks for an executable file in theworking directory or uses the PATH environment variable.

  ```
  exec date
  ```
- **global**. Makes global variables available inside aprocedure.

  ```
  global errorCode
  ```
- **lappend**. Appends new elements to a list stored in avariable.

  ```
  set vars {value1 value2 value3}value1 value2 value3lappend vars value4value1 value2 value3 value4
  ```
- **lindex**. Extracts an element from a list.

  ```
  lindex $vars  2value3
  ```
- **lreplace**. Deletes elements from a list and optionallyadds new elements. The first argument is a list, and the second andthird arguments are the indices of the first and last elements to bedeleted.

  ```
  lreplace $vars 1 2value1 value4
  ```
- **lsearch**. Searches a list for an element with aparticular pattern and returns the index of the first matching element that isfound.

  ```
  lsearch $vars value43
  ```
- **proc**. Creates a named procedure and assigns a list ofarguments to be used with that procedure.

  ```
  proc checkStatus {expectedStatus}proc getAttrs {name array args}
  ```

6.6.0.2.2.4: Example wscp commands, procedures, and scripts

This section contains example **wscp** commands, Tcl procedures, andTcl scripts. You can use these examples as provided, or customize themto develop procedures and scripts suited to your application. Many ofthe procedures and scripts are available as stand-alone files in 6.6.0.2.2.5: Sample Tcl procedures and scripts.

Note that not all **wscp** commands are represented in theseexamples. The examples are intended to demonstrate only the morefrequently used commands (such as commands for creating and installingapplication servers or displaying the attributes of an object). Theexamples include several custom procedures and other files that offer guidancein developing your own scripts and procedures.

Consult the **wscp** command-line help for a complete list ofoperations for each object type.

The following list contains a description of the administrative task thateach example performs; states whether the example is a stand-alone**wscp** command, a Tcl procedure, or a Tcl script; and includesthe name of the Tcl procedure or script.

Note:

> Many of the example scripts and procedures require that the fileinit.tcl be loaded prior to running. This file initializesseveral variables used elsewhere and contains useful Tcl procedures foraccessing attributes, obtaining error status information, and othertasks. See 6.6.0.2.2.4.1: Initialization and general-purpose procedures for a description of the contents of this file.

- 6.6.0.2.2.4.1: Initialization and general-purpose procedures
- 6.6.0.2.2.4.2: Configuring objects
  - Creating objects. Stand-alone **wscp** commands. See Creating an object.
  - Viewing the default values of attributes. Stand-alone**wscp** commands. See Working with the default values of attributes.
  - Modifying objects. Stand-alone **wscp** commands. SeeModifying an object.
  - Modifying the Environment attribute. A custom procedure, modEnv,for modifying or replacing values in the Environment attribute. Theprocedure can be customized for modifying the values of other attributes thatalso require a Tcl list of strings. See Modifying an Environment attribute (modEnv procedure). This procedure is available in 6.6.0.2.2.5: Sample Tcl procedures and scripts.
- Starting live repository objects. Stand-alone **wscp**commands for starting and stopping application servers and other liveobjects. See 6.6.0.2.2.4.3: Starting and stopping live repository objects.
- 6.6.0.2.2.4.4: Displaying information about objects
  - Listing instances of an object type, including listing the objectsrecursively. Stand-alone **wscp** commands. See Listing objects.
  - Querying an object (displaying the values of an instance'sattributes). Stand-alone **wscp** commands and a customprocedure named display. See Querying (displaying) attributes. This procedure is available in 6.6.0.2.2.5: Sample Tcl procedures and scripts.
  - Printing the attributes of an object type. Custom procedure,printAttributes, for printing the attributes of all or select objects.See Printing an object's attributes. This procedure is available in 6.6.0.2.2.5: Sample Tcl procedures and scripts.
  - Viewing the containment hierarchy for all object types or for selectedobject types. Custom procedure, printContainment, for displaying thecontainment hierarchy for all or selected objects.

    See Viewing the containment hierarchy. This procedure is available in 6.6.0.2.2.5: Sample Tcl procedures and scripts.

  - Displaying selected attributes of an application server. Customprocedure, showServerStatus, that displays the name and current state of allservers in a domain. Can be customized to display otherattributes. See Displaying select attributes. This procedure is available in 6.6.0.2.2.5: Sample Tcl procedures and scripts.

- Removing objects recursively. Stand-alone **wscp**commands. The commands are useful in testing situations where objectsmust be routinely created and destroyed. See 6.6.0.2.2.4.5: Removing objects and applications.
- Invoking the XML Configuration Management Tool (XMLConfig) from within**wscp**. Stand-alone **wscp** command. See 6.6.0.2.2.4.6: Importing and exporting a configuration by using XMLConfig.
- Creating a JDBCDriver and DataSource object. Stand-alone**wscp** commands. See 6.6.0.2.2.4.7: Creating and installing drivers and data sources.
- Creating an enterprise application. Tcl scripts and stand-alone**wscp** commands that demonstrate installing and starting anenterprise application and installing Enterprise JavaBeans (EJB) and WebArchive (WAR) modules. See 6.6.0.2.2.4.8: Creating an enterprise application. These scripts are available in 6.6.0.2.2.5: Sample Tcl procedures and scripts.
- Configuring server groups and clones. Sample **wscp**commands that exercise most operations associated with server groups andclones. The commands can be used alone or in combination. See 6.6.0.2.2.4.10: Configuring server groups and clones.
- Administering connections to enterprise information system (EIS) backendsystems. Stand-alone **wscp** commands for installing andcreating instances of J2C resource adapters and J2C connectionfactories. See 6.6.0.2.2.4.11: Administering EIS connections.
- Configuring Java Message Service (JMS) clients. Stand-alone**wscp** commands for creating JMS providers, JMS connection factoriesand JMS destinations. See 6.6.0.2.2.4.12: Administering the Java Message Service (JMS).
- Administering JavaMail sessions. Stand-alone **wscp**commands for creating and destroying JavaMail sessions. See 6.6.0.2.2.4.13: Administering JavaMail sessions.
- Administering URLs and URL providers. Stand-alone **wscp**commands for installing and creating instances of URLs and URLproviders. See 6.6.0.2.2.4.14: Administering URL providers and URLs.

6.6.0.2.2.4.1: Initialization and general-purpose procedures

The init.tcl script initializes variables used elsewhere in theexamples. Note that init.tcl was used for a specific test suite,but it contains several procedures that can be generally useful. If youare writing scripts that must accept some common command-line arguments, youcan use or add to the predefined variables as needed. The contents ofinit.tcl are as follows:

- getAttrs and setAttrs. Procedures that get and set an array ofattributes for a specified object.
- getProperty. Procedure that retrieves a specified system property,such as the operating system name and operating system-specific fileseparator.
- which. Procedure that provides the Tcl equivalent of thecorresponding UNIX command. Retrieves the path to the specifiedargument (must be an executable) on java.library.path.
- parseArguments. Sets global variables for the host name, hostinternet address, and node. Also sets the VERBOSE globalvariable. If VERBOSE is set to 1, the **wscp** command beingexecuted is echoed to the screen.
- initConstants. Sets up lists or arrays of lists for **wscp**commands, operations, actions, and status values. Also sets thelocation of home directories for WebSphere Application Server, DB2, the IBMDebugger; the port number of the administrative server; and othermiscellaneous information.
- statusToString. Translates a specified **wscp** status toits corresponding string equivalent, or if called with no arguments,translates the current value of $errorCode.
- checkStatus. Tests whether the expected status matches thespecified status, or as in the previous item, the current value of $errorCodeif no status argument is provided.

The init.tcl script uses the global variable VERBOSE to echo the**wscp** commands as they are being executed. To set thisvariable to true, specify 1 as the value of the -verbose option when runningwscp.bat.

```
C:\> wscp -verbose 1
```

The init.tcl script is available in 6.6.0.2.2.5: Sample Tcl procedures and scripts.

6.6.0.2.2.4.2: Configuring objects

The following examples demonstrate how to create and modify objects usingthe **wscp** interface:

- Creating an object
- Working with the default values of attributes
- Modifying an object
- Modifying an Environment attribute (modEnv procedure)

Creating an object

The **wscp** create operation creates an object instance. Thesyntax is as follows:

```
object_type create object_name -attribute attr_list
```

The arguments are as follows:

- object_type. Specifies the object type of theinstance.
- object_name. Specifies the object instance to becreated.
- -attribute attr_list. Specifies a Tcl list ofattribute-value pairs to set.

The following interactive **wscp** command creates an applicationserver. The object_type argument (ApplicationServer) specifiesthat an application server object is to be operated on. Theoperation argument (create) specifies that an object is to becreated. The object_name argument(/Node:dev-pc/ApplicationServer:myServer/) is the name of theserver object to be created.

```
wscp> ApplicationServer create /Node:dev-pc/ApplicationServer:myServer/
```

The following **wscp** command creates a DataSource object nameds1.

```
wscp> DataSource create /JDBCDriver:Db2Jdbc/DataSource:ds1/
```

You must specify the name of the desired JDBC provider when creating a datasource -- in this case, the driver /JDBCDriver:Db2Jdbc/

Working with the default values of attributes

An object type's attributes can have default values. When youcreate an object instance, the instance inherits any default values (unlessyou explicit set them in the create operation). For example, theSecurityEnabled attribute of an ApplicationServer object has a default valueof False. Some attributes have a default value of null or the emptylist--for example, the value of the Environment attribute ofApplicationServer objects defaults to the empty list.

The defaults operation is used to view the default values forattributes. (All object types support the defaults operation with theexception of the ServerGroup object type.) The syntax is asfollows:

```
<object_type> defaults  [-all] [-attribute <attribute list>]
```

If you do not specify any options, the defaults operation displays thedefault values for attributes of the specified object type. Thefollowing example displays the default values for attributes of the DataSourceobject type. Note that, while other attributes exist for this objecttype, the defaults operation displays only those attributes that have defaultvalues.

```
wscp> DataSource defaults{Description {}} {ConfigProperties {}} {ConnTimeout 180}{DefaultPassword
{}} {DefaultUser {}} {DisableAutoConnectionCleanupFalse} {IdleTimeout 1800} {JNDIName {}}
{MaxPoolSize 10} {MinPoolSize1} {OrphanTimeout 1800} {StatementCacheSize 100}
```

The -all option displays all attributes (those that have default values aswell as those that are not set). The following example displays allattributes of the DataSource object type. Even though unset attributesdo not have an initial setting, they are displayed as having the string valueAttributeNotSet.

```
wscp> DataSource defaults -all{Name AttributeNotSet} {FullName AttributeNotSet} {Description
{}}{ConfigProperties {}} {ConnTimeout 180} {DatabaseName AttributeNotSet}{DefaultPassword {}}
{DefaultUser {}} {DisableAutoConnectionCleanupFalse} {IdleTimeout 1800} {JNDIName {}} {MaxPoolSize
10} {MinPoolSize1} {OrphanTimeout 1800} {StatementCacheSize 100}
```

The -attribute option is used to display the default value for one or morespecified attributes. Its argument must be a Tcl list. If thespecified attribute has a default value, the value is displayed. Thefollowing example displays the default values of the MinPoolSize andMaxPoolSize attributes of DataSource objects:

```
wscp> DataSource defaults -attribute {MaxPoolSize MinPoolSize}{MaxPoolSize 10} {MinPoolSize 1}
```

Modifying an object

The **wscp** modify operation sets the value of one or moreattributes. If a value already exists for an attribute, that value isreplaced. The syntax is as follows:

```
object_type modify  object_name -attribute  attr_list
```

The arguments are as follows:

- object_type. Specifies the object type of theinstance.
- object_name. Specifies the object instance whoseattributes are to be modified.
- -attribute attr_list. Specifies a Tcl list ofattribute-value pairs to set.

The following example shows how to modify the value of the PingIntervalattribute of an application server:

```
wscp> ApplicationServer show /Node:dev-pc/ApplicationServer:myServer/  \-attribute PingInterval
{PingInterval 60}
```

```
wscp> ApplicationServer modify /Node:dev-pc/ApplicationServer:myServer/ \-attribute {{PingInterval
120}}
```

```
wscp> ApplicationServer show /Node:dev-pc/ApplicationServer:myServer/  \-attribute
PingInterval{PingInterval 120}
```

Modifying an Environment attribute (modEnv procedure)

The following Tcl procedure, modEnv, can be used to modify the Environmentattribute of one or more application servers in a domain. The value ofthe Environment attribute is a list of strings of the form*name*=*value*. This procedure modifies a specifiedelement of the list (or adds the element if it is not present) and retains thevalues of the other elements. The procedure takes threearguments: a server name, an environment variable name, and the value towhich the variable is to be set.

The procedure does the following:

1. Sets a Tcl variable named oldEnv to the existing value of the Environmentattribute.
2. Searches the variable names in oldEnv for the variable name supplied as anargument to the command. If the supplied variable name is not inoldEnv, the procedure appends the supplied variable and value to oldEnv in theform name=value, and sets the variable newEnv to the value of oldEnv.If the supplied variable name is in oldEnv, the procedure replaces thevariable's value with the supplied value and sets newEnv to the value ofoldEnv.
3. Modifies the server's Environment attribute so that it is equal tothe value of newEnv.

```
## modEnv - procedure for modifying the Environment attribute of one or# more application servers in
a domain.  The specified environment variable# is modified (or added if it is not present), and the
values of other # variables are retained.## Arguments:## server - the fully qualified name of the
application server whose# Environment attribute is to be modified.## variable - the name of the
environment variable to modify.## value - the new value of the environment variable.## To modify the
Environment attribute of multiple servers, use# the Tcl foreach command, for example# wscp> foreach
server [ApplicationServer list] {modEnv $server TEST_VARIABLE 3.5}## The file init.tcl must be
loaded prior to using this procedure.  proc modEnv {server variable value} {    set oldEnv {}
getAttrs $server attr Environment    if {[info exists attr(Environment)]} {set oldEnv
$attr(Environment)}    # append to environment if not found; replace if it is found    set i
[lsearch -regexp $oldEnv ^$variable=]    if {$i == -1} { set newEnv [lappend oldEnv
"$variable=$value"]    } else { set newEnv [lreplace $oldEnv $i $i "$variable=$value"]    }    set
attr(Environment) $newEnv    setAttrs $server attr}modEnv
```

The following example shows the use of the modEnv procedure to modify thevalue of the PATH variable in two application servers in a domain (or to addthe PATH variable if it does not exist). The existing value of theEnvironment attribute of each server is as follows:

```
wscp> ApplicationServer show $serv1 -attribute {Environment}{Environment PATH=/myPath}wscp>
ApplicationServer show $serv2 -attribute {Environment}{Environment {{OTHERVARIABLE=word1 word2}}}
```

The following calls to the modEnv procedure modify each Environmentattribute. The resulting change is shown for the two exampleservers:

```
wscp> modEnv $serv1 PATH /revisedPathwscp> ApplicationServer show $serv1 -attribute
{Environment}{Environment PATH=/revisedPath}wscp> modEnv $serv2 PATH /revisedPathwscp>
ApplicationServer show $serv2 -attribute {Environment}{Environment {{OTHERVARIABLE=word1 word2}
PATH=/revisedPath}}
```

The modEnv procedure can be used to change the Environment attribute ofmultiple servers of the same type. In the following example, the Tcl**foreach** command is used to call the modEnv procedure on eachApplicationServer instance in a domain. The value TEST_VARIABLE=1 isadded (or appended) to the attribute as needed.

```
wscp> foreach server [ApplicationServer list] {modEnv $server TEST_VARIABLE 1}
```

6.6.0.2.2.4.3: Starting and stopping live repository objects

The **wscp** interface can be used to start and stop any liverepository object. Live repository objects include the following:

- Nodes
- Application servers
- Generic servers
- Enterprise applications
- Modules
- Server groups

The syntax for the start operation and stop operation is as follows:

```
object_type start  object_name object_type stop  object_name
```

The following example starts the application server named myServer:

```
wscp> ApplicationServer start /Node:dev-pc/ApplicationServer:myServer/
```

6.6.0.2.2.4.4: Displaying information about objects

The following examples include **wscp** operations and custom Tclprocedures for displaying information about objects:

- Listing objects
- Querying (displaying) attributes
- Printing an object's attributes
- Viewing the containment hierarchy
- Displaying select attributes

Listing objects

The **wscp** list operation lists all instances of an object typeor, optionally, only those instances that meet the specified criteria.The syntax is as follows:

```
object_type list  [-constraint attr_list]  [-recursive]
```

The arguments are as follows:

- -constraint attr_list. Specifies a Tcl list of attributesto use as a constraint. Only objects with the specified attribute-valuepairs are listed.
- -recursive. Lists all instances of any object type that belongs tothe containment hierarchy of the specified type.

The following example command lists all instances of the ApplicationServerobject type in the domain:

```
wscp> ApplicationServer list{/Node:dev-pc/ApplicationServer:Default Server/}
{/Node:dev-pc/ApplicationServer:AcctServer1/} {/Node:dev-pc/ApplicationServer:AppServerGroup1/}
{/Node:dev-pc/ApplicationServer:AppServerGroup2/} {/Node:dev-pc/ApplicationServer:CustServer1/}
```

The -recursive option lists all instances of any object type that belongsto the containment hierarchy of the specified type. TFor instance, fora JDBCDriver object type, the -recursive option lists all instances ofDataSource.

```
wscp> JDBCDriver list -recursive/JDBCDriver:OracleDrv/ /JDBCDriver:OracleDrv/DataSource:OracleDB/
/JDBCDriver:OracleDrv/DataSource:AppDb1/ /JDBCDriver:DB2Drv/ /JDBCDriver:DB2Drv/DataSource:WAS/
/JDBCDriver:DB2Drv/DataSource:AppDb2/ /JDBCDriver:DB2Drv/DataSource:SampleDB/
```

The following example lists only those application servers whosePingInterval attribute has 60 as its value:

```
wscp> ApplicationServer list -constraint {{PingInterval
60}}{/Node:dev-pc/ApplicationServer:AcctServer1/} {/Node:dev-pc/ApplicationServer:CustServer1/}
{/Node:dev-pc/ApplicationServer:AppServer2/}
```

Querying (displaying) attributes

The **wscp** show operation displays the values of all attributes ora specified subset of attributes for an object instance. The syntax isas follows:

```
object_type show object_name [-all] [-attribute attr_list]
```

The arguments are as follows:

- object_type. Specifies the object type of theinstance.
- object_name. Specifies the object instance whoseattributes are to be displayed.
- -all. Displays the values of all attributes (those that are set aswell as those that are not set). See Working with the default values of attributes for details on attributes.
- -attribute attr_list. Specifies a Tcl list of attributesto display.

The following show operation displays all attributes of a node namedws2. (The variable $node is set to the fully qualified name of thenode.)

```
wscp> Node show $node -all{Name ws2} {FullName /Node:ws2/} {CurrentState Running}{DesiredState
Running} {StartTime 988812380570} {HostNamewssol2.transarc.ibm.com} {HostSystemType sparc}
{ProcessId 10266}{InstallRoot /opt/WebSphere/AppServer} {PathMap
{WSCP0008I:EditorNotDefinedForThisProperty}}
```

The following show operation displays the values of specific attributes(the Name and CurrentState attributes) of an application server:

```
wscp> ApplicationServer show /Node:dev-pc/ApplicationServer:myServer/ \-attribute {Name
CurrentState}
{Name myServer} {CurrentState running}
```

The **wscp** show operation is implemented somewhat differently forapplication servers because several attributes have very long values.By default, the ApplicationServer show operation displays all applicationserver attributes *except* the JVMConfig, ORBConfig, andWebContainerConfig attributes, which have very long multipart values.However, **wscp** does provide two ways to display the values of theseattributes:

- By using the -attribute flag to explicitly display their values.For example, the following show operation displays the value of the JVMConfigattribute:

  ```
  wscp> ApplicationServer show /Node:node1/ApplicationServer:myServ/ -attribute {JVMConfig}{JVMConfig
  {{JvmPropertiesArray {}} {AdditionalCommandLineArgs {}}{BootClasspathAppend {}}
  {BootClasspathPrepend {}}{BootClasspathReplace {}} {Classpaths {}} {DebugMode false}{DebugString {}}
  ```

```
    {DisableJIT false} {HProfArgs {}} {InitialHeapSize0} {MaxHeapSize 0} {RunHProf false}
    {SystemProperties {}}{VerboseGC false} {VerboseJNI false} {GeneratedCommandLineArgs {}}}}
```

- By using the ApplicationServer showall operation to display the values ofall application server attributes, regardless of their length.

The show and showAttrs operations perform different functions. Allobject types have a show operation for displaying attributes. Servergroup objects have, in addition to the show operation, the showAttrsoperation. For these objects, the show operation displays theattributes associated with the server group--for example, the IfStartedand StartTime attributes. The showAttrs operation displays the defaultattributes of clones associated with the server group. (Theseattributes match the properties for the application server resource.)

The following examples illustrate output for a show and showAttrsoperation. The object ServGrp1 is a server group created for anapplication server:

```
# show commandwscp> ServerGroup show /ServerGroup:ServGrp1/{Name ServGrp1} {FullName
/ServerGroup:ServGrp1/} {StartTime 0}{IfStarted False} {EJBServerAttributes {{Name
AttributeNotSet}{FullName AttributeNotSet} {CurrentState Stopped} {DesiredStateStopped} {StartTime
0} {ProcessId 0} {Environment {}} {Executablejava} {ExecutableActive java} {CommandLineArgs
{}}{CommandLineArgsActive {}} {EnvironmentActive {}} {UserId {}}{UserIdActive {}} {GroupId {}}
{GroupIdActive {}} {WorkingDirectory{}} {WorkingDirectoryActive {}} {Umask 18} {UmaskActive 18}
{Stdin {}}{StdinActive {}} {Stdout /tmp/Broker.stdout} {StdoutActive/tmp/Broker.stdout} {Stderr
/tmp/Broker.stderr} {StderrActive/tmp/Broker.stderr} {MaxStartupAttempts 2} {ProcessPriority
20}{ProcessPriorityActive 20} {PingInterval 60} {PingTimeout 200}{PingInitialTimeout 300} {TraceSpec
{}} {SystemProperties {}}{ThreadPoolConfig {MinimumSize 10} {MaximumSize 50} {InactivityTimeout10}
{IsGrowable false}} {Transports {{{Protocol http} {Host *} {Port9080} {SSLConfig {}} {MaxKeepAlive
25} {MaxReqKeepAlive 100}{KeepAliveTimeout 5} {ConnectionTimeout 5} {BacklogConnections
50}{HttpProperties {}} {SSLEnabled false}}}} {DynamicCacheConfig{CacheSize 1000} {Enabled false}
{CacheGroups {}}}}} {ModuleVisibility3} {ModuleVisibilityActive 3} {UseDomainQualifiedUserNames
False}{UseDomainQualifiedUserNamesActive False} {DefaultDataSource {}}#showAttrs commandwscp>
ServerGroup showAttrs /ServerGroup:ServGrp1/{CurrentState Stopped} {DesiredState Stopped} {StartTime
0} {ProcessId0} {Environment {}} {Executable java} {ExecutableActive java}{CommandLineArgs {}}
{CommandLineArgsActive {}} {EnvironmentActive {}}{UserId {}} {UserIdActive {}} {GroupId {}}
{GroupIdActive {}}{WorkingDirectory {}} {WorkingDirectoryActive {}} {Umask 18}{UmaskActive 18}
{Stdin {}} {StdinActive {}} {Stdout/tmp/Broker.stdout} {StdoutActive /tmp/Broker.stdout}
{Stderr/tmp/Broker.stderr} {StderrActive /tmp/Broker.stderr}{MaxStartupAttempts 2} {ProcessPriority
20} {ProcessPriorityActive 20}{PingInterval 60} {PingTimeout 200} {PingInitialTimeout 300}{TraceSpec
{}} {SystemProperties {}} {ThreadPoolConfig {MinimumSize10} {MaximumSize 50} {InactivityTimeout 10}
{IsGrowable false}}{Transports {{{Protocol http} {Host *} {Port 9080} {SSLConfig {}}{MaxKeepAlive
25} {MaxReqKeepAlive 100} {KeepAliveTimeout 5}{ConnectionTimeout 5} {BacklogConnections 50}
{HttpProperties {}}{SSLEnabled false}}}} {DynamicCacheConfig {CacheSize 1000} {Enabledfalse}
{CacheGroups {}}}}} {ModuleVisibility 3}{ModuleVisibilityActive 3} {UseDomainQualifiedUserNames
False}{UseDomainQualifiedUserNamesActive False} {DefaultDataSource {}}
```

The following custom procedure, display, displays the output of the showoperation in a readable format--one attribute per line. Theprocedure's arguments are an object type and the name of an objectinstance.

```
## display - a procedure for displaying attributes in a readable format.## Arguments:## type - the
object type whose attributes are to be displayed.# # name - the fully-qualified name of the object
instance whose attributes # are to be displayed.# proc display {type name} {        set attrs [$type
show $name]    foreach attr $attrs { puts $attr    }}display
```

The following example demonstrates output from the displayprocedure. The procedure is used display the attributes of a JDBCDriverobject named DB2_Drv:

```
wscp> display JDBCDriver /JDBCDriver:DB2_Drv/Name DB2_DrvFullName /JDBCDriver:DB2_Drv/Description
nullImplClass COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource
```

This procedure is available in 6.6.0.2.2.5: Sample Tcl procedures and scripts.

Printing an object's attributes

The custom Tcl procedure printAttributes uses the attributes operation toprint all or a subset of attributes for one or more object types.

- If no arguments are specified, the procedure prints the names of allattributes for all object types.
- If one or more options of the attributes operation are specified (forexample, if the -required and -readOnly options are specified), the commandprints only those groups of attributes.
- If one or more object types are specified, the command prints theattributes for the specified object types.

```
## printAttributes - prints the attributes for any objects specified or # for all objects if no
objects are specified.## Arguments:# # options - a list of options to control the types of
attributes printed. # Every option in the list of option names must be a valid option to the # wscp
attributes command (for example, -required or -cloneOnly)# AND the option name must begin with a
"-".## objects - a list of objects whose attributes are to be printed.# # The file init.tcl must be
loaded prior to using this procedure.# proc printAttributes {{options all} {objects all} args} {
global OBJECTS    if {[string first "-" $options] != 0} {       set objects $options    set options
""    }    if {[string compare $objects "all"] == 0} {set objects $OBJECTS}    if {$args != ""} {
foreach arg $args {         lappend objects $arg        }    }     foreach o $objects {          set
cmd [concat $o attributes $options] puts "# $cmd"   set result [eval $cmd]  puts $result
}}printAttributes
```

The following example commands demonstrate the use of the printAttributesprocedure. The first example prints only the attributes associated with clones of

ApplicationServer objects.

```
wscp> printAttributes -cloneOnly ApplicationServer# ApplicationServer attributes -cloneOnlyName
FullName
```

The following command prints all required attributes for all objecttypes:

```
wscp> printAttributes -required# JDBCDriver attributes -requiredName ImplClass# VirtualHost
attributes -requiredName AliasList # ApplicationServer attributes -requiredName# DataSource
attributes -requiredName# EnterpriseApp attributes -requiredName OrigEarFile OrigNodeName#
GenericServer attributes -requiredName Executable# J2CConnectionFactory attributes -requiredName#
J2CResourceAdapter attributes -requiredName ArchiveFile# JMSConnectionFactory attributes
-requiredName ConnectionType ExternalJNDIName# JMSDestination attributes -requiredName
ExternalJNDIName DestinationType# JMSProvider attributes -requiredName ExternalInitialContextFactory
ExternalProviderUR# MailSession attributes -requiredName MailTransportHost# Module attributes
-requiredName ModuleType RelativeURI ModuleTypeActive# Node attributes -requiredName# ServerGroup
attributes -requiredName# URL attributes -requiredName Spec# URLProvider attributes -requiredName
Protocol StreamHandlerClassName
```

The following command prints, for ApplicationServer and EnterpriseAppobjects, only those attributes that are required and that are specified atstartup:

```
wscp> printAttributes {-required -startUp} {ApplicationServer EnterpriseApp}# ApplicationServer
attributes -required -startUpName Environment CommandLineArgs UserId GroupId WorkingDirectory
UmaskStdin Stdout Stderr MaxStartupAttempts ProcessPriority TraceSpecSystemProperties TraceOutput
LogFileSpec DebugEnabled SourcePathOLTEnabled OLTServerHost OLTServerPort IsAClone
SecurityEnabledCacheConfig WebContainerConfig ModuleVisibilityUseDomainQualifiedUserNames
DefaultDataSource# EnterpriseApp attributes -required -startUpName OrigEarFile OrigNodeName Bindings
```

This procedure is available in 6.6.0.2.2.5: Sample Tcl procedures and scripts.

Viewing the containment hierarchy

The custom procedure printContainment can be used to display thecontainment hierarchy of a single object type or the entire object typehierarchy. The procedure takes zero or more arguments. Argumentsare object types. If no arguments are supplied, the procedure printsthe containment hierarchy for all object types. If one or morearguments are supplied, the procedure prints the containment hierarchy for thespecified object types only. Prior to using the procedure, you mustalso load the init.tcl file.

```
## printContainment - a procedure that prints the containment hierarchy for # one or more object
types. # # The script init.tcl must be loaded prior to using this procedure.## Arguments:## objects
- one or more object types whose containment hierarchies are to be printed.## The init.tcl file must
be loaded prior to using this procedure.# proc printContainment {{objects all} args} {    global
OBJECTS    if {$objects == "all"} {set objects $OBJECTS}        if {"$args" != "" } {        foreach
elem $args {        lappend objects $elem       }       }     foreach o $objects {      set cmd
[concat $o containment] puts "# $cmd"    set result [eval $cmd]  puts $result    }}printContainment
```

The following example command demonstrates the use of the printContainmentprocedure to display the containment hierarchy of ApplicationServer andJDBCDriver objects:

```
wscp> printContainment {ApplicationServer JDBCDriver}# ApplicationServer containmentNode
ApplicationServer# JDBCDriver containmentJDBCDriver
```

To print the containment heirarchy of all objects, issue theprintContainment command without any options.

The printContainment procedure is available in 6.6.0.2.2.5: Sample Tcl procedures and scripts.

Displaying select attributes

The following custom procedure, showServerStatus, displays the status (thevalue of the CurrentState attribute) of all application servers in adomain. The procedure can be customized to display additionalattributes or attributes of other objects.

```
## showServerStatus - a procedure for displaying the value of the# Name and CurrentState attribute
of all application servers # in a domain.# proc showServerStatus {} {                puts "\nStatus
of servers in the domain:\n"             foreach ejbserver [ApplicationServer list] {    puts
[ApplicationServer show $ejbserver -attribute {Name CurrentState}] }}showServerStatus
```

The following example demonstrates output of the showServerStatusprocedure:

```
wscp> showServerStatusStatus of servers in the domain:{Name {Default Server}} {CurrentState
{Initialization Failed}}{Name {Appl Server1}} {CurrentState Running}{Name {ServerGroup1}}
{CurrentState Stopped}{Name {ServerGroup2}} {CurrentState Stopped}{Name {Appl Server2}}
{CurrentState Running}{Name {My Server}} {CurrentState Stopped}{Name {Test Server}} {CurrentState
Stopped}
```

The display procedure is available in 6.6.0.2.2.5: Sample Tcl procedures and scripts.

6.6.0.2.2.4.5: Removing objects and applications

The **wscp** remove operation removes objects (and all references tothose objects) from the domain. *A live repository object must bestopped before being removed.* The syntax of the remove operationis as follows:

```
object_type remove  object_name  [-recursive]
```

If the object to be removed contains other objects, removal fails unlessyou use the -recursive option. The -recursive option removes allinstances of any object type that belongs to the containment hierarchy of theobject instance being removed. For example, if a node instance isremoved recursively, any object instance belonging to the containmenthierarchy of the node object is also removed. The recursion takes placein a downward direction (removal of object instances *below* thespecified object in the hierarchy).

The following example command recursively removes components of anapplication server named AppServ:

```
wscp> ApplicationServer remove /Node:AppNode/ApplicationServer:AppServ/ -recursive
```

6.6.0.2.2.4.6: Importing and exporting a configuration by using XMLConfig

The XML Configuration Management Tool (XMLConfig) can be invoked fromwithin **wscp**. The syntax is as follows:

```
wscp> XMLConfig export file_name [-partial file_name] wscp> XMLConfig import file_name [-substitute list]
```

For an export, specify the name of an XML output file for thefile_name argument. Specify the name of an XML data file forthe -partial option. For an import, specify the name of an XML datafile for the file_name argument. Specify a list ofvariable-value pairs for the -substitute option as follows:

```
{{variable1 value1}{variable2 value2}}
```

If a value string contains spaces, it must also be enclosed in braces({}).

6.6.0.2.2.4.7: Creating and installing drivers and data sources

The following example **wscp** commands create and install a JDBCdriver and data source. A JDBC provider and data source must beconfigured for each brand and version of database from which applicationservers or enterprise applications require connections.

The script init.tcl must be loaded prior to using these commands(init.tcl initializes NODE, DB2_HOME, and other variables usedhere).

```
wscp> JDBCDriver create /JDBCDriver:DB2Driver/ -attribute \{{ImplClass
com.ibm.db2.jdbc.app.DB2ConnectionPoolDataSource}} # Create a DataSource object wscp> DataSource
create /JDBCDriver:DB2Driver/DataSource:testDataSource/ # Install the JDBCDriver object wscp>
JDBCDriver install /JDBCDriver:DB2Driver/ -node $NODE \ -jarFile
${DB2_HOME}${FILE_SEPARATOR}java${FILE_SEPARATOR}db2java.zip
```

The following example commands remove the DataSource object and uninstallthe JDBCDriver object:

```
# Remove the DataSource object just created    wscp> DataSource remove
/JDBCDriver:DB2Driver/DataSource:testDataSource/    # Uninstall the JDBCDriver object    wscp>
JDBCDriver uninstall /JDBCDriver:DB2Driver/ -node $NODE    # Remove the JDBCDriver object just
created    wscp> JDBCDriver remove /JDBCDriver:DB2Driver/
```

6.6.0.2.2.4.8: Creating an enterprise application

This section discusses the **wscp** operations for deployingenterprise applications and modules.

- Administering enterprise applications describes how to install, start, and stop enterpriseapplications by using the **wscp** EnterpriseApp operations.
- Administering modules describes how to install, start, and stop modules by usingthe **wscp** Module operations. Modules can be deployedindependently of the enterprise applications into which they arepackaged.

Administering enterprise applications

An enterprise application is a collection of resources (such as XML files,enterprise beans, servlets, HTML files, and JSP files) that work together toperform a business function. An application server, combined with a Webserver, makes the enterprise application available to users. Enterpriseapplications are installed and configured as a single unit.

The **wscp** EnterpriseApp commands allow you to perform variousoperations on enterprise applications, including the following tasks:

- Install an enterprise application from an enterprise archive (EAR)file. In addition to installing the application, you can optionallyspecify the default application server on which its modules run, assignsecurity roles, specify JNDI mappings, and specify data sources for enterprisebeans.
- Start and stop instances of an enterprise application.
- List the modules that compose an enterprise application.
- List the nodes onto which an enterprise application is deployed.

The **wscp** EnterpriseApp install operation installs an enterpriseapplication from an EAR file. The syntax of this operation is:

```
EnterpriseApp install nodeName earFileName [other options]
```

where:

- *nodeName* -- The fully-qualified name of the WebSphereApplication Server node onto which the application is being installed.
- *earFileName*-- The full path name of the EAR filecontaining the application.
- *other options*-- Represents the following enterpriseapplication attributes that can optionally be set at installation:
  - **-defappserver** *appServName*-- Installs allmodules are installed on the specified application server. Specify thefully-qualified name of the application server onto which the enterpriseapplication is installed, such as/Node:DS1/ApplicationServer:DSServ/.

    Note:

    You can specify either this option or the -moduleappservers option, but notboth.
  - **-moduleappservers** {*moduleName appServName*}--Allows enterprise application modules to be installed on different applicationservers. You must specify a Tcl list of module URI-application serverpairs, including the fully-qualified name of the application server (forexample, {shoppingCart.JAR/Node:DS1/ApplicationServer:DSAux/}).

    Note:

    You can specify either this option or the -defappserver option, but notboth.
  - **-appname** *applicationName* -- The name of theenterprise application.
  - **-userroles** {*roleName userName*}-- Defines J2EEsecurity role to user mappings. Specify a Tcl list of role name anduser name pairs, such as {{Manager Mary}}.
  - **-grouproles** {*roleName groupName*}-- Defines J2EEsecurity role to user group mappings. Specify a Tcl list of role nameand group name pairs, such as {{Manager AdminGroup}}.
  - **-specialroles** {*roleName specialGroupName*}--Defines J2EE security role to special group mappings. (Special groupsare the predefined user groups Everyone and AllAuthenticatedUsers.)Specify a Tcl list of role name and special group name pairs, such as{{Manager Everyone}}.
  - **-runasroles** {*roleName identity password*}--Defines the security roles and execution identities that enterprise beanmethods run under in the enterprise application. A password must alsobe specified, for example {Manager Mary marypwd}.
  - **-resourcereferences** *resourceRefName* Specifies howmodule resource references are mapped to JNDI names. The type ofmapping depends on the type of module, but the mapping is always specified asa Tcl list of resource reference-JNDI name pairs.
    - Enterprise bean resource references are mapped as {*module_URI*::*bean_name*::*resource_ref_nameJNDIName*}
    - Web archive (WAR) resource references are mapped as{*module_URI::resource_ref_name*}. The referencesin WAR modules do not require bean names.
  - **-ejbnames**{*module_URI*::*bean_nameJNDIName*}-- Specifies the JNDI names for enterprise beans as aTcl list of pairs, where *module_URI* is the name of the JAR filecontaining the enterprise bean module. This option is only valid forJAR modules.
  - **-modvirtualhosts** {*module_URIvirtual_host_name*}-- Specifies the virtual host mapping for a WARmodules as a Tcl list of pairs, where *module_URI* is the name of theWAR file containing the web archive module. Enter the virtual hostalias for *virtual_host_name* (such as default_host) not thefully-qualified **wscp** virtual host name (such as/VirtualHost:default_host/)
  - **-ejbreferences**--Specifies the JNDI names for enterprisebean references. The type of mapping depends on the type of module, butthe mapping is always specified as a Tcl list of enterprise beanreference-JNDI name pairs.
    - Enterprise bean references are of the form {*module_URI*::*bean_name*::*ejb_ref_nameJNDIName*}
    - Web archive references are of the form{*module_URI::ejb_ref_name*}. The references inWAR modules do not require bean names.
  - **-ejbdatasources** {*module_URIJNDI_name*}--Sets the JNDI name of the default data source forJAR files as a Tcl list of pairs, where *module_URI* is the name ofthe JAR file. This option is only valid for JAR files.
  - **-cmpdatasources** {module_URI::bean_nameJNDIName}--Specifies the data source for entity beans withcontainer-managed persistence (CMP) as a Tcl list of pairs, where*module_URI* is the name of the JAR file.
  - **-redeploy**--Forces the archive to be installed, regardlessof whether it has been deployed before.
  - **-dbname** *name* --Specifies the name of theenterprise application database. It can only be used when an enterpriseapplication is first installed or when an application is reinstalled using the**-redeploy** option.
  - **-schemaname** *schema* -- Specifies the databaseschema to be used in the enterprise application database. It can onlybe used when an enterprise application is first installed or when anapplication is reinstalled using the **-redeploy** option.
  - **-dbtype** *type* -- Specifies the type ofapplication database, where *type* is one of the following supporteddatabases:
    - DB2UDBWIN_V72
    - DB2UDBOS390_V6
    - DB2UDBAS400_V4R5
    - INFORMIX_V92
    - MSSQLSERVER_V7
    - ORACLE_V8
    - SQL92
    - SQL99
    - SYBASE_V1192
    - MYSQL_V323

    It can only be used when an enterprise application is first installed orwhen an application is reinstalled using the **-redeploy**option.

To view a complete list of enterprise application attributes, use theEnterpriseApp attributes command. See Querying (displaying) attributes for more information on displaying object attributes.

The following command example shows how to install an enterpriseapplication. The node under which the application runs is DS1; thedefault application server under which its modules are installed isDSserv; and the security role mappings assign all authenticated users tothe Client role and the site administrator to the Administrator role.

```
wscp> EnterpriseApp install /Node:DS1/ C:/drugstore/DrugStoreApp.ear -defappserver /Node:DS1/ApplicationServer:DSserv/-userroles
{{Administrator SiteAdmin} {Supervisor Mary}} -specialroles {Client AllAuthenticatedUsers}
```

The following command example starts an instance of the DrugStoreenterprise application, then waits 60 seconds to see if it reallystarted. If the application does not start within this time period, theEnterpriseApp start operation is assumed to have failed.

```
wscp> EnterpriseApp start /EnterpriseApp:DrugStore/ -wait 60
```

The following command example stops an instance of the DrugStore enterpriseapplication:

```
wscp> EnterpriseApp stop /EnterpriseApp:DrugStore/
```

The following command example lists the modules that comprise an enterpriseapplication:

```
wscp> EnterpriseApp listmodules
/EnterpriseApp:DrugStore//Node:DS1/ApplicationServer:DSserv/Module:ShoppingCart//Node:DS1/ApplicationServer:DSserv/Module:StoreFront//Node:DS1/ApplicationServer:DSserv/Module:Accounts//Node:DS2/ApplicationServer:DSserv/Module:InventoryMgr/
```

The following command example lists the nodes onto which an enterpriseapplication is deployed:

```
wscp> EnterpriseApp listnodes /EnterpriseApp:DrugStore//Node:DS1/ /Node:DS2/
```

Several Tcl scripts for installing enterprise applications areavailable. Each installs one of the sample applications provided withWebSphere Application Server.

- t1.tcl installs jmsample.ear
- t2.tcl installs perfServletApp.ear
- t3.tcl installs sampleApp.ear
- t4.tcl installs Samples.ear
- t5.tcl installs ServletCacheMonitor.ear
- t6.tcl installs soapsamples.ear
- t7.tcl installs TradeSample.ear

The following example shows the script t1, which is used to install thesample enterprise application jmsample.

```
set mynode xxxx set instdir i:/WebSphere/AppServer/installableApps/ set earfile jmsample.ear set appname "MailSampleApp" set
sname "Default Server" set mailth "AHost" set mailSessName "DefaultMailSession" set vhostName "default_host"   # create
prerequiste objects  # # first, the MailSession set mailtransportattr [list MailTransportHost $mailth] set attributelist [list
$mailtransportattr]   MailSession create /MailSession:$mailSessName/ -attribute $attributelist   # next, the virtual host set
aliaslist [list *:80 *:9080] set aliasattr [list AliasList $aliaslist] set attributelist [list $aliasattr]   VirtualHost create
/VirtualHost:$vhostName/ -attribute $attributelist   # Now install the application # #     construct -modvirtualhosts option set
modhost1 [list mtcomps.war $vhostName] set modhosts  [list $modhost1]   #     construct -resourcereferences option set resref1
[list mtcomps.war::mail/MailSession9 mail/$mailSessName] set resref2 [list deplmtest.jar::MailEJBObject::mail/MailSession9
mail/$mailSessName] set resrefs [list $resref1 $resref2]     EnterpriseApp install /Node:$mynode/ $instdir$earfile -appname
$appname -defappserver /Node:$mynode/ApplicationServer:$sname/ -modvirtualhosts $modhosts -resourcereferences $resrefs
```

This script (and the others listed in this section) is available in 6.6.0.2.2.5: Sample Tcl procedures and scripts.

Administering modules

The resources that compose an enterprise application are grouped accordingto function into modules. Enterprise applications can contain threetypes of modules:

- Enterprise JavaBeans (EJB) modules, which contain Java class files forenterprise beans, Java class files for functionality that is not included withthe J2EE platform, and an EJB deployment descriptor.
- Web modules, which contain Java class files for servlets and applets;JSP files and their helper classes; static HTML, sound, image, video, andother content files; and a Web deployment descriptor.
- Application client modules, which contain the Java classes that implementthe client and an application deployment descriptor.

The **wscp** Module operations allow you to deploy modulesindependently from enterprise applications. Application client and EJBmodules are installed from Java archive (JAR) files; Web modules areinstalled from Web archive (WAR) files. When you install a module,**wscp** actually generates a simple EAR file with the specified JAR orWAR file as its only module. It then installs the EAR file. Youcan optionally assign such things as security roles, specify JNDI mappings,and specify data sources for enterprise beans.

The **wscp** Module install operation installs an enterpriseapplication module. The syntax of this operation is:

```
Module install nodeName module_name-moduleappservers module_URI appServName [other options]
```

where:

- *nodeName* -- The fully-qualified name of the WebSphereApplication Server node onto which the module is being installed.
- *module_name*-- The full path name of the file containingthe module.
- **-moduleappservers** {*module_URI appServName*}--Specifies the application server onto which a modules is installed.Specify the file name of the JAR or WAR file and the fully-qualified name ofthe application server, such as {shoppingCart.JAR/Node:DS1/ApplicationServer:DSAux/}.
- *other options*-- Represents the following module attributesthat can optionally be set at installation:
  - **--contextroot** *contextroot*--The contextroot for a WAR file.
  - **-appname** *applicationName* -- The name of theenterprise application to which the module belongs.
  - **-userroles** {*roleName userName*}-- Defines J2EEsecurity role to user mappings. Specify a Tcl list of role name anduser name pairs, such as {{Manager Mary}}.
  - **-grouproles** {*roleName groupName*}-- Defines J2EEsecurity role to user group mappings. Specify a Tcl list of role nameand group name pairs, such as {{Manager AdminGroup}}.
  - **-specialroles** {*roleName specialGroupName*}--Defines J2EE security role to special group mappings. (Special groupsare the predefined user groups Everyone and AllAuthenticatedUsers.)Specify a Tcl list of role name and special group name pairs, such as{{Manager Everyone}}.
  - **-runasroles** {*roleName identity password*}--Defines the security roles and execution identities that enterprise beanmethods run under in the enterprise application. A password must alsobe specified, for example {Manager Mary marypwd}.
  - **-resourcereferences** *resourceRefName* Specifies howmodule resource references are mapped to JNDI names. The type ofmapping depends on the type of module, but the mapping is always specified asa Tcl list of resource reference-JNDI name pairs.
    - Enterprise bean resource references are mapped as {*module_URI*::*bean_name*::*resource_ref_nameJNDIName*}
    - Web archive (WAR) resource references are mapped as{*module_URI::resource_ref_name*}. The referencesin WAR modules do not require bean names.
  - **-ejbnames**{*module_URI*::*bean_nameJNDIName*}-- Specifies the JNDI names for enterprise beans as aTcl list of pairs, where *module_URI* is the name of the JAR filecontaining the enterprise bean module. This option is only valid forJAR modules.
  - **-ejbreferences**--Specifies the JNDI names for enterprisebean references. The type of mapping depends on the type of module, butthe mapping is always specified as a Tcl list of enterprise beanreference-JNDI name pairs.
    - Enterprise bean references are of the form {*module_URI*::*bean_name*::*ejb_ref_nameJNDIName*}
    - Web archive references are of the form{*module_URI::ejb_ref_name*}. The references inWAR modules do not require bean names.
  - **-ejbdatasources** {*module_URIJNDI_name*}--Sets the JNDI name of the default data source forJAR files as a Tcl list of pairs, where *module_URI* is the name ofthe JAR file. This option is only valid for JAR files.
  - **-cmpdatasources** {module_URI::bean_nameJNDIName}--Specifies the data source for entity beans withcontainer-managed persistence (CMP) as a Tcl list of pairs, where*module_URI* is the name of the JAR file.
  - **-dbname** *name* --Specifies the name of thedatabase.
  - **-schemaname** *schema* -- Specifies the databaseschema to be used.
  - **-dbtype** *type* -- Specifies the type of database,where *type* is one of the following supported databases:
    - DB2UDBWIN_V72
    - DB2UDBOS390_V6
    - DB2UDBAS400_V4R5
    - INFORMIX_V92
    - MSSQLSERVER_V7
    - ORACLE_V8
    - SQL92
    - SQL99
    - SYBASE_V1192
    - MYSQL_V323
- **-defappserver** *appServName*-- Installs allmodules are installed on the specified application server. Specify thefully-qualified name of the application server onto which the enterpriseapplication is installed, such as/Node:DS1/ApplicationServer:DSServ/.

  Note:

  You can specify either this option or the -moduleappservers option, but notboth.
- **-moduleappservers** {*moduleName appServName*}--Allows enterprise application modules to be installed on different applicationservers. You must specify a Tcl list of module URI-application serverpairs, including the fully-qualified name of the application server (forexample, {shoppingCart.JAR/Node:DS1/ApplicationServer:DSAux/}).

  Note:

  You can specify either this option or the -defappserver option, but notboth.
- **-appname** *applicationName* -- The name of theenterprise application.

To view a complete list of enterprise application attributes, use theEnterpriseApp attributes command. See Querying (displaying) attributes for more information on displaying object attributes.

The following command example shows how to install a Web module. Themodule type and active module type are web; the relative URI of the WARfile containing the module is storefront.war; the context root is/WebApp.

```
wscp> Module install /Node:DS1/ C:/storefront.war -contextroot /WebApp
```

6.6.0.2.2.4.10: Configuring server groups and clones

The **wscp** ServerGroup operations allow you to configure servergroups and clones. Server groups are templates for creating clones ofapplication servers. Clients see the clones associated with a servergroup as a single application server image. The clones of anapplication server automatically participate in workload management.

The **wscp** ServerGroup operations included are as follows:

- Creating a server group from an existing application server instance
- Starting and stopping a server group and its clones
- Adding a clone to a server group

See Managing workloads for information on servergroups, clones, and workload management.

The following example creates a server group from an existing instance ofan application server, specifying its name, server selection policy, and thevalue of an environment variable:

```
wscp> ServerGroup create /ServerGroup:EjbServerGroup/ -baseInstance /ApplicationServer:EjbAppServ/
\-serverGroupAttrs {{EJBServerAttributes {SelectionPolicy roundrobin}} {Environment {VAR1 0}}}
```

The following example starts a server group after a delay of 30seconds:

```
wscp> ServerGroup start /ServerGroup:EjbServerGroup/ -wait 30
```

The following example stops a server group, using the -force option to stopit regardless of any other conditions:

```
wscp > ServerGroup stop /ServerGroup:EjbServerGroup/ -force
```

The following example adds a clone to the server groupEjbServerGroup. The clone runs on node Server1:

```
wscp > ServerGroup clone /ServerGroup:EjbServerGroup/ -cloneAttrs{{Name Clone1}} -node
/Node:Server1/
```

The fully qualified name of the new application server instance is/Node:Server1/ApplicationServer:Clone1/.

The following example lists all clones that are associated with a servergroup:

```
wscp > ServerGroup listClones /ServerGroup:EjbServerGroup/
/Node:Server1/ApplicationServer:EjbServerGroup1/ /Node:Server1/ApplicationServer:EjbServerGroup2/
```

6.6.0.2.2.4.11: Administering EIS connections

The **wscp** J2CResourceAdapter and J2CConnectionFactory operationsallow you to manage connections to enterprise information system (EIS) backendsystems, such as PeopleSoft or Customer Information Control System(CICS). These operations can be used to perform the followingtasks:

- Create a J2C resource adapter by using the J2CResourceAdapter createcommand. A J2C resource adapter is a module that enables connections toa specific back-end system.
- Install a J2C resource adapter on a node by using the J2CResourceAdapterinstall command. You can install a J2C resource adapter on multiplenodes.
- Create an instance of a J2C connection factory by using theJ2CConnectionFactory create command. A J2C connection factory is a setof connection configuration values that are used by the J2C connection poolmanager to specify how applications connect to a backend system.

The following command example creates a J2C resource adapter. TheArchiveFile attribute is required.

```
wscp> J2CResourceAdapter create /J2CResourceAdapter:ResAd/  -attributes {{ArchiveFile
C:/apps/resources.rar}}
```

The following command example installs a J2C resource adapter. Thenode and resource archive (RAR) file attributes are required.

```
wscp> J2CResourceAdapter install /J2CResourceAdapter:myRes/ -node /Node:Appserv1/ -rarfile
"C:/apps/resources.rar"
```

The following command example creates a J2C connection factoryobject. You must specify the name of the J2C resource adapter withwhich the connection factory is associated with. The optionalattributes specify the timeout period in milliseconds and the maximum numberof connections supported by that connection factory.

```
wscp> J2CConnectionFactory create
/J2CResourceAdapter:ResAd/J2CConnectionFactory:factory1/-attributes {{ConnectionTimeout 1000}
{MaxConnections 10}}
```

For more information on administering J2C connections, see Administering J2C related administrative objects.

6.6.0.2.2.4.12: Administering the Java Message Service (JMS)

The Java Messaging Service (JMS) enables applications to exchange data inthe form of messages. The **wscp** operations JMSProvider,JMSConnectionFactory and JMSDestination enable you to configure JMS clientsfrom the command line as follows:

- Create a JMS provider by using the JMSProvider create command. AJMS provider implements the JMS messaging interfaces.
- Install a JMS provider on a node by using the JMSProvider installcommand. You can install JMS providers on multiple nodes.
- Create an instance of a JMS connection factory by using theJMSConnectionFactory create command. A client uses a JMS connectionfactory object to connect to the messaging system.
- Create an instance of a JMS destination by using the JMSDestination createcommand. A client uses a JMS destination object to specify the targetof the messages it sends or the source of the messages it receives.

The following example command creates a JMS provider. TheExternalInitialContextFactory and ExternalProviderURL attributes arerequired.

```
wscp> JMSProvider create /JMSProvider:Prov1/ -attribute{{ExternalInitialContextFactory
com.ibm.websphere.naming.WsnInitialContextFactory}{ExternalProviderURL iiop://localhost}}
```

The following example command installs a JMS provider on the nodedev1:

```
wscp> JMSProvider install /JMSProvider:Prov1/ -node /Node:dev1/ -jarFile
"/opt/WebSphere/ApplicationServer/jars/JMS.jar"
```

The following example command creates an instance of a JMS connectionfactory. The ExternalJNDIName attribute is required.

```
wscp> JMSConnectionFactory create /JMSProvider:Prov1/JMSConnectionFactory:connect1/ -attribute
{{ExternalJNDIName jms/connect1}}
```

The following example command creates an instance of a JMSdestination. The ExternalJNDIName attribute is required.

```
wscp> JMSDestination create /JMSProvider:Prov1/JMSDestination:dest1/ -attribute {{ExternalJNDIName
jms/dest1}}
```

For more information, see Administering messaging and JMSproviders.

6.6.0.2.2.4.13: Administering JavaMail sessions

The **wscp** MailSession operations are used to manage JavaMailsessions. JavaMail enables applications to compose, send and receiveelectronic mail.

The following example command shows how to create a new JavaMail sessionthat uses the SMTP mail protocol. The session name andMailTransportHost attributes are required.

```
wscp> MailSession create /MailSession:Session1/ -attribute {{MailTransportHost mailhost}
{MailTransportProtocol smtp}}
```

The following example command shows how to destroy a JavaMailsession:

```
wscp> MailSession remove /MailSession:Session1/
```

6.6.0.2.2.4.14: Administering URL providers and URLs

The **wscp** URL and URLProvider operations allow you to create andinstall instances of URLs and URL providers.

The following example command creates an instance of a URL provider.The name, Protocol and StreamHandlerClass attributes are required.

```
wscp> URLProvider create /URLProvider:Prov1/ -attribute {{Protocol http}{StreamHandlerClass
com.ibm.ejs.myStreamHandler}}
```

The following example command installs a URL provider on a node:

```
wscp> URLProvider install /URLProvider:AppProv/ -node /Node:AppServ1/ -jarFile
"C:/jars/provider.jar"
```

The following example command creates an instance of a URL. You mustspecify a URL provider for the URL; the Spec attribute is required:

```
wscp> URL create /URLProvider:Prov1/URL:testURL/ -attribute {{Spec mySpec}}
```

6.6.0.2.2.5: Sample Tcl procedures and scripts

The following files contain example Tcl procedures and scripts.

- attrs.tcl script. Procedure for printingthe attributes of all or selected objects.
- contain.tcl script. Procedure fordisplaying the containment hierarchy for all or select objects.
- disp.tcl script. Procedure fordisplaying the output of the show operation in a readable format--oneattribute per line.
- init.tcl script. Many of the examplescripts and procedures require that the file init.tcl be loaded priorto running. This file initializes several variables used elsewhere andcontains useful Tcl procedures for accessing attributes, obtaining errorstatus information, and other tasks.
- modEnv.tcl script. Procedure formodifying or replacing values in the Environment attribute.
- servInfo.tcl script. Procedure fordisplaying the name and current state of all servers in a domain.
- The following scripts install the example enterprise applications providedwith WebSphere Application Server:
    - t1.tcl script. Procedure for installingthe sample application jmsample.ear.
    - t2.tcl script. Procedure for installingthe sample application perfServletApp.ear.
    - t3.tcl script. Procedure for installingthe sample application sampleApp.ear.
    - t4.tcl script. Procedure for installingthe sample application Samples.ear.
    - t5.tcl script. Procedure for installingthe sample application ServletCacheMonitor.ear.
    - t6.tcl script. Procedure for installingthe sample application soapsamples.ear.
    - t7.tcl script. Procedure for installingthe sample application TradeSample.ear.

6.6.0.2.2.6: Migrating wscp scripts from version 3.5.x to version 4.0

This section summarizes some of the changes to **wscp** in version4.0 that can affect existing **wscp** tcl scripts. It isintended as a guide for determining whether these scripts need to be updatedand is not an exhaustive description of **wscp** changes.

You need to retest all scripts created under earlier versions of**wscp** before migrating them to a production environment that runsthe latest version of the software.

Discontinued objects

The following objects are no longer supported in **wscp**.

- EJBContainer
- EnterpriseBean
- Servlet
- ServletEngine
- ServletRedirector
- SessionManager
- WebApplication
- WebResource
- UserProfile

Although they can no longer be administered on an individual basis,enteprise beans, servlets, and Web applications that have been packaged intoenterprise applications or modules can be installed, started, and stopped withthe EnterpriseApp and Module operations. See 6.6.0.2.2.4.8: Creating an enterprise application.

Renamed objects and functional or syntax changes

The following objects have been renamed or their functionality and syntaxhave changed. See the listed articles for more information on their newsyntax and options.

Old name	**New name**	**Functional changes**
EnterpriseApplication	EnterpriseApp	Application components are now packaged in modules. You no longerhave to explicitly install all components of an enterprise application.See 6.6.0.2.2.4.8: Creating an enterprise application.
Model	ServerGroup	Can only create server groups and clones of application servers.See 6.6.0.2.2.4.10: Configuring server groups and clones.
DataSource	DataSource	Syntax changes. See 6.6.0.2.2.4.7: Creating and installing drivers and data sources.

New objects and new functionality

The following objects are new for version 4.0 and represent areas offunctionality that can now be handled in scripts. See the listedarticles for more information on these objects and services.

New objects and services	New functional areas	Documentation
Module	Java 2 Enterprise Edition (J2EE) modules	6.6.0.2.2.4.8: Creating an enterprise application
J2CConnectionFactory	Enterprise information systems (EIS) connectivity	6.6.0.2.2.4.11: Administering EIS connections
J2CResourceAdapter		
JMSProvider	Java Message Service (JMS)	6.6.0.2.2.4.12: Administering the Java Message Service (JMS)
JMSConnectionFactory		
JMSDestination		
URL	URL	6.6.0.2.2.4.14: Administering URL providers and URLs
URL provider		
PmiService	Performance data	6.6.0.2.2.3.10: Monitoring performance
SecurityConfig	Global security settings	6.6.0.2.2.3.11: Setting global security defaults
SecurityRoleAssignment	J2EE security roles	6.6.0.2.2.3.12: Managing security roles

6.6.0.2.2.7: Using the wscpCommand interface

Use the**com.ibm.ejs.sm.ejscp.wscpcommand.WscpCommand**interface to embed **wscp** operations in Java applications.This enables applications to evaluate **wscp** operations withoutrepeated startup costs. The interface is linked through the product_installation_root/lib/wscp.jar file.

Constructors

`WscpCommand(String node, String port)`

where:

- *node*--The name of the WebSphere Application Server node onwhich the **wscp** operation is executed (such as localhost).
- *port*--The port number of the administrative server.

`WscpCommand()`

The default values for this constructor are localhost for node and 900 forport (the default port number of the administrative server).

Public methods

`WscpResult evalCommand(String command)`

This method evaluates a **wscp** operation and returns theresults. Use the same syntax as you do for interactive or scripted**wscp** operations. Using abbreviated command names is notrecommended.

The results of the **evalCommand** method are encoded in a**com.ibm.ejs.sm.ejscp.wscpcommand.WscpResult**object. The following methods on this object can be used to evaluatethe returned value:

- **toString**-- Returns a string representation of the resultsof the wscp operation.

  Note:

  > The WscpCommand interface has no way of knowing the format of the expectedoutput from a **wscp** operation. When the return format is notan attribute-value pair or list, use the **toString** method toevaluate the output. The application programmer must then write code toparse the output.

- **success**-- Returns a boolean value indicating whether thewscp operation was successful.
- **listToVector**-- If the expected result of the wscpoperation is a list, use this method to return a vector containing the listelements.
- **attribPairsToVector**-- If the expected result of the wscpoperation is a list of attribute-value pairs, use this method to return avector containing the list. Each element of the vector is anattribute-value pair.
- **attribPairsToHashTable**-- If the expected result of thewscpoperation is a set of attribute-value pairs, use this method to return a hashtable containing the values keyed on the attributes.

`String getErrorInfo()`

This method returns the status of an **evalCommand** method whoseresults are returned in a WscpResult object.

- If the **wscp** operation specified in the **evalCommand**executed successfully, the **getErrorInfo** method returns an emptystring.
- If the **wscp** operation did not execute successfully, the**getErrorInfo** method returns the exception information that wasreceived from **wscp**.

The following is an example of how to use the **evalCommand** and**getErrorInfo** methods.

```
String cmd = "ApplicationServer list";WscpResult results = wscpCommand.evalCommand(cmd);if
(!results.success()) {   System.out.println("command failed; exception information: " +
results.getErrorInfo());   }
```

Utility class

The **WscpCommand** interface contains a utility class,**WscpQualifiedName**, that allows the manipulation of fully qualfiedwscp object names. The constructor is as follows:

`WscpQualifiedName(fullyqualifiedname)`

where *fullyqualifiedname* is the fully-qualified **wscp**name of an object.

The methods for this class are as follows:

- **getName**(*level*)--Gets the object name at thespecified containment level.
- **getObject**(*level*)--Gets the object type at thespecified containment level.
- **toString**--Returns a string representation of the objectname.
- **numberOfLevels**--Returns the number of containmentlevels.

The **WscpQualifiedName** class contains the following constants,which define the containment levels at which various components arenamed.

- ENTERPRISE_APPLICATION
- DATASOURCE
- JDBC_DRIVER
- NODE
- SERVER_GROUP
- VIRTUAL_HOST
- APPLICATION_SERVER
- GENERIC_SERVER
- WEB_RESOURCE
- WEB_APPLICATION
- MODULE
- JMS_PROVIDER
- JMS_CONNECTION_FACTORY
- JMS_LISTENER
- J2C_RESOURCE_FACTORY
- J2C_CONNECTOR
- MAIL_SESSION
- URL
- URL_PROVIDER

An example of how this class is used is as follows:

```
String name = "{/Node:MyNode/ApplicationServer:MyAppServer/}"WscpQualfiedName qName = new
WscpQualifiedName(name);qName.numberOfLevels() returns 2qName.getObject(2) returns
"ApplicationServer"qName.getName(2) returns
"MyAppServer"qName.getName(WscpQualifiedName.APPLICATION_SERVER)       returns "MyAppServer"
```

Security

The **wscp** command-line tool runs with security enabled because itis started by using SAS from the command line, as you can see in thewscp.bat (Windows) and wscp.sh (Unix) files. However, the**WscpCommand** interface cannot guarantee support for a server withsecurity enabled because it is used in client programs. If the Javaprogram is started using SAS, **WscpCommand** methods execute withsecurity enabled.