

Appendix A. Classpath

The following table contains information gathered over time by WebSphere developers and testers. It is provided as reference material but has not been fully verified by the WebSphere development team.

Component Interaction	Where dependent classpath needs to be specified	What resources need to be there	Other options to try/ (Do they work?)
Deploying Enterprise JavaBeans through the Administrative Console	"Dependent Classpath" for the node object ¹	Utility classes needed by the beans being loaded	com.ibm.ejs.sm.adminserver.classpath in admin.config/(?)
Servlet looking up an enterprise bean and calling methods of that bean	-classpath option for the command line arguments of the application server	home and remote interfaces of the enterprise bean and the deployed stubs for the bean/home	Classpath for application in web application that corresponds with the servlet./(Yes, it works.)
Session bean looking up an entity bean. Both are in the same EJBContainer (both in the same deployed JAR)	-classpath option for the command line arguments of the application server	Home and remote interfaces of the entity bean, and the deployed stubs for the bean/home	Some have tried to get by without specifying this the JAR file in any classpath, assuming that the EJB class loader should know where the deployed JAR is./ (experience says that it doesn't work)
Servlet/other component using a class that has native methods	-classpath option for the command line arguments of the application server	classes that have the native methods	com.ibm.ejs.sm.adminserver.classpath in admin.config/(?/yes) classpath for application in web application/(no)
Components that require CORBA-related classes that may conflict with what exists on the whole-scope classpath	depends on the ideal order of resources on the whole-scope classpath: -classpath option for the command line arguments of the application server prepends to the existing classpath com.ibm.ejs.sm.adminserver.classpath in admin config allows you to set the order of the resources in the classpath		Classpath for application in web application/(maybe, in certain cases?)

¹There seems to be a problem on AIX (and possibly other platforms) regarding the length of this classpath.

- Servlets using JNI must be added to the classpath of the application server. Do this by adding the path to your servlet in the command line arguments field of the application server.
- DLLs have to go in the WebSphere **bin** directory because there is no equivalent `use.user.libpath` directory.
- Any JavaBeans or programmatic elements should go in the web application classpath. These will be reloaded when changed.

Setting Classpaths

The IBM WebSphere Application Server environment has these classpath components:

1. The Application Server (JVM) Classpath
2. The EJB JAR Classpath
3. The Web Application Classpath

Appendix A. Classpath

Application Server (JVM) Classpath

Where to set	<p>There is one application server classpath for each application server in the IBM WebSphere Application Server environment. Each application server corresponds to a JVM.</p> <p>Set the classpath in one of two ways:</p> <ul style="list-style-type: none"> • Use an administrative client to set the Command Line Arguments field of the application server. Specify the classpath with the flag <code>-classpath</code>. • Use an administrative client to set the application server Environment property to include a <code>CLASSPATH</code> variable and value.
Scope	This classpath is visible to all enterprise beans, servlets, and JSP files contained by an application server.
Behavior	The classes in this classpath are loaded by the JVM primordial <code>ClassLoader</code> . Hence, after the application server is started, any changes to this classpath will not take effect until the application server is stopped and started again.
Reloadable?	Classes loaded from this classpath will not be reloaded if they are changed while the application server is running.
Typical contents	<ul style="list-style-type: none"> • Classes referenced from servlets whose objects are added to sessions, such objects that are serialized and whose classes must not be reloaded. • Classes that call Java Native Interface (JNI) methods. Those classes and any imported classes must be placed in the application server classpath to prevent loading errors. • For AS/400, any servlet classes and helper classes on which you are going to run the AS/400 system debugger or IBM Distributed Debugger. <p>Note that classes which are put in this classpath should not reference other classes that cannot be found in this classpath.</p>
ClassLoader	JVM primordial <code>ClassLoader</code>
Trace component	<p>To see the contents of this classpath, enable the trace component:</p> <pre>com.ibm.ejs.sm.active.ActiveEJBServerProcess=all=enabled</pre> <p>for the WebSphere administrative server.</p> <p>Note that this trace component should be enabled for the administrative server because that server is responsible for constructing the application server command line before starting the application server process.</p> <p>Enable the trace before starting the application server.</p>

EJB JAR Classpath

Where to set	This classpath is automatically constructed by each application server. The administrator can add EJB JAR files to this classpath by deploying new EJB JAR files and starting the
--------------	---

Appendix A. Classpath

	<p>enterprise beans while the application server is either running or stopped. EJB JAR files are put in the classpath in the order in which they are deployed.</p> <p>The Dependent Classpath for the administration server node is prepended before each EJB JAR file in this classpath, to provide means of modifying this classpath's behavior.</p> <p>To set the node Dependent Classpath, use an administrative client to access node properties.</p>
Scope	This classpath is visible to all enterprise beans, servlets, and JSP files contained by an application server.
Behavior	<p>This classpath is used to locate EJB JAR classes and their dependent classes while the application server is running.</p> <p>The Dependent Classpath of the node contains the dependent classes and precedes all JAR file in this classpath. It is checked during runtime to locate any helper classes for an enterprise bean that are not contained in the EJB JAR file of the enterprise bean.</p> <p>The Dependent Classpath is also used during the EJB JAR deployment process for the same purpose. However, it is important to note the Dependent Classpath is used by all application servers in a node, when the application servers are running. Hence, it should not contain any common classes requiring different implementations for each application server on the same node. Instead, place these common classes in the application server (JVM) classpaths.</p>
Reloadable?	Classes loaded from this classpath will not be reloaded if they are changed while the Application Server is running. However, new JAR files or directories can be added to this classpath by using the EJB JAR deployment process even while the application server is running.
Typical contents	<ul style="list-style-type: none"> • EJB JAR files containing EJB classes are automatically put in this classpath. The node Dependent Classpath is prepended to each EJB JAR file in this classpath. • The Node Dependent classpath should contain JAR files or directories containing helper classes referenced from EJB classes that are not present in the EJB JAR files. <p>Note, if these helper classes need to have different implementations for each application server on the same node, then these classes should be kept in the Dependent Classpath only during the EJB JAR deployment process, and application servers should be kept stopped.</p> <p>After deployment of the EJB JAR file, their containing JAR files or directories (containing the helper classes) should be moved from the Dependent Classpath to the respective application server JVM classpath before starting the application servers.</p>
ClassLoader	JARClassLoader. There is one instance of it for each application server accessible through the EJBJARManager.
Trace component	<p>Debug the JARClassLoader by enabling trace for the component:</p> <pre>com.ibm.ejs.util.JAR.*=all=enabled</pre>

Appendix A. Classpath

for the application server.

Enable the trace while the application server is running, before accessing the EJB class.

Keep in mind that the Dependent Classpath is referenced by all application servers running on the same administrative server node.

Web Application Classpath

Where to set	Use an administrative client to specify the Classpath setting of a Web application.
Scope	The classpath is visible to all servlets and JSP files in the corresponding Web application.
Behavior	<p>This classpath is monitored and all components (JAR or class files) are reloaded whenever it is automatically detected that a component has been updated.</p> <p>A new JAR file is automatically loaded upon detection in any directory already contained in this classpath. This means that it is not necessary to explicitly specify a new JAR file in this classpath. Rather, it suffices to just put the JAR file in a directory that is already present in the classpath.</p> <p>Automatic reloading at the Web application level keeps all of the application components synchronized and conserves system resources compared to the Version 2.0x reloading scheme.</p> <p>Version 3.5 does not support remote servlet loading (that is, loading servlets across a network). All application components must be on the machine containing the application server hosting the application.</p>
Reloadable?	Yes
Typical contents	<ul style="list-style-type: none"> • Directories or JAR files with Servlet classes. • Directories or JAR files with helper classes that are not included in the servlet JAR file and that are expected to be reloadable. • Directories or JAR files with Access Bean classes that are referenced from servlet classes and that refer to enterprise beans.
ClassLoader	PowerClassLoader. There is one for each Web application.
Trace component	<p>Trace the Web application classpath by tracing the component</p> <pre>com.ibm.servlet.classloader.*=all=enabled</pre> <p>on a running application server.</p> <p>Enable the trace before invoking a servlet or JSP file in the Web application from a Web browser.</p>

Appendix A. Classpath

For improved performance, set the automatic reloading property to false in the properties of Web applications contained by production application servers.

Administrative Server Classpath

One last classpath you should be aware of is the WebSphere administrative server classpath, though it is recommended that you do not modify the classpath for the use of particular applications.

The administrative server classpath is set automatically when you install the product. The default classpath setting contains all of the IBM WebSphere Application Server APIs (the JAR files in the lib directory of the product installation root).

The administrative server classpath value depends on how you start the administrative server.

If you use one of these methods to start the administrative server:

- The *startupServer.sh* script on UNIX platforms
- The Windows NT Services panel

then the administrative classpath is set based on the value of the property:

```
com.ibm.ejs.sm.adminserver.classpath
```

in the file:

```
installation_root/bin/admin.config
```

If you use the *adminserver.[bat|sh]* script to start the administrative server, the classpath is set based on its value as specified by the *adminserver.[bat|sh]* script.

The administrative server classpath is appended to the each application server JVM classpath by default. If it is necessary to override this behavior, use the property:

```
com.ibm.ejs.sm.adminServer.managedServerClassPath
```

in the file:

```
installation_root/bin/admin.config
```

Placing Files

When you deploy servlets, Web applications, and Enterprise JavaBean applications, ensure that the component files are in the correct directories. The Application Server administrative interface, and tools such as IBM WebSphere Studio and IBM VisualAge for Java, make it easy to deploy the components. If you manually deploy your servlets and applications, use this quick reference table as a guide.

Appendix A. Classpath

The Related information below contains step-by-step instructions for placing the files contained in a Web application.

File Description	File Extension	Directory Path
HTML documents and related static files	.html, .shtml, .jhtml, .gif, .au, and so on	These can be either served by the Web server, or placed in the Web application document root with the WebSphere file servlet enabled.
JavaServer Pages files	.jsp	Web application document root
Servlets that are to be reloaded	.class or .jar	Web application classpath If the servlets are in a package, mirror the package structure as subdirectories under the Web application classpath.
Servlet that are not to be reloaded	.class or .jar	Application server classpath
Servlet configuration file	.servlet	Directory that contains the servlet
Enterprise bean	.jar	Application server deployableEJBs directory
JavaBean (not an enterprise bean) or other object to be reloaded	.ser or .jar	Web application classpath
JavaBean (not an enterprise bean) or other object not to be reloaded, such as serialized objects and servlets that use Java Native Interface methods	.ser or .jar	Application server classpath
Java objects added to a session	.class, .jar, or .ser	Application server classpath This requirement applies to non-EJB objects in either of the following conditions: <ul style="list-style-type: none"> • Session persistence is enabled (the default setting). • The application server is part of a session cluster. <p>In a session cluster, be sure to place the objects in the application server classpath on each cluster host and cluster client.</p>

Appendix A. Classpath

		An object in the application server classpath is not reloaded when its source file changes.
Objects passed as arguments for remote calls		Application server classpath

Deploying Enterprise Applications

The person deploying an enterprise application should divide classes belonging to an enterprise application into at least two JAR files. If the application contains both servlets and enterprise beans, these classes should be in separate JAR files. The EJB JAR file should be deployed under an application server container.

The Servlet JAR file should be placed in the Web application "servlets" directory. All EJB helper classes can be either put in the EJB JAR file or can be placed in a separate JAR file which should be placed in the Dependent Classpath of the Administrative Server Node. All Servlet helper classes can be either placed in the Servlet JAR file or in a separate JAR file which should be placed in the Web Application servlets directory.

The following types of classes are exceptions:

- 1.Classes which call Java Native Interface (JNI) methods should be placed in the Application Server (JVM)'s classpath.
- 2.Classes which are passed to the HTTP session state should be placed in the Application Server (JVM)'s classpath.
- 3.Classes which are referenced from both Servlets and enterprise beans, such as method arguments and return values should be placed along with the other EJB helper classes.
- 4.Access Bean classes which are referenced from Servlets and which refer to enterprise beans should be placed along with Servlet helper classes.

Appendix B. General Unix Help

UNIX Shells

Whenever you login to a UNIX system you are placed in a program that is called the shell. You can see its prompt at the bottom left of your screen. The shell acts as a command interpreter; it takes each command and passes it to the operating system kernel (the essential center of a computer operating system, the core that provides basic services for all other parts of the operating system) where the action is performed. It then displays the results of this operation on your screen. Three common types of UNIX shells are Bourne shell, C shell, and Korn shell.

Bourne shell (sh) is the original UNIX shell written by Steve Bourne of Bell Labs. It is available on all UNIX systems. This shell does not have the interactive facilities provided by modern shells such as the C shell and Korn shell. You are advised to use another shell that has these features.

C shell (csh) was written at the University of California, Berkeley. It provides a C-like language with which to write shell scripts - hence its name.

Korn shell (ksh) was written by David Korn of Bell labs. It is now provided as the standard shell on UNIX systems. It is the most efficient shell.

	Bourne	C	Korn
command history	No	Yes	Yes
command alias	No	Yes	Yes
shell scripts	Yes	Yes	Yes
filename completion	No	Yes*	Yes*
command line editing	No	No	Yes*
job control	No	Yes	Yes

* not the default setting for this shell

To display the shell you are currently using, type the command **echo \$SHELL**. The shell will be identified by the shell command (sh, csh, ksh) at the end of the displayed path name.

To switch to a different shell, enter the shell command (sh, csh, ksh) at the system prompt.

UNIX File System

The UNIX file system is organized as a hierarchy of directories starting from a single directory called **root** which is represented by a / (slash). Immediately below the root directory are several system directories that contain information required by the operating system. As in DOS, '.' represents the current directory, and '..' represents the parent directory. Unlike DOS, the UNIX file system is case sensitive. UNIX does not use drive letters to differentiate between partitions, instead it "mounts" partitions at different places in the file system. For example, you might mount your

Appendix B. General Unix Help

CD-ROM drive at `/dev/cdrom`. To access files on a CD, you would change to the directory `/dev/cdrom`. For more information, see the manual pages or documentation for your operating system.

Working with Environment Variables

UNIX uses environment variables to configure the environment. Common system environment variables are *SHELL* and *PATH*. WebSphere looks for an environment variable called *JAVA_HOME* to determine where the Java Development Kit (JDK) is installed. You can use the `echo` command to display the value of an environment variable. The command `echo $PATH` displays the value of the environment variable *PATH*. The name of an environment variable is given in UPPER CASE. The \$ sign is a shell metacharacter that uses the value of the variable instead of its name. The commands to work with environment variables are different depending on which shell you are using.

Shell	View all set variables	Set a variable
Bourne/Korn	<code>set</code>	<code>VARNAME=VALUE</code> <code>export VARNAME</code>
C	<code>env</code>	<code>setenv VARNAME VALUE</code>

Common UNIX Commands

The following table contains a list of commonly used UNIX commands. For more information, see the manual pages documentation for your operating system.

Command	Description	Example	
cd	change working directory - This command is very similar to the cd command from MSDOS.	<code>cd /opt/WebSphere/AppServer</code>	change to the default WebSphere installation directory
		<code>cd ..</code>	change to the parent of the current directory
chmod	change the permission mode of a file -	<code>chmod 600 filename</code> OR <code>chmod u=rw</code>	give the current user read and write permissions to "filename", and deny access to all other users
clear	clear your terminal screen	<code>clear</code>	clear your terminal screen
cp	copy files	<code>cp /usr/Big3/Big3_deployable.jar Big3.jar</code>	copy the Big3_deployable jar file to the current directory and rename it Big3.jar
		<code>cp /usr/Big3/html/* .</code>	copy all of the files in the html directory to the current directory
df	report free disk space on the file system	<code>df</code>	report free disk space mod 512 bytes
		<code>df -k</code>	report free disk space in kilobytes
du	display the number of disk blocks used per directory or file	<code>du</code>	display the number of disk blocks used per directory or file

Appendix B. General Unix Help

Command	Description	Example	
echo	echo arguments to standard output	echo \$SHELL	echo the value of the SHELL system variable to the screen
find	find files by name, or by other characteristics	find / -name WebSphere*	finds all files starting with "WebSphere"
grep	search a file for a string	grep e-business tracefile	displays all of the lines from tracefile which contain the string "e-business"
kill	send a signal to a process, or end a process	kill -9 2138	force process 2138 to exit
ls	list the contents of a directory - very similar to the MSDOS dir command	ls	list the files in the current directory
		ls -l	list the files in the current directory along with attributes of each file
		ls -a	list the files in the current directory, including hidden files
man	display reference manual pages (help)	man ls	displays the manual page for the ls command
		man -k keyword	display the list of manual pages containing the keyword (must be correctly configured on your system)
mkdir	make a directory	mkdir Big3	create a Big3 directory in the current directory
more	page through a text file	more tracefile	display the tracefile page by page
mv	move or rename files	mv tracefile ./save/tracefile	move the tracefile to the save directory within the current directory
		mv tracefile tracefile.old	rename the tracefile to tracefile.old
ps	display the status of current processes	ps -ef	display a full listing of all running processes
		ps -ef grep java	display a full listing of all running processes containing the string "java"
pwd	display the pathname of the current directory	pwd	display the path name of the current directory
rm	remove files or directories	rm index.html	remove the index.html file in the current directory -- WARNING: there is no verification prompt for this command, be careful when using wild cards
tail	display the last part of a file	tail tracefile	displays the last ten lines of the tracefile

Appendix B. General Unix Help

Command	Description	Example	
		tail -f tracefile	displays the last ten lines of the tracefile on a continuous one second refresh (can be used to monitor the tracefile)
which	locate commands in your search path, display its pathname or alias	which java	displays the path or alias for the "java" command
whoami or who am i	display the current user name	whoami or who am i	display the current user name

For more information, view the UNIX help for users Webpage at <http://cc.uoregon.edu/unixhelp/index.html>.

Appendix C. Programmatic Login Examples

Login URL file: login.html

```
<html>
<head>
<title>Custom Login Sample</title>
</head>

<body>
<strong>WAS 3.5 Std/Adv</strong>
<h2>Custom Login Sample</h2>
<FORM METHOD=POST ACTION="/servlet/CustomLoginServlet">
<p align="left">
<font size="2">
<em><strong>Please Enter:</strong></em><br>
</font>
</p>

<strong>UserName</strong><input type="text" size="20" name="userid">
<br>
<strong>Password</strong><input type="password" size="20" name="password">
<br><br>

<em><strong> Jump To:</strong></em>
<select name="jumpto" size="1">
<option selected>http://ratnam.raleigh.ibm.com/servlet/snoop</option>
<option>http://ratnam.raleigh.ibm.com/servlet/hello</option>
</select>

<p align="center"><input type="submit" name="login" value="Login">
</p>
</form>
</body>
</html>
```

Appendix C. Programmatic Login Examples

Servlets that performs login and exploits Single Sign-On

AbstractLoginServlet

```
// IBM Confidential OCO Source Material
// 5648-C83, 5648-C84 (C) COPYRIGHT IBM 1997,1998,1999
// The source code for this program is not published or otherwise divested
// of its trade secrets, irrespective of what has been deposited with the U.S. Copyright Office.

import java.rmi.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import com.ibm.websphere.security.SSOAuthenticator;
/**

 */
public abstract class AbstractLoginServlet extends HttpServlet{

    private String userid;
    private String password;
    private String redirectURL;
    private ServerSideAuthenticator authenticator;

    public void init() {
        authenticator = new ServerSideAuthenticator();
    }

    protected void setUserData(String uid, String pwd) {
        userid = uid;
        password = pwd;
    }

    protected void setRedirectURL(String url) {
        redirectURL = url;
    }

    /*

    // Based on the userid and password, authenticate the user. If setSSO is set to true, then
    // set up appropriate SSO cookie on the header so that Websphere will recognize it when
    // secure resources are accessed.
```

Appendix C. Programmatic Login Examples

```
*/  
  
public Object login(HttpServletRequest req, HttpServletResponse res, boolean setSSO)  
throws ServletException {  
  
    PrintWriter out =null;  
    try {  
        out = res.getWriter();  
    } catch (java.io.IOException e){  
    }  
  
    boolean success = false;  
    org.omg.SecurityLevel2.Credentials retCreds =null;  
    try {  
        retCreds = authenticator.login(userid, password, true);  
    } catch(Exception e) {  
        // catch all possible exceptions if you want or handle them separately  
        throw new ServletException();  
    }  
  
    // Set up the SSO cookie so that WebSphere can use that instead of prompting for userid  
    // and password, or in the case of Custom challenge type will not redirect the user back to  
    // the login form.  
    if (setSSO) {  
        try {  
            setupSSO(userid, password, req, res);  
        } catch (RemoteException e) {  
            throw new ServletException("Error setting single sign-on");  
        } catch (org.omg.SecurityLevel2.LoginFailed e) {  
            throw new ServletException("Login Failed");  
        }  
    }  
  
    // do whatever is required of postlogin  
    postLogin(req, res);  
  
    // redirect may fail if setupSSO failed  
  
    try {  
        res.sendRedirect(redirectURL);  
    } catch (java.io.IOException e) {
```

Appendix C. Programmatic Login Examples

```
        throw new ServletException("Error redirecting to URL: "+redirectURL);
    }

    return retCreds;

}

public abstract void postLogin(HttpServletRequest req, HttpServletResponse res)
throws ServletException;

public void logout(HttpServletRequest req, HttpServletResponse res, Object retCreds)
throws ServletException {

    try {
        authenticator.setInvocationCredentials(null);
        unsetSSO(req, res);

    } catch (Exception e) {
        throw new ServletException(e);
    }

}

/*
    Sets up the SSO cookie on the request header
*/
private void setupSSO(String userid, String password,
    HttpServletRequest req, HttpServletResponse
res)
throws java.rmi.RemoteException,
    org.omg.SecurityLevel2.LoginFailed {

    SSOAuthenticator ssoAuth = new SSOAuthenticator();

    ssoAuth.login(userid, password, req, res);

}

/*
    Sets up the SSO cookie on the request header
*/
private void unsetSSO(HttpServletRequest req, HttpServletResponse res)
```

Appendix C. Programmatic Login Examples

```
throws java.rmi.RemoteException {  
  
    SSOAuthenticator ssoAuth = new SSOAuthenticator();  
  
    ssoAuth.logout(req, res);  
}  
  
}
```

Appendix C. Programmatic Login Examples

CustomLoginServlet

```
// IBM Confidential OCO Source Material
// 5648-C83, 5648-C84 (C) COPYRIGHT IBMachines 1997,1998,1999
// The source code for this program is not published or otherwise divested
// of its trade secrets, irrespective of what has been deposited with the U.S. Copyright Office.
```

```
import java.io.*;
import javax.servlet.http.*;
import javax.servlet.*;
import com.ibm.ejs.security.util.Base64Coder;
import com.ibm.ejs.security.*;
import java.rmi.*;
import javax.naming.InitialContext;

/**
 * CustomLoginServlet
 *
 * Process a custom HTML 'login' form (HTML form attached at the bottom of this class) and
 performs a login using the login() method found within the the base class (LoginServlet).
 *
 * This sample servlet is used as an example of the Custom Login feature of
 * WebSphere S/A 3.5.
 *
 */

public class CustomLoginServlet extends AbstractLoginServlet {

    // Strings
    private String defaultRedirectUrl = null;

    /**
     * init
     *
     * Initializes the servlet. Reads and processes parameters to servlet
     *
     */
    public void init(ServletConfig conf) throws ServletException {

        super.init(conf);

        // read the default redirect URL for this servlet (...if it exists).
```

Appendix C. Programmatic Login Examples

```
defaultRedirectUrl = getInitParameter ("DefaultRedirectURL");
}
```

doPost Method

```
/**
 * doPost
 *
 * The doPost method is the primary entry point of this servlet. It is designed to be called as the
 * result of an HTML form post. This servlet will read and validate the posted parameters, then* call
 * the 'login' API found in the base class (LoginServlet)
 *
 * @param req the request object
 * @param res the response object
 * @return <code>void</code>
 * @exception IOException problem reading from or writing to
 *         the req or res streams
 */
public void doPost (HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException {

    String userId=null;        // user id found on form
    String userPw=null;        // password
    String redirectURL=null;    // redirect url

    userId =req.getParameter("userid"); // obtain userid data from form
    userPw =req.getParameter("password"); // obtain password from form
    redirectURL =req.getParameter("jumpto"); // obtain redirect url from form

    if (defaultRedirectUrl!=null) { // override redirect url with default
        redirectURL = defaultRedirectUrl;
    }

    int err; // validate parameters
    if ( (err=checkParams( userId, userPw,
        redirectURL))!=0) {
        showInvalidParameterPage(err, res); // display error page
        return;
    }

    // Set login data
    setUserData(userId, userPw);
}
```

Appendix C. Programmatic Login Examples

```
setRedirectURL(redirectURL);

// perform the log on
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
boolean success = false;

    boolean setSSO = true;

    try {
login( req, res, setSSO );
        success = true;
    } catch (ServletException se) {
        out.println("CustomLoginServlet: ERROR logging on"+se);
    }
}
```

Appendix C. Programmatic Login Examples

postLogin Method

```
/**
 * postLogin
 *
 * The postLogin method is called after a login has successfully occurred. This method is a good *
 place to establish a HTTP session or other actions that relate to the Logged on user.
 * This code is now running under the identity of the user
 *
 * @param req the request object
 * @param res the response object
 * @return <code>void</void>
 * @exception IOException problem reading from or writing to
 *         the req or res streams
 */
public void postLogin( HttpServletRequest req, HttpServletResponse res)
throws ServletException {

}
```

Appendix C. Programmatic Login Examples

checkParams

```
/**
 * checkParams
 *
 * Check the values of the parameters read from the HTML login form
 *
 * @param  userId the userid
 * @param  userPw the Password
 * @param  redirectURL the url to jump to a successful after login
 * @return <code>int</code> error 1=userid, 2=password, 3=other
 */
private int checkParams( String userId, String userPw, String redirectURL) {
    int err=0; // zero mean no error

    // validate userid
    if (!(userId != null && (userId.length() != 0))) {
        // Invalid userID (empty string)
        err=1;
        return err;
    }

    // validate password
    if (!(userPw != null && (userPw.length() != 0))) {
        // Invalid Password (empty string)
        err=2;
        return err;
    }

    // validate redirect URL
    if (!(redirectURL != null && (redirectURL.length() != 0))) {
        // Invalid redirect URL (empty string)
        err=3;
        return err;
    }
    return err;
}
```

Appendix C. Programmatic Login Examples

showInvalidParameterPage

```
/**
 * showInvalidParameterPage
 *
 * Displays an error page...
 *
 * @param err the type of error encountered 1=bad userid, 2=bad password
 * @param res the response object
 * @return <code>void</void>
 * @exception IOException problem reading from or writing to
 *         the req or res streams
 */
private void showInvalidParameterPage( int err, HttpServletResponse res)
throws ServletException, IOException {

    ServletOutputStream out = res.getOutputStream();

    out.println("<html>");
    out.println("<head><title>Invalid Login Parameter</title></head>");
    out.println("<body>");
    out.println("<h1>Invalid Login Parameter</h1>");
    out.print("<p><strong>");
    if (err==1) {
        out.print("Invalid UserID");
    } else if (err==2) {
        out.print("Invalid UserID");
    } else {
        out.print("Invalid redirect URL");
    }
    out.println("</strong>");
    out.println("</body></html>");
    out.close();
}

public String getServletInfo(){
    return "CustomLoginServlet";
}

}
```

Appendix C. Programmatic Login Examples

Client Side Login

You can obtain the SecurityCurrent object from the underlying Object Request Broker (ORB). Obtain the ORB and resolve the name as illustrated in the following code segment:

```
// obtain a reference to the underlying ORB
com.ibm.CORBA.iiop.ORB orb = com.ibm.ejs.oa.EJSORB.getORBInstance();

// obtain the SecurityCurrent from the ORB
CurrentImpl current = orb.resolve_initial_references("SecurityCurrent");
```

You can set authentication data on the client's security context by performing a request_login on the Common Object Request Broker Architecture (CORBA) LoginHelper object.

```
com.ibm.IExtendedSecurity._LoginHelper loginHelper = current.login_helper();
loginHelper.request_login(userid, "", password,
    new org.omg.SecurityLevel2.CredentialsHolder(),
    new org.omg.Security.OpaqueHolder());
```

When a server receives the method request, the authentication data is extracted and authentication will take place.

WebSphere 3.5 ships with an example to perform client side login. You can use the LoginHelper class to achieve the same. Initialize the client application with the ORB security properties by setting the value of **com.ibm.CORBA.configURL** to the file that contains the properties. The **sas.client.props** file under the properties directory contains valid values. For example, if the client application is called TestClient, then launch the application as follows:

```
java -Dcom.ibm.CORBA.configURL=file:\\E:\WebSphere\AppServer\properties\sas.client.props
TestClient
```

Appendix C. Programmatic Login Examples

TestClient

This TestClient can use the LoginHelper class to achieve client login. The following set of steps illustrate how the LoginHelper class can be used by TestClient to perform a login:

```
public class TestClient {  
  
    :  
    private void performLogin() {  
        // get userid and password  
        String userid = customGetUserid();  
        String password = customGetPassword();  
  
        LoginHelper loginHelper = new LoginHelper();  
        try {  
            // set up the client authentication data so that  
            // a server will extract this userid and password and perform authentication  
            Org.omg.SecurityLevel2.Credentials credentials =  
                loginHelper.login(userid, password);  
            String username = loginHelper.getUserName(credentials);  
            System.out.println("Security context set up for user: "+ username);  
  
        } catch (org.omg.SecurityLevel2.LoginFailed e) {  
            // handle login failed exception  
        }  
    }  
  
}
```

The second category of login in the server side login functionality. A server side login pattern can be followed when a programmatic login is required not only to imply setting up the security context, but also to perform actual authentication. CORBA security APIs provide the framework for programmatically providing authentication data - similar to the client side login. The framework is extended so that authentication is forced to occur when required.

The credentials obtained from the request_login method invocation can be used for authentication by performing a **get_mapped_credentials** method invocation on the credential object.

```
// getting the mapped credentials forces authentication to take place  
if (credentials instanceof com.ibm.ISecurityLocalObjectBasicAuthImpl.CredentialsImpl) {  
    mapCreds = ((com.ibm.ISecurityLocalObjectBasicAuthImpl.CredentialsImpl)  
                credentials).get_mapped_credentials(null, "", null);  
}
```

Appendix C. Programmatic Login Examples

}

Appendix C. Programmatic Login Examples

ServerSideAuthenticator

The `ServerSideAuthenticator` class is a helper class available with WebSphere 3.5 that can be used to perform authentication on the server side.

```
public class TestServer {  
  
    :  
    private void performLoginAndAuthentication() {  
        // get userid and password  
        String userid = customGetUserid();  
        String password = customGetPassword();  
  
        // make sure authentication takes place immediately  
        boolean forceAuthentication = true;  
  
        ServerSideAuthenticator serverAuth = new ServerSideAuthenticator();  
        try {  
  
            // perform authentication based on the authentication data  
  
            org.omg.SecurityLevel2.Credentials credentials =  
                serverAuth.login(userid, password, forceAuthentication);  
  
            String username = serverAuth.getUserName(credentials);  
  
            System.out.println("Authentication successful for the user: "+ username);  
  
        } catch (Exception e) {  
            // handle exception...  
        }  
    }  
}
```

Appendix C. Programmatic Login Examples

LoginHelper

```
// IBM Confidential OCO Source Material
// 5648-C83, 5648-C84 (C) COPYRIGHT IBM 1997,1998,1999
// The source code for this program is not published or otherwise divested
// of its trade secrets, irrespective of what has been deposited with the U.S. Copyright Office.
```

```
import java.io.*;
import com.ibm.ISecurityLocalObjectBaseL13Impl.*;
import org.omg.Security.*;
import org.omg.SecurityLevel2.Credentials;
import com.ibm.IExtendedSecurity.*;
```

```
/**
```

LoginHelper is a utility class that uses CORBA security APIs to perform a programmatic login. It uses the CORBA "SecurityCurrent" to obtain a CORBA login_helper and invokes request_login on that object. This method sets up the userid and password on the thread so that any further method invocation on a secure server will be invoked under that userid/password. The server will extract the userid and password and will perform authentication before invoking the method.

```
*/
```

```
public class LoginHelper {

    protected static final int PUBLIC = 0;

    protected static org.omg.Security.AttributeType secAttrType[];

    // use the CORBA attribute type to get public attribute (user name),
    // access id (user registry specific unique ID)

    static {
        secAttrType = new org.omg.Security.AttributeType[1];

        org.omg.Security.ExtensibleFamily familyOMG =
            new org.omg.Security.ExtensibleFamily((short)0, (short)1);

        secAttrType[PUBLIC] = new org.omg.Security.AttributeType(familyOMG,

                                                                    org.omg.Security.Public.value);
    }
}
```

Appendix C. Programmatic Login Examples

```
// IBM's implementation of SecurityCurrent
private com.ibm.ISecurityLocalObjectBaseL13Impl.CurrentImpl current = null;

public LoginHelper() throws IllegalStateException {
    // obtain the security current from the ORB instance
    current = getSecurityCurrent();
}

/**
    The method login() sets the authentication data on the thread of execution
    by calling the login_helper.request_login

        @param userid userid
        @param password user's password
        @return Credentials set on the security context based on the userid
            and password
        @throws LoginFailed exception
    */

public org.omg.SecurityLevel2.Credentials login(String userid, String password)
throws org.omg.SecurityLevel2.LoginFailed

{

    com.ibm.IExtendedSecurity._LoginHelper loginHelper = current.login_helper();
    return loginHelper.request_login(userid, "", password,
        new org.omg.SecurityLevel2.CredentialsHolder(),
        new org.omg.Security.OpaqueHolder());
}

/**

    This method sets the specified credentials as the invocation credentials. Any
    method invoked after this will be executed under the authority of the credentials.

        @param invokedCreds credentials under which future methods should be invoked
        @throws InvalidCredentialType, InvalidCredential
    */

public void setInvocationCredentials(org.omg.SecurityLevel2.Credentials invokedCreds)
throws org.omg.Security.InvalidCredentialType,
```

Appendix C. Programmatic Login Examples

```
    org.omg.SecurityLevel2.InvalidCredential
{

    current.set_credentials(org.omg.Security.CredentialType.SecInvocationCredentials,
                           invokedCreds);

}

/**

    This method gets the credentials under which the current
    method is getting executed.

    @return credentials under which the current method is invoked

    @throws InvalidCredentialType

*/
public org.omg.SecurityLevel2.Credentials getInvocationCredentials()
throws org.omg.Security.InvalidCredentialType
{
return current.get_credentials(org.omg.Security.CredentialType.SecInvocationCredentials,
                              false, false, null);
}

/**

    Returns a human-readable user name attribute of the specified credentials.

    @return String the user name

*/
public String getUsername(org.omg.SecurityLevel2.Credentials creds)
throws org.omg.Security.DuplicateAttributeType,
       org.omg.Security.InvalidAttributeType
{

    org.omg.Security.Attribute[] publicAttrs =
creds.get_attributes(secAttrType);
    String uname = "";
    try {
        uname = new String(publicAttrs[PUBLIC].value, "UTF8");
    } catch (java.io.UnsupportedEncodingException e) {
```

Appendix C. Programmatic Login Examples

```
    }
    return uname;
}

private com.ibm.ISecurityLocalObjectBaseL13Impl.CurrentImpl getSecurityCurrent()
    throws IllegalStateException
{
    com.ibm.ISecurityLocalObjectBaseL13Impl.CurrentImpl current =null;

    try {
        // obtain a reference to the underlying ORB
        com.ibm.CORBA.iiop.ORB orb =
com.ibm.ejs.oa.EJSORB.getORBInstance();

        // obtain the SecurityCurrent from the ORB
        if (orb != null) {
            current=(com.ibm.ISecurityLocalObjectBaseL13Impl.CurrentImpl)
                orb.resolve_initial_references("SecurityCurrent");
        } else {
            throw new IllegalStateException("SecurityCurrent: null");
        }
    } catch (Exception e) {
        throw new IllegalStateException("Error getting
SecurityCurrent from the ORB");
    }
    return current;
}
}
```

Appendix C. Programmatic Login Examples

Form-Based Login

A sample HTML file that uses the CustomLoginServlet is given below. If this file is named login.html, then the URL that points to this file should be specified in the LoginURL and ReLoginURL fields in the configuration panel.

```
<FORM METHOD=POST ACTION="/servlet/CustomLoginServlet"><p align="left"><font  
size="2"><em><strong>Please Enter:</strong></em><br></font></p>
```

```
<strong>UserName</strong><input type="text" size="20" name="userid"><br>  
<strong>Password</strong><input type="password" size="20" name="password"><br><br>
```

:

```
<p align="center"><input type="submit" name="login" value="Login"></p>
```

The CustomLoginServlet extracts the userid, password, and the redirect URL information from the HTML form. It then invokes the necessary methods on its parent AbstractLoginServlet.

```
// Set login data  
setUserData(userId, userPw);  
setRedirectURL(redirectURL);  
  
// perform the log on  
boolean success = false;  
  
// After successful authentication, set up the single sign-on cookie so that  
// the user is not required to enter the userid and password again  
  
boolean setSSO = true;  
  
try {  
    // invoke the login method on the AbstractLoginServlet class  
    login( req, res, setSSO );  
    success = true;  
} catch (ServletException se) {  
    // handle the error: "CustomLoginServlet: ERROR logging on"  
}
```

Appendix C. Programmatic Login Examples

SSOAuthenticator Object

The SSOAuthenticator object can be constructed. The servlet can then invoke the login method on the SSOAuthenticator by providing the userid, password, and the HTTP request and response objects.

```
public org.omg.SecurityLevel2.Credentials login(String userid, String password,
                                              HttpServletRequest req, HttpServletResponse res)
    throws org.omg.SecurityLevel2.LoginFailed;
```

The login method will authenticate the user based on the userid and password values. If authentication is successful, it will create a SSO cookie based on the SSO configuration and will set the cookie on the response header. If authentication fails, it will throw an **org.omg.SecurityLevel2.LoginFailed** exception. This method returns the credentials based on successful login.

The AbstractLoginHelper uses the SSOAuthenticator to set up the SSO context.

```
/*
    Sets up the SSO cookie on the request header
*/
private void setupSSO(String userid, String password, HttpServletRequest req,
HttpServletResponse res)
    throws java.rmi.RemoteException, org.omg.SecurityLevel2.LoginFailed {

    SSOAuthenticator ssoAuth = new SSOAuthenticator();
    ssoAuth.login(userid, password, req, res);
}
```

All the subsequent requests to the WebSphere will include the SSO cookie in the request header. WAS will base its authentication and authorization decisions on the information that is contained in the SSO cookie. Once a user has ended the session, the user might want to logout so that anyone that is using the browser at a later time will not be authorized to same the resources the user. The logout functionality of the SSOAuthenticator can be used to remove the SSO context information.

```
/*
    Sets up the SSO cookie on the request header
*/
private void unsetSSO(HttpServletRequest req, HttpServletResponse res)
    throws java.rmi.RemoteException {

    SSOAuthenticator ssoAuth = new SSOAuthenticator();
    ssoAuth.logout(req, res);
}
```

Appendix C. Programmatic Login Examples

Server Side Login

ServerSideAuthenticator

```
// IBM Confidential OCO Source Material
// 5648-C83, 5648-C84 (C) COPYRIGHT IBM 1997,1998,1999
// The source code for this program is not published or otherwise divested
// of its trade secrets, irrespective of what has been deposited with the U.S. Copyright Office.
```

```
import java.io.*;
import com.ibm.ISecurityLocalObjectBaseL13Impl.*;
import org.omg.Security.*;
import org.omg.SecurityLevel2.Credentials;
import com.ibm.IExtendedSecurity.*;
```

```
/**
```

ServerSideAuthenticator is a utility class that uses CORBA security APIs to perform a programmatic login. It uses the CORBA "SecurityCurrent" to obtain a CORBA login_helper and invokes request_login on that object. This method sets up the userid and password on the thread. It optionally forces authentication to take place. If the authentication is successful, any further method invocation will be invoked under that authenticated user identity.

```
*/
```

```
public class ServerSideAuthenticator extends LoginHelper {
```

```
    public ServerSideAuthenticator() throws IllegalStateException {
        super();
    }
```

```
    /**
```

```
        The method login()
```

```
        * sets the authentication data on the thread of execution
        by calling the login_helper.request_login
```

```
        * forces authentication to take place if force_authn is set to true
```

```
    */
```

```
    public org.omg.SecurityLevel2.Credentials login(String userid, String password,
                                                    boolean force_authn)
        throws org.omg.SecurityLevel2.LoginFailed,
```

Appendix C. Programmatic Login Examples

```
com.ibm.IExtendedSecurity.RealmNotRegistered,  
com.ibm.IExtendedSecurity.UnknownMapping,  
com.ibm.IExtendedSecurity.MechanismTypeNotRegistered,  
com.ibm.IExtendedSecurity.InvalidAdditionalCriteria  
{  
  
    Credentials creds = super.login(userid, password);  
    Credentials mapCreds = creds;  
  
    if (force_authn) {  
  
        // getting the mapped credentials forces authentication to take place  
        if (creds instanceof com.ibm.ISecurityLocalObjectBasicAuthImpl.CredentialsImpl) {  
            mapCreds = ((com.ibm.ISecurityLocalObjectBasicAuthImpl.CredentialsImpl)  
                creds).get_mapped_credentials(null, "", null);  
        }  
  
    }  
    return mapCreds;  
}
```

```
/**
```

The authenticate method performs authentication by using the specified userid and password. The security context is not necessarily affected by the authentication.

```
*/
```

```
public org.omg.SecurityLevel2.Credentials authenticate(String userid, String password)  
throws org.omg.Security.InvalidCredentialType,  
    org.omg.SecurityLevel2.LoginFailed,  
    org.omg.SecurityLevel2.InvalidCredential,  
    com.ibm.IExtendedSecurity.RealmNotRegistered,  
    com.ibm.IExtendedSecurity.UnknownMapping,  
    com.ibm.IExtendedSecurity.MechanismTypeNotRegistered,  
    com.ibm.IExtendedSecurity.InvalidAdditionalCriteria  
{  
  
    Credentials invocCreds = getInvocationCredentials();  
    Credentials authCreds = login(userid, password, true);  
    setInvocationCredentials(invocCreds);
```

Appendix C. Programmatic Login Examples

```
        return authCreds;
    }
}
```

Appendix C. Programmatic Login Examples

Obtaining Identity Information

The information about the identity initiating a method on a secure servlet or as secure EJB can be programmatically obtained by using the mechanisms that are specified in the servlet and EJB specifications.

From a Servlet: The `HttpServletRequest` object is passed to the service method of the servlet so that it can be used to obtain the information on the identity making the method initiation. The `getRemoteUser` method on the `HttpServletRequest` can be used to get the name of the identity. If the user is authenticated, it will return the user name; whereas, if the user is unauthenticated, it will return the key word "anonymous."

```
public void doGet(HttpServletRequest req, HttpServletResponse res) {  
    // obtain the user name  
    String user = req.getRemoteUser();  
}
```

From an EJB: The `EJBContext` can be used to obtain the information about the identity making the EJB method initiation. The `getCallerIdentity` method can be used to obtain the information.

```
public void myGetUserName(EJBContext context) {  
    return context.getCallerIdentity().getName();  
}
```

Limitations

If the authentication mechanism is Native Operating System and if the operating system is Windows NT:

- The user ID that is associated with WebSphere should be different than the machine name. For example, if the machine name is "bob" then the server ID should not be "bob".
- If the machine is part of a NT domain, then the user ID that is associated with WebSphere should be a valid ID in that domain.

Appendix D. Transaction Terms

The definitions come from the Object Management Group (OMG) Office of Technical Services (OTS) V1.1 Specification.

2PC: See Two-phase commit.

Abort: See Rollback

Active: The state of a transaction when processing is in progress and completion of the transaction has not yet commenced.

Atomicity: A transaction property that ensures that if work is interrupted by failure, any partially completed results will be undone. A transaction whose work completes is said to commit. A transaction whose work is completely undone is said to rollback (abort).

Begin: An operation on the Transaction Service which establishes the initial boundary of a transaction.

Commit: Commit has two definitions as follows: An operation in the Current and Terminator interfaces that a program uses to request that the current transaction end normally and that the effects of that transaction be made permanent. An operation in the Resource interface which causes the effects a transaction to be made permanent.

Commit Coordinator: In a two-phase commit protocol, the program that collects the vote from the participants.

Commit Participant: In a two-phase commit protocol, the program that returns a vote on the completion of a transaction.

Committed: The property of a transaction or a transactional object, when it has successfully performed the commit protocol. See also in-doubt, active, and rolled back.

Completion: The processing required (either by commit or rollback) to obtain the durable outcome of a transaction.

Coordinator: A coordinator involves Resource objects in a transaction when they are registered. A coordinator is responsible for driving the two-phase commit protocol. See also Commit coordinator and Commit participant.

Consistency: A property of a transaction that ensures that the transaction's actions, taken as a group, do not violate any of the integrity constraints associated with the state of its associated objects. This requires that the application program be implemented correctly: the Transaction Service provides the functionality to support application data consistency.

Appendix D. Transaction Terms

Decided Commit State: A root coordinator enters the decided commit state when it has written a log-commit record; a subordinate coordinator or resource is in the decided commit state when it has received the commit instruction from its superior; in the latter case, a log-commit record may be written but this is not essential.

Decided Rollback State: A coordinator or resource enters the decided rollback state when it decides to rollback the transaction or has received a signal to do so.

Direct Context Management: An application manipulates the Control object and the other objects that are associated with the transaction. See also Indirect context management

Durability: A transaction property that ensures the results of a successfully completed transaction will never be lost, except in the event of catastrophe. It is generally implemented by a combination of persistent storage and a logging service that provides a backup copy of permanent changes.

Execution Environment: An implementation-dependent factor that may determine the outcome of certain operations on the Transaction Service. Typically the execution environment is the scope within which shared state is managed.

Flat Transaction: A transaction that has no subtransactions - and that cannot have subtransactions.

Forgotten "State": This is not really a transaction state, because there is no memory of the transaction: it has either completed or rolled back and all records on permanent storage have been deleted.

Heuristic Commit or Rollback: To unilaterally make the commit or rollback decision about in-doubt transactions when the coordinator fails or contact with the coordinator fails.

Indirect Context Management: An application that uses the Current object, that is provided by the Transaction Service, to associate the transaction context with the application thread of control. See also Direct context management.

In-Doubt: The state of a transaction if it is controlled by a transaction manager that can not be contacted, so the commit decision is in doubt. See also active, committed, rolled back.

Interposition: Adding a sequence of one or more subordinate coordinators between a root coordinator and its participants.

Appendix D. Transaction Terms

Isolation: A transaction property that allows concurrent running of the transaction, but the results will be the same as if the transaction was serialized. Isolation ensures that concurrently running transactions cannot observe inconsistencies in shared data.

Lock Service: Called the Concurrency Control Service, it is an Object Service that is used by resources to control access to shared objects by concurrently running methods.

Log-Ready Record (and contents): For an intermediate coordinator, a log-ready record contains the identification of the (superior) coordinator and that of Resource objects (this includes the subordinate coordinators) that are registered with the coordinator which replied VoteCommit (that is, it excludes registered objects which replied VoteReadOnly). For a Resource object, a log-ready record includes the identification of the coordinator with which it is registered.

Log-commit record (and contents): A log-commit record contains the identification of all registered Resource objects which replied VoteCommit.

Log-Heuristic Record: This contains a record of a heuristic decision; either HeuristicCommit or HeuristicRollback.

Log-Damage Record: This contains a record of heuristic damage. (Where it is known that a heuristic decision conflicted with the decided outcome (HeuristicMixed), or where there is a risk that a heuristic decision conflicted with the decided outcome (HeuristicHazard).)

Log Service: A service used by resource managers for recording recovery information and the Transaction Service for durably recording the transaction state.

Nested Transaction: A transaction that either has subtransaction or is a subtransaction on some other transaction.

Participant: See Commit participant.

Persistent Storage: Generally speaking, a synonym for Stable storage. The Persistent Object Service (POS) provides an object representation of stable storage.

Prepared: The state that a transaction is in when phase one of a two-phase commit has completed.

Presumed Rollback: An optimization of the two-phase commit protocol that results in more efficient performance as the root coordinator does not need to log anything before the commit decision and the Participants (Resource objects) do not need to log anything before they prepare. So called because, at restart, if no record of the transaction is found, it is safe to assume that the transaction rolled back.

Appendix D. Transaction Terms

Propagation: A function of the Transaction Service that allows the Transaction context of a client to be associated with a transactional operation on a server object. The Transaction Service supports both implicit and explicit propagation of transaction context.

Recoverable Object: An object whose data is affected by committing or rolling back a transaction.

Recoverable Server: A transactional object with recoverable state that registers a Resource (not necessarily itself) with a Coordinator to participate in transaction completion.

Recovery Service: A service that is used by resource managers for restoring the state of objects to a prior state of consistency.

Resource: An object in the Transaction Service that is registered for involvement in two-phase commit - 2PC. Corresponds to a Resource Manager.

Resource Manager: A component which manages the integrity of the state of a set of related resources.

Rollback: Rollback (also known as Abort) has two definitions, as follows: An operation in the Current and Terminator interfaces that is used to indicate that the current transaction has ended abnormally and its effects should be discarded. An operation in the Resource interface which causes all the state changes in the transaction to be undone.

Rolled Back: The property of a transaction or a transactional object when it discards all changes that were made in the current transaction. See also in-doubt, active, and committed.

Root Coordinator: The first coordinator in a sequence of coordinators where there is interposition. The coordinator that is associated with the transaction originator.

Security Service: An object service which provides the identifications of users (authentication), controls access to the resources (authorization), and provides auditing of resource access.

Stable Storage: Storage that is not likely to be damaged as the result of node failure.

Sub-Coordinator: See Subordinate Coordinator.

Subordinate Coordinator: A coordinator that is subordinate to the root coordinator when interposition has been performed. A subordinate coordinator appears as a Resource object to its superior. Also known as a Sub-coordinator.

Synchronization: An object in the Transaction Service which controls the transfer of the persistent object state data so that it can be made durable by its associated resource.

Appendix D. Transaction Terms

Thread: The entity that is currently in control of the processor.

Thread Service: A service which enables methods to run concurrently by the same process. Where two or more methods can run concurrently, each method is associated with its own thread of control.

TP Monitor: A system component that accepts input work requests and associates resources with the programs that act on these requests to provide a run-time environment for program completion.

Transaction: A collection of operations that are on the physical and abstract application state.

Transactional Client: An arbitrary program that can invoke operations of many transactional objects in a single transaction. Not necessarily the Transaction originator.

Transaction Context: The transaction information that is associated with a specific thread. See Propagation.

Transactional Operation: An operation on an object that participates in the propagation of the current Transaction.

Transaction Originator: An arbitrary program - typically, a transactional client, but not necessarily an object - that begins a transaction.

Transaction Manager: A system component that implements the protocol engine for 2-phase commit protocol. See also Transaction Service.

Transactional Object: An object whose operations are affected by being started within the scope of a transaction.

Transactional Server: A collection of one or more objects whose behavior is affected by the transaction, but has no recoverable state of its own.

Transaction Service: An Object Service that implements the protocols required to guarantee the ACID (Atomicity, Consistency, Isolation, and Durability) properties of transactions. See also Transaction Manager.

TSPortability: An interface of the Transaction Service which allows it to track transactional operations and propagate transaction context to another Transaction Service implementation.

Appendix D. Transaction Terms

Two-Phase Commit: A transaction manager protocol for ensuring that all changes to recoverable resources occur atomically. Furthermore, the failure of any resource to complete will cause all other resource to undo the changes. Also called 2PC.

Appendix E. Database Hints

Configuring local DB2 as the administrative server repository

When you install WebSphere Application Server, the installation program automatically creates the DB2 database using the createdb2.bat file on Windows NT. On Solaris and AIX, to create the DB2 database, run the createdb2.sh file that was installed in the `<asroot>/bin` directory.

OR:

Execute the following from a db2 command window:

```
CREATE DATABASE was
UPDATE DB CFG FOR was USING APPLHEAPSZ 256;
```

If you have a previous instance of the was database, be sure to drop it first by executing the following from a db2 command window:

```
DROP DATABASE was
```

Catalog NODE and ALIAS to remote DB2 server

Suppose you are on maxilla and want to create an alias "samsac" on maxilla that points to the "sample" database on machine sacrum **and** use the fast ethernet for the connection:

1. Create a node on maxilla (lets call it nsacrum) which points to sacrum:

```
DB2 CATALOG TCPIP NODE nsacrum REMOTE e-sacrum SERVER db2cdb2inst1
```

Where

- sacrum is the name of the node being created.
- e-sacrum is what you can ping from maxilla to go over the fast ethernet to sacrum.
- db2cdb2inst1 is the "db2c..." name in `/etc/services` on maxilla.

2. Now create the alias "samsac" on maxilla that points to the "sample" database:

```
DB2 CATALOG DATABASE sample AS samsac AT NODE nsacrum
```

Where

- sample is the actual database name on sacrum
- samsac is the new alias name being created on maxilla that can be used on maxilla to get to (sample on sacrum)
- nsacrum is the node on maxilla which points to sacrum over the fast ethernet

Other interesting commands:

```
db2 list node directory <-- to see defined nodes
db2 list database directory <-- to see databases and aliases
db2 uncatalog node nsacrum <-- to get rid of node nsacrum
db2 uncatalog db samsac <-- to get rid of the samsac alias
```

Appendix E. Database Hints

DB2 Backup Restore syntax

```
db2 backup database trade user db2inst1 using password to
/usr/WebSphere/AppServer/perf/archive
```

```
db2 restore database trade user db2inst1 using password from
/usr/WebSphere/AppServer/perf/archive
```

Configuring Oracle as the administrative server repository

WebSphere Application Server supports Oracle 8.1.6 but requires the latest driver manager, JDBC-Thin/100% Java for JDK1.2.2_05a. You can obtain the driver manager from the Oracle website: <http://technet.oracle.com/software/download.htm> located under driver managers for Oracle 8 and Oracle 8i, Version 8.1.6). Place the driver manager file, classes12.zip, in your /jdbc/lib directory.

Creating and configuring the database

After you obtain the driver manager, do the following to create and configure an Oracle database:

1. Create a database. For example, create a database with the following specifications:

```
Database name orcl
SID orcl
Initialization file name
<oracle_root>/database/initorc1.ora
```

2. Add the following line to the initialization file:

```
open_cursors = 200
```

On Windows NT, the initialization file is typically the `\orant\Database\Init xxx.ora` file, where xxx is your SID
On Solaris or AIX, the file typically is `<oracle_root>/dbs/init xxx.ora`, where xxx is your SID.

3. Stop the database, then restart it.

4. Define a WebSphere administration ID with database authority:

```
sqlplus system/manager
create user EJSADMIN identified by xxxxxxxxx;
grant connect, resource, dba to EJSADMIN;
quit
```

where xxxxxxxxx is the password for EJSADMIN.

5. If needed, define an ID to deploy entity beans:

```
sqlplus system/manager
create user EJB identified by EJB;
grant connect, resource, dba to EJB;
Quit
```

Appendix E. Database Hints

6. Test access to the new database using the EJSADMIN user ID.

Appendix F. XML Package Summary

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

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

```
XMLConfig -adminNodeName buccaneer -export export.xml
```

where the hostname of the machine containing the administrative server is "buccaneer" and the output will be directed to a file named export.xml.

When you perform an export, the XML output file will 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 you try the export command yourself to see exactly the output that is produced.

```
<?xml version="1.0"?>
<!DOCTYPE websphere-sa-config SYSTEM "$server_root$$dsep$bin$dsep$xmlconfig.dtd" >

<websphere-sa-config>
```

The above 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 some are omitted from the example code below.

The MIME types are followed by the URIs (also known as Web resources) hosted by default_host:

```
<virtual-host name="default_host" action="update">
  <mime-table>
    <mime type="audio/x-wav">
      <ext>wav</ext>
    </mime>
    <mime type="application/x-sv4cpio">
      <ext>sv4cpio</ext>
    </mime>
    <mime type="text/x-ssi-html">
```

Appendix F. XML Package Summary

```
<ext>htmls</ext>
<ext>shtml</ext>
</mime>
...
...
<mime type="application/x-netcdf">
  <ext>nc</ext>
  <ext>cdf</ext>
</mime>
<mime type="video/x-motion-jpeg">
  <ext>mjpg</ext>
</mime>
</mime-table>
<alias-list>
  <alias>localhost</alias>
  <alias>127.0.0.1</alias>
  <alias>buccaneer.raleigh.ibm.com</alias>
  <alias>buccaneer</alias>
  <alias>9.67.127.58</alias>
</alias-list>
<uri name="/" rootURI="/" action="create"/>
<uri name="/servlet/snoop" rootURI="/" action="create"/>
<uri name="/servlet/snoop2" rootURI="/" action="create"/>
<uri name="/servlet/hello" rootURI="/" action="create"/>
<uri name="/ErrorReporter" rootURI="/" action="create"/>
<uri name="/servlet" rootURI="/" action="create"/>
<uri name="/*.*.jsp" rootURI="/" action="create"/>
<uri name="/*.*.jsw" rootURI="/" action="create"/>
<uri name="/*.*.jsw" rootURI="/" action="create"/>
<uri name="/admin" rootURI="/admin" action="create"/>
<uri name="/admin/install" rootURI="/admin" action="create"/>
<uri name="/admin/*.jsp" rootURI="/admin" action="create"/>
<uri name="/admin/*.jsw" rootURI="/admin" action="create"/>
<uri name="/admin/*.jsw" rootURI="/admin" action="create"/>
<uri name="/admin/" rootURI="/admin" action="create"/>
<uri name="/admin/servlet" rootURI="/admin" action="create"/>
<uri name="/admin/ErrorReporter" rootURI="/admin" action="create"/>
<uri name="/webapp/examples" rootURI="/webapp/examples" action="create"/>
<uri name="/webapp/examples/simpleJSP.servlet" rootURI="/webapp/examples" action="create"/>
<uri name="/webapp/examples/simpleJSP" rootURI="/webapp/examples" action="create"/>
<uri name="/webapp/examples/ErrorServlet" rootURI="/webapp/examples" action="create"/>
<uri name="/webapp/examples/ping" rootURI="/webapp/examples" action="create"/>
```

Appendix F. XML Package Summary

```
<uri name="/webapp/examples/SourceCodeViewer" rootURI="/webapp/examples"
action="create"/>
<uri name="/webapp/examples/showCfg" rootURI="/webapp/examples" action="create"/>
<uri name="/webapp/examples/ShowCfg" rootURI="/webapp/examples" action="create"/>
<uri name="/webapp/examples/showConfig" rootURI="/webapp/examples" action="create"/>
<uri name="/webapp/examples/ShowConfig" rootURI="/webapp/examples" action="create"/>
<uri name="/webapp/examples/HitCount" rootURI="/webapp/examples" action="create"/>
<uri name="/webapp/examples/verify" rootURI="/webapp/examples" action="create"/>
<uri name="/webapp/examples/*.jsp" rootURI="/webapp/examples" action="create"/>
<uri name="/webapp/examples/*.jsw" rootURI="/webapp/examples" action="create"/>
<uri name="/webapp/examples/*.jsw" rootURI="/webapp/examples" action="create"/>
<uri name="/webapp/examples/" rootURI="/webapp/examples" action="create"/>
<uri name="/webapp/examples/HelloPervasive" rootURI="/webapp/examples" action="create"/>
<uri name="/webapp/examples/StockQuote" rootURI="/webapp/examples" action="create"/>
<uri name="/webapp/examples/BeenThere" rootURI="/webapp/examples" action="create"/>
</virtual-host>
```

This section contains the **database driver and data source information**:

```
<jdbc-driver name="Admin DB Driver" action="update">
  <implementation-class>jdbc.idbDriver</implementation-class>
  <url-prefix>jdbc:idb</url-prefix>
  <jta-enabled>>false</jta-enabled>
  <install-info>
    <node-name>buccaneer</node-name>
    <jdbc-zipfile-location>/usr/WebSphere/AppServer/lib/idb.jar</jdbc-zipfile-location>
  </install-info>
</jdbc-driver>
<jdbc-driver name="AccountDB2Driver" action="update">
  <implementation-class>com.ibm.db2.jdbc.app.DB2Driver</implementation-class>
  <url-prefix>jdbc:db2</url-prefix>
  <jta-enabled>>false</jta-enabled>
  <install-info>
    <node-name>buccaneer</node-name>
    <jdbc-zipfile-location>/home/db2as/sqllib/java/db2java.zip</jdbc-zipfile-location>
  </install-info>
</jdbc-driver>
<data-source name="Sample Datasource" action="update">
  <database-name>/usr/WebSphere/AppServer/bin/myidb.prp</database-name>
  <jdbc-driver-name>Admin DB Driver</jdbc-driver-name>
  <minimum-pool-size>1</minimum-pool-size>
  <maximum-pool-size>10</maximum-pool-size>
```

Appendix F. XML Package Summary

```
<connection-timeout>120000</connection-timeout>
<idle-timeout>180000</idle-timeout>
<orphan-timeout>1800000</orphan-timeout>
```

The following section contains the **node**, or physical machine, in the administrative domain. Its hostname is buccaneer:

```
<node name="buccaneer" action="update">
  <deployed-jar-directory>/usr/WebSphere/AppServer/deployedEJBs</deployed-jar-directory>
  <dependent-classpath></dependent-classpath>
```

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

```
<application-server name="Default Server" action="update">
  <executable>java</executable>
  <command-line-arguments/>
  <environment/>
  <user-id></user-id>
  <group-id></group-id>
  <working-directory></working-directory>
  <umask>18</umask>
  <stdin></stdin>
  <stdout>/usr/WebSphere/AppServer/logs/default_server_stdout.log</stdout>
  <stderr>/usr/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-specification>
  <trace-output></trace-output>
  <transaction-log-file></transaction-log-file>
  <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>
```

In the Advanced edition, the application server maintains a **container for the EJBs** (only one EJB is shown in the example):

Appendix F. XML Package Summary

```
<container name="Default Container" action="update">
  <user-id></user-id>
  <password></password>
  <cache-config>
    <size>2047</size>
    <soft-limit>2000</soft-limit>
    <hard-limit>2047</hard-limit>
    <sweep-interval>1000</sweep-interval>
    <passivation-directory></passivation-directory>
  </cache-config>
  <ejb name="HitCount Bean" action="update">
    <jar-file>/usr/WebSphere/AppServer/deployedEJBs/_wlm_Inc.jar</jar-file>
    <home-name>IncBean</home-name>
    <user-id></user-id>
    <password></password>
    <create-db-table>>false</create-db-table>
    <find-for-update>>true</find-for-update>
    <minimum-pool-size>5</minimum-pool-size>
    <maximum-pool-size>500</maximum-pool-size>
    <primary-key-check>>true</primary-key-check>
    <db-exclusive-access>>false</db-exclusive-access>
    <data-source name="Sample Datasource"/>
  </ejb>
</container>
```

The application server also contains a **servlet engine**.

```
<servlet-engine name="Default Servlet Engine" action="update">
  <maximum-connections>25</maximum-connections>
  <transport-port>-1</transport-port>
  <transport-type name="ose">
    <ose-transport>
      <link-type>local</link-type>
      <log-file-mask trace="false" inform="false" warning="false" error="true"/>
      <queue-name>ibmoselink</queue-name>
      <clone-index>1</clone-index>
      <native-log-file>native.log</native-log-file>
    </ose-transport>
  </transport-type>
  <isclone>>false</isclone>
</servlet-engine>
```

Appendix F. XML Package Summary

In turn, the servlet engine contains a few **Web applications**. Two are omitted from this example. The third (shown here) is named "examples":

```
<web-application name="examples" action="update">
  <description>Useful Example Servlets</description>

</web-application>

<document-root>/usr/WebSphere/AppServer/hosts/default_host/examples/web</document-root>
  <classpath>
    <path value="/usr/WebSphere/AppServer/hosts/default_host/examples/servlets"/>
  </classpath>
  <error-page>/debug_error.jsp</error-page>
  <filter-list/>
  <group-attributes/>
  <auto-reload>true</auto-reload>
  <reload-interval>9000</reload-interval>
  <enabled>true</enabled>
  <root-uri>default_host/webapp/examples</root-uri>
  <shared-context>>false</shared-context>
  <shared-context-jndi-name>SrdSrvltCtxHome</shared-context-jndi-name>
  <isclone>>false</isclone>
</document-root>
```

The example application contains **several servlets**. The first one is administered by the name "simpleJSP":

```
<servlet name="simpleJSP" action="update">
  <description>Simple JSP Servlet</description>
  <code>SimpleJSPServlet</code>
  <init-parameters>
    <parameter name="" value=""/>
  </init-parameters>
  <load-at-startup>>false</load-at-startup>
  <debug-mode>>false</debug-mode>
  <uri-paths>
    <uri value="/simpleJSP.servlet"/>
    <uri value="/simpleJSP"/>
  </uri-paths>
  <enabled>true</enabled>
  <isclone>>false</isclone>
</servlet>
```

The Web application contains some other servlets that are not listed here. The Web application ends:

Appendix F. XML Package Summary

```
</web-application>
```

Now, a **session manager** follows. Recall, the tag for the servlet engine is still open, indicating that all of these objects are contained by the servlet engine:

```
<session-manager name="Session Manager" action="update">
  <enable-sessions>true</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="sesessionid">
    <comment>servlet session support</comment>
    <domain></domain>
    <maximum>-1</maximum>
    <path></path>
    <secure>>false</secure>
  </cookie>
  <interval-invalidation-time>1800</interval-invalidation-time>
  <persistent-sessions>>false</persistent-sessions>
  <persistence-type>directodb</persistence-type>
  <database location="jdbc:db2:was">
    <driver>COM.ibm.db2.jdbc.app.DB2Driver</driver>
    <user-id></user-id>
    <password></password>
    <number-of-connections>30</number-of-connections>
  </database>
  <enable-stat-collection>true</enable-stat-collection>
  <using-cache>>false</using-cache>
  <using-multi-row>>false</using-multi-row>
  <using-manual-update>>false</using-manual-update>
  <using-native-access>>false</using-native-access>
  <base-memory-size>1000</base-memory-size>
  <allow-overflow>true</allow-overflow>
  <data-source name=""/>
</session-manager>
```

Next, the servlet engine contains a user **profile manager**:

```
<user-profile-manager name="User Profile Manager" action="update">
  <enable-user-profile>>false</enable-user-profile>
  <data-wrapper>com.ibm.servlet.personalization.userprofile.UserProfile</data-wrapper>
```

Appendix F. XML Package Summary

```
<remote-interface-ro>com.ibm.servlet.personalization.userprofile.UP_ReadOnly</remote-interface-ro>

<remote-interface-rw>com.ibm.servlet.personalization.userprofile.UP_ReadWrite</remote-interface-rw
>

<home-interface-ro>com.ibm.servlet.personalization.userprofile.UP_ReadOnlyHome</home-interface-r
o>

<home-interface-rw>com.ibm.servlet.personalization.userprofile.UP_ReadWriteHome</home-interface
-rw>
  <jndi-name-ro>UP_ReadOnlyHome</jndi-name-ro>
  <jndi-name-rw>UP_ReadWriteHome</jndi-name-rw>
</user-profile-manager>
```

The user profile manager is the last object in the servlet engine and the application server. The end tags for each are displayed:

```
</servlet-engine>
</application-server>
```

The **servlet redirector** portion comes next:

```
<servlet-redirector name="Remote Servlet Redirector" action="update">
  <executable>java</executable>
  <command-line-arguments/>
  <environment/>
  <user-id></user-id>
  <group-id></group-id>
  <working-directory></working-directory>
  <umask>18</umask>
  <stdin></stdin>
  <stdout>/usr/WebSphere/AppServer/logs/redirector_stdout.log</stdout>
  <stderr>/usr/WebSphere/AppServer/logs/redirector_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>roundrobin</selection-policy>
  <trace-specification>*=all=disabled</trace-specification>
  <trace-output></trace-output>
```

Appendix F. XML Package Summary

```
<system-properties/>
<transport-port>9357</transport-port>
<max-con>50</max-con>
<enabled>>false</enabled>
<transport-type name="ose">
  <ose-transport>
    <link-type>local</link-type>
    <log-file-mask trace="false" inform="false" warning="false" error="true"/>
    <queue-name>redirector</queue-name>
    <clone-index>1</clone-index>
    <native-log-file>redirector.log</native-log-file>
  </ose-transport>
</transport-type>
</servlet-redirector>
```

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

```
</node>
```

Several **enterprise application entries** come after this (only the first is shown here):

```
<enterprise-application name="AdminApplication" action="create">
  <uri name="/admin"/>
  <web-application name="admin"/>
</enterprise-application>
```

Followed by several **URI security entries** (only one is shown):

```
<uri-security>
  <uri name="/admin"/>
  <method-group-mapping method="HTTP_GET" method-group="ReadMethods"/>
  <method-group-mapping method="HTTP_POST" method-group="ReadMethods"/>
  <method-group-mapping method="HTTP_PUT" method-group="WriteMethods"/>
  <method-group-mapping method="HTTP_DELETE" method-group="RemoveMethods"/>
</uri-security>
```

Followed by the **server security configuration**:

```
<security-config security-enabled="false" security-cache-timeout="600">
  <app-security-defaults>
    <realm-name>raleigh.ibm.com</realm-name>
```

Appendix F. XML Package Summary

```
<challenge-type ssl-enabled="false">
  <basic-challenge/>
</challenge-type>
</app-security-defaults>
<auth-mechanism>
  <localos>
    <user-id>root</user-id>
    <password>$server-password$</password>
  </localos>
</auth-mechanism>
</security-config>
```

The last set of tags defines the **method groups**:

```
<method-group>ReadMethods</method-group>
<method-group>WriteMethods</method-group>
<method-group>RemoveMethods</method-group>
<method-group>CreateMethods</method-group>
<method-group>ExecuteMethods</method-group>
<method-group>FinderMethods</method-group>
```

The last item is the end tag for export itself:

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

XMLConfig - Example of a partial export

To do a partial export of your Websphere Administrative Domain configuration, 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: `xmlconfig -export <fileName> -adminNodeName <adminNodeName> -partial <partialFileName>`

The partial xml file always starts with `<websphere-sa-config>` and ends with `</websphere-sa-config>`. What goes in between these tags depends on what you want to export. This example of a partial xml file you would create to export the default applications on a node named 'mynode':

1. Export one Application Server's contents (PartialFile1.xml):

```
<?xml version="1.0"?>
<!DOCTYPE websphere-sa-config SYSTEM "$server_root$$dsep$bin$dsep$xmlconfig.dtd" >
<websphere-sa-config>
```

Appendix F. XML Package Summary

```
<node name="mynode" action="locate">
  <application-server name="Default Server" action="locate">
    <servlet-engine name="Default Servlet Engine" action="locate">
      <web-application name="default_app" action="export">
        </web-application>
      </servlet-engine>
    </application-server>
  </node>
</websphere-sa-config>
```

2. The XMLConfig command to export a single web application (as defined by PartialFile2.xml above) would be:

```
XMLConfig -export NewExport.xml -adminNodeName mynode -partial PartialFile2.xml
```

3. An XML file named 'NewExport.xml' is created with the following output:

```
<?xml version="1.0"?>
<!DOCTYPE websphere-sa-config SYSTEM "$server_root$$dsep$bin$dsep$xmlconfig.dtd" >

<websphere-sa-config>
  <node name="mynode" action="locate">
    <application-server name="Default Server" action="locate">
      <servlet-engine name="Default Servlet Engine" action="locate">
        <web-application name="default_app" action="update">
          <description>Default Application</description>
          <document-root>/usr/HTTPServer/htdocs/en_US</document-root>
          <classpath>
            <path value="/usr/WebSphere/AppServer/hosts/default_host/default_app/servlets"/>
            <path value="/usr/WebSphere/AppServer/servlets"/>
          </classpath>
          <error-page>/ErrorReporter</error-page>
          <filter-list/>
          <group-attributes/>
          <auto-reload>true</auto-reload>
          <reload-interval>9000</reload-interval>
          <enabled>true</enabled>
          <root-uri>default_host</root-uri>
          <shared-context>false</shared-context>
          <shared-context-jndi-name>SrdSrvltCtxHome</shared-context-jndi-name>
          <isclone>false</isclone>
          <servlet name="snoop" action="update">
```

Appendix F. XML Package Summary

```
<description>Snoop servlet</description>
<code>SnoopServlet</code>
<init-parameters>
  <parameter name="param1" value="test-value1"/>
</init-parameters>
<load-at-startup>>false</load-at-startup>
<debug-mode>>false</debug-mode>
<uri-paths>
  <uri value="/servlet/snoop"/>
  <uri value="/servlet/snoop2"/>
</uri-paths>
<enabled>>true</enabled>
<isclone>>false</isclone>
</servlet>
<servlet name="hello" action="update">
  <description>Simple Hello servlet</description>
  <code>HelloWorldServlet</code>
  <init-parameters/>
  <load-at-startup>>false</load-at-startup>
  <debug-mode>>false</debug-mode>
  <uri-paths>
    <uri value="/servlet/hello"/>
  </uri-paths>
  <enabled>>true</enabled>
  <isclone>>false</isclone>
</servlet>
<servlet name="ErrorReporter" action="update">
  <description>Default error reporter servlet</description>
  <code>com.ibm.servlet.engine.webapp.DefaultErrorReporter</code>
  <init-parameters/>
  <load-at-startup>>true</load-at-startup>
  <debug-mode>>false</debug-mode>
  <uri-paths>
    <uri value="/ErrorReporter"/>
  </uri-paths>
  <enabled>>true</enabled>
  <isclone>>false</isclone>
</servlet>
<servlet name="invoker" action="update">
  <description>Enables Serving Servlets By Classname</description>
  <code>com.ibm.servlet.engine.webapp.InvokerServlet</code>
  <init-parameters/>
```

Appendix F. XML Package Summary

```
<load-at-startup>true</load-at-startup>
<debug-mode>false</debug-mode>
<uri-paths>
<uri value="/servlet"/>
</uri-paths>
<enabled>true</enabled>
<isclone>false</isclone>
</servlet>
<servlet name="jsp10" action="update">
<description>JSP 1.0 support servlet</description>
<code>com.sun.jsp.runtime.JspServlet</code>
<init-parameters/>
<load-at-startup>true</load-at-startup>
<debug-mode>false</debug-mode>
<uri-paths>
<uri value="*.jsp"/>
<uri value="*.jsw"/>
<uri value="*.jsw"/>
</uri-paths>
<enabled>true</enabled>
<isclone>false</isclone>
</servlet>
</web-application>
</servlet-engine>
</application-server>
</node>
</websphere-sa-config>
```

XMLConfig - Using the Tool Programmatically

The XMLConfig class is structured so that you can use it programmatically to retrieve information as a TXDocument 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. These variables are available:

\$server_root\$

Appendix F. XML Package Summary

Replace with the WebSphere Application Server installation directory, such as C:\WebSphere\AppServer on Windows NT.

\$psep\$

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

- On Windows NT, it is ; (semicolon)
- On AIX and Solaris, it is : (colon)

\$dsep\$

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

- On Windows NT, it is \ (backward slash)
- On AIX and Solaris, it is / (forward slash)

Security XML Configurations

When configuring security with XML, you should be familiar with the following two factors:

1. Passwords and variable substitution
2. Searches of the user registry

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 xml for repository objects.

Javadoc for 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 the class names, such as ApplicationServerConfig. The Javadoc is labeled by the class name preceded by the package name, com.ibm.websphere.

XMLConfig - Passwords and Variable Substitution

Passwords are required for the WebSphere security configuration, but exposing passwords during an XML import or export by putting an unencrypted password in the XML text file would create an unacceptable security risk. Thus, WebSphere Application Server uses special password tags in the exported XML file that signify XML variables that replace the real passwords during an import. Consequently, the passwords are only part of the XMLConfig command line, which you can clear after invoking the utility.

Three of the password variables have constant names while the others depend on the name to what they are related. The three constant variable names are:

Appendix F. XML Package Summary

server-password: This is the password for the ID that has all permissions and rights to access the administrative server. It is the password that is entered during the installation process, and the one that appears on the User Registry tab of the Configure Global Security Settings task.

ltpa-password: This is the password for generating LTPA keys. It is the password entered on the LTPA Password dialog when changing the Authentication Mechanism to LTPA for the first time.

ldap-bindpwd: This is the password that the security server will use to bind to an LDAP directory during searches. It is the password entered on the User Registry tab of the Configure Global Security Settings task when LTPA is the Authentication Mechanism.

<enterprise app name>_AppSecurityPwd [*where <enterprise app name> is the name of an enterprise application with a defined application identity*] This is the password associated with the identity defined as the application identity. Application identities are set on the last panel of the Configure Application Security task and are limited to users defined in the configured user registry.

All password variables must be substituted on the XMLConfig command line using the -substitute parameter. Multiple substitutions are separated by a semicolon. For example, the server-password and LTPA-password variables may be substituted with the following command line:

```
XMLConfig -import was.xml -adminNodeName myhost -substitute  
"server-password=pwd1;LTPA-password=pwd2"
```

Appendix G. Concurrency

Entity EJB

Container Managed Persistence (CMP)

TransactionAttribute = REQUIRED

IsolationLevel = READ_COMMITTED

Oracle	increment(amount)	DB2
*	Start transaction	
*	pstmt1 = "select y from table where x = ?"	
*	pstmt1.setInt(1,key)	
* ----->	rs = executeQuery(pstmt1)	
*	rs.next()	<-----
*Lock life	oldY = rs.getInt(1)	Lock life
*	rs.close	<-----
@	newY = oldY + amount	
#	pstmt2.setInt(1,newY)	
#	pstmt2.setInt(2,key)	
#	executedUpdate(pstmt2)	
# ----->	End transaction	

* = preInvoke() processing

@ = increment() processing

= postInvoke() processing

Appendix H. "Where To Find It" Guide

Enterprise JavaBean

- Developer's Guide to Understanding Enterprise JavaBeans Applications, 1999
<http://www.nova-labs.com/ejbguide/ejbguide.htm>
- Enterprise JavaBeans Technology: Server Component Model for the Java Platform, December 1998
http://java.sun.com/products/ejb/white_paper.html
- Enterprise JavaBeans Specification Version 1.1, February 2000
<http://java.sun.com/products/ejb/docs.html>
- Writing Enterprise Beans in WebSphere
<http://www.ibm.com/software/webservers/appserv/library.html>

Java

- Application Programming Guide and Reference for Java (Chapter 2).
<ftp://ftp.software.ibm.com/software/os390/db2server/books/DSNJV0G2.PDF>
- An Introduction to Sevlets, Web Developers Journal, March 1999
http://WebDevelopersJournal.com/articles/intro_to_servlets.html
- Thinking in Java Second Edition, by Bruce Eckel, 2000
<http://205.178.180.120/filepump.com/codecuts/pdfs/bruceeckel/TIJ2.pdf>

Lightweight Directory Access Protocol (LDAP)

- IBM SecureWay LDAP Directory
<http://www.ibm.com/software/network/directory/library/>
- IBM WebSphere and VisualAge for Java Database Integration with DB2, Oracle, and SQL Server (SG24-5471-00) September 1999
<http://www.redbooks.ibm.com/abstracts/sg245471.html>
- LDAP Implementation Cookbook (SG24-5110-00) June 1999
<http://www.redbooks.ibm.com/abstracts/sg245110.html>
- Specialized LDAP configuration settings information (from *WebSphere Security* lecture)
<http://www-4.ibm.com/software/webservers/appserv/doc/v30/ae/web/help/secure3.htm>
- Scaling Directory Size with IBM eNetwork LDAP Directory Version 2.1
http://www.ibm.com/software/network/directory/library/whitepapers/ldap_scaling.html
- Understanding LDAP (SG24-4986-00) November 1999
<http://www.redbooks.ibm.com/abstracts/sg244986.html>

Professional Certification Program from IBM

- Road maps for Application Development Certification paths are available at:
http://www.ibm.com/education/certify/certs/ad_index.phtml

Appendix H. “Where To Find It” Guide

Resource Description Framework

Resource Description Framework (RDF)
<http://www.w3.org/RDF>

S/390

To obtain whitepapers, look under the “e-business” heading at:
<http://www.s390.ibm.com/marketing/position.html>

For S/390 e-business information:
<http://www.s390.ibm.com/ebusiness>

Java for OS/390
<http://www.s390.ibm.com/java>

Security

Deploying a Public Key Infrastructure (SG24-5512-00) March 2000
<http://www.redbooks.ibm.com/abstracts/sg245512.html>

IBM Application Framework for e-business: Security, November 1999
<http://www.ibm.com/software/developer/library/security/index.html>

IBM WebSphere Standard/Advanced 3.0 Security Overview, February 2000
<http://www.ibm.com/software/webservers/appserv/security.pdf>

Internet Security in the Network Computing Framework (SG24-5220-00) September 1998
<http://www.redbooks.ibm.com/abstracts/sg245220.html>

Introduction to Public-Key Cryptography, October 1998
<http://developer.netscape.com/docs/manuals/security/pkin/index.htm>

Introduction to SSL, October 1998
<http://developer.netscape.com/docs/manuals/security/sslin/index.htm>

Java 2 Network Security (SG24-2109-01) June 1999
<http://www.redbooks.ibm.com/abstracts/sg242109.html>

UNIX

UNIXhelp for Users
<http://cc.uoregon.edu/unixhelp/index.html>

VisualAge for Java

Application Development with VisualAge for Java Enterprise (SG24-5081-00) April 1998
<http://www.redbooks.ibm.com/abstracts/sg245081.html>

Configuring the VisualAge for Java WebSphere Test Environment, February 2000
<http://www.software.ibm.com/vad.nsf/Data/Document2202?OpenDocument&p=1&BCT=1&Footer=1>

Appendix H. "Where To Find It" Guide

- Creating DataSources in the VisualAge for Java WebSphere Test Environment, February 2000
<http://www.software.ibm.com/vad.nsf/Data/Document2331?OpenDocument&p=1&BCT=1&Footer=1>
- Debugging Servlets and JSP with WebSphere Application Server and VisualAge for Java, August 1999
<http://www.software.ibm.com/vad.nsf/Data/Document3425?OpenDocument&p=1&BCT=1&Footer=1>
- Developing Enterprise JavaBeans with VisualAge for Java, August 1999
<http://www.software.ibm.com/vad.nsf/Data/Document3424?OpenDocument&p=1&BCT=1&Footer=1>
- Enterprise JavaBeans Development Using VisualAge for Java (SG24-5429-00) June 1999
<http://www.redbooks.ibm.com/abstracts/sg245429.html>
- IBM VisualAge for Java 3.0 PDF Documentation
<http://www7.software.ibm.com/vad.nsf/Data/Document3831>
- JavaServer Pages (JSP) Technology Syntax 1999
<http://java.sun.com/products/jsp/syntax.pdf>
- Programming with VisualAge for Java Version 2 (SG24-5264-00) November 1998
<http://www.redbooks.ibm.com/abstracts/sg245264.html>
- Using VisualAge for Java Enterprise Version 2 to Develop CORBA and EJB Applications (SG24-5276-00) December 1998
<http://www.redbooks.ibm.com/abstracts/sg245276.html>
- Using VisualAge for Java To Develop Domino Applications (SG24-5424-00) September 1999
<http://www.redbooks.ibm.com/abstracts/sg245424.html>
- VisualAge for Java Enterprise Version 2: Data Access Beans - Servlets - CICS Connector (SG24-5265-00) December 1998
<http://www.redbooks.ibm.com/abstracts/sg245265.html>
- VisualAge for Java Enterprise Version 2 Team Support (SG24-5245-00) September 1998
<http://www.redbooks.ibm.com/abstracts/sg245245.html>
- VisualAge for Java Version 3: Persistence Builder with GUIs, Servlets, and Java Server Pages (SG24-5426-01) March 2000
<http://www.redbooks.ibm.com/abstracts/sg245426.html>
- WebSphere Support in VisualAge for Java 3.0
<http://www.software.ibm.com/vad.nsf/Data/Document2332?OpenDocument&p=1&BCT=1&Footer=1>

WebSphere

- A Firm Foundation For Developing Web Applications
<http://www-4.ibm.com/software/developer/features/feat-framework.html>
- An Approach to Designing e-business Solutions (SG24-5949-00) March 2000
<http://www.redbooks.ibm.com/abstracts/sg245949.html>
- Additional documentation for WebSphere Application Server
<http://www.ibm.com/software/webservers/appserv/additionaldoc.html>
- Building Business Solutions with WebSphere Version 3.0, January 2000
<http://www.transarc.com/Library/documentation/websphere/appserv/atwswfg/atwswfg00.pdf>
- Converting XML to HTML using XSL in the IBM WebSphere environment
<http://www.developer.ibm.com/library/ref/xmlxsl.html>
- Deploying BEA WebLogic Enterprise Beans in WebSphere Application Server Advanced Edition, September 1999
<http://www.ibm.com/software/webservers/appserv/weblogic.pdf>
- Design Considerations: From Client Server Applications to e-business Applications (SG24-5503-00) November 1999
<http://www.redbooks.ibm.com/abstracts/sg245503.html>

Appendix H. “Where To Find It” Guide

- Developing an e-business Application for the IBM WebSphere Application Server (SG24-5423-00) September 1999
<http://www.redbooks.ibm.com/abstracts/sg245423.html>
- IBM Application Framework for e-business: Web Application Programming Model, 1999
<http://www.ibm.com/developer/features/framework/framework.html>
- IBM WebSphere and VisualAge for Java Database Integration with DB2, Oracle, and SQL Server (SG24-5471-00) September 1999
<http://www.redbooks.ibm.com/abstracts/sg245471.html>
- IBM WebSphere Application Server Library
<http://www.ibm.com/software/webservers/appserv/library.html>
- IBM WebSphere Application Server Whitepapers
<http://www.ibm.com/software/webservers/appserv/whitepapers.html>
- IBM WebSphere Application Server with Site Analysis Technology, 1999
<http://www.ibm.com/software/webservers/appserv/siteanalysiswpa.pdf>
- IBM WebSphere Standard/Advanced 3.0 Security Overview, February 2000
<http://www.ibm.com/software/webservers/appserv/security.pdf>
- IBM WebSphere Application Server Version 3.0 Documentation Center
http://www.ibm.com/software/webservers/appserv/doc/v30/ae/web/doc/begin_here/index.html
- IBM WebSphere Application Server Advanced Edition architecture, July 1999
<http://www.ibm.com/software/developer/library/wsarchitecture/wsarchitecture.html>
- IBM Websphere Application Server Troubleshooter for workstations
<http://www-4.ibm.com/software/webservers/appserv/doc/troubleshooter/2tabcont.htm>
- Installing WebSphere Application Server Advanced Edition, Version 3.02, on AIX
http://www.ibm.com/software/webservers/appserv/installing_on_aix.html
- Integrating Web Application Server and Multi-Database Server Technologies, January 2000
<http://www7.software.ibm.com/vad.nsf/Data/Document2201?OpenDocument&p=1&BCT=1&Footer=1>
- Introduction to WebSphere Application Server Version 3.0, June 1999
<http://www.transarc.com/Library/documentation/websphere/appserv/atswig/atswig00.pdf>
- Linux for WebSphere and DB2 Servers (SG24-5850-00) October 1999
<http://www.redbooks.ibm.com/abstracts/sg245850.html>
- Post-release developer notes for IBM WebSphere Application Server Version 3.02
<http://www.ibm.com/software/webservers/appserv/postrelease.html>
- The Front of IBM WebSphere Building e-business User Interfaces (SG24-5488-00) January 2000
<http://www.redbooks.ibm.com/abstracts/sg245488.html>
- WebSphere Application Server Architecture and Programming Model, 1998
<http://www.ibm.com/java/education/websphere-app/websphere-app.html>
- WebSphere Application Server Advanced Edition Version 3.02 Getting Started, December 1999
<http://www.ibm.com/software/webservers/appserv/doc/v30/ae/as3admsb.pdf>
- WebSphere Application Server Advanced Edition, Version 3.0 Specifications, September 1999
<http://www.ibm.com/software/webservers/appserv/nss3912n.pdf>
- WebSphere Application Server Enterprise Edition Version 3.0 Specifications, 1999
<http://www.ibm.com/software/webservers/appserv/nss3920.pdf>
- WebSphere Application Server Standard Edition Version 3.0 Specifications, 1999
<http://www.ibm.com/software/webservers/appserv/nss3911j.pdf>

Appendix H. “Where To Find It” Guide

- WebSphere Application Servers: Standard and Advanced Editions (SG24-5460-00) July 1999
<http://www.redbooks.ibm.com/abstracts/sg245460.html>
- WebSphere Application Server Standard and Advanced Editions, Version 3.0 Performance Report, February 2000
http://www.ibm.com/software/webservers/appserv/was_performance.pdf
- WebSphere Catalog Architect Specifications, January 2000
<http://www.ibm.com/software/webservers/commerce/servers/CAfinal.pdf>
- WebSphere Commerce Suite Specifications, January 2000
http://www.ibm.com/software/webservers/commerce/servers/nss3952s_wcsspec.pdf
- WebSphere Product Family Brochure, 1999
<http://www.ibm.com/software/webservers/webbrochure.pdf>
- WebSphere Site Analyzer Version 3.02
<http://www.ibm.com/software/webservers/appserv/doc/v302/SiteAnalyzer/sadocs.pdf>
- WebSphere Studio, Version 3.0 Specifications, August, 1999
<http://www.ibm.com/software/webservers/studio/nss3858i.pdf>
- WebSphere Studio and VisualAge for Java Servlet and JSP Programming (SG24-5755-00) March 2000
<http://www.redbooks.ibm.com/redpieces/abstracts/sg245755.html>
- WebSphere Studio Library
<http://www.ibm.com/software/webservers/studio/library.html>
- WebSphere V3 Performance Tuning Guide (SG24-5657-00) March 2000
<http://www.redbooks.ibm.com/abstracts/sg245657.html>
- Writing Enterprise Beans in WebSphere (SC09-4431-01) June 1999
<http://www.ibm.com/software/webservers/appserv/library>

Workload Management

- Caching and Filtering to Manage Internet Traffic and Bandwidth Demand (REDP0003) January 1999
<http://www.redbooks.ibm.com/redpapers/abstracts/redp0003.html>
- Highly Available IBM eNetwork Firewall Using HACMP or eNetwork Dispatcher (SG24-5136-00) July 1999
<http://www.redbooks.ibm.com/abstracts/sg245136.html>
- IBM Network Dispatcher Version 2.1 - Scalability, Availability and Load-balancing for TCP/IP Applications, April 1999
<ftp://ftp.software.ibm.com/software/network/dispatcher/whitepapers/nd21wp.pdf>
- IBM WebSphere Application Server servlet redirectors: Features, installation, and configuration
http://www.ibm.com/software/webservers/appserv/servletredirector_fic.html
- IBM WebSphere Performance Pack Cache Manager for Multiplatforms Getting Started Version 3.0, May 1999
<http://www.ibm.com/software/webservers/cacheman/doc/v30/wtegsfst.pdf>
- IBM WebSphere Performance Pack Cache Manager for Multiplatforms, V3.0 Spec Sheet
<http://www.ibm.com/software/webservers/cacheman/nss3926d.pdf>
- IBM WebSphere Performance Pack: Caching and Filtering with IBM Web Traffic Express (SG24-5859-00) October 1999
<http://www.redbooks.ibm.com/abstracts/sg245859.html>
- IBM WebSphere Performance Pack for Multiplatforms Getting Started Version 3.0 (GC34-4787-00) August 1999
<http://www.ibm.com/software/webservers/perfpack/library.html>
- IBM WebSphere Performance Pack for Multiplatforms Version 3.0 Spec Sheet, August 1999
<http://www.ibm.com/software/webservers/perfpack/nss3816e.pdf>

Appendix H. "Where To Find It" Guide

IBM WebSphere Performance Pack Library

<http://www.ibm.com/software/webservers/perfpack/library.html>

IBM WebSphere Performance Pack: Load Balancing with IBM SecureWay Network Dispatcher (SG24-5858-00) October 1999

<http://www.redbooks.ibm.com/abstracts/sg245858.html>

IBM WebSphere Performance Pack Usage and Administration (SG24-5233-00) December 1998

<http://www.redbooks.ibm.com/abstracts/sg245233.html>

IBM WebSphere Performance Pack: Web Content Management with IBM AFS Enterprise File System (SG24-5857-00) October 1999

<http://www.redbooks.ibm.com/abstracts/sg245857.html>

IBM Web Traffic Express for Multiplatforms Programming Guide Version 3.0, 1999

<http://www.ibm.com/software/webservers/perfpack/doc2/wtepg/wtepgmst.pdf>

IBM Web Traffic Express for Multiplatforms Webmaster's Guide Version 3.0, 1999

<http://www.ibm.com/software/webservers/perfpack/doc2/wtewg/wtewgmst.pdf>

Load-Balancing Internet Servers (SG24-4993-00) April 1998

<http://www.redbooks.ibm.com/abstracts/sg244993.html>

Load Balancing for eNetwork Communications Servers (SG24-5305-00) April 1999

<http://www.redbooks.ibm.com/abstracts/sg245305.html>

The Web site for SecureWay Network Dispatcher

<http://www.software.ibm.com/network/dispatcher/library>

Web Caching and Filtering with IBM WebSphere Performance Pack (REDP0009) March 1999

<http://www.redbooks.ibm.com/redpapers/abstracts/redp0009.html>

Workload Management: SP and Other RS/6000 Servers (SG24-5522-00) March 2000

<Http://www.redbooks.ibm.com/abstracts/sg245522.html>

XML - Extensible Markup Language

Frequently Asked Questions about the Extensible Markup Language

<http://www.ucc.ie/xml/>

IBM Developer Works XML Zone

<http://www.ibm.com/developer/xml/>

Recently Adopted Specifications: OMG Modeling - XMI

<http://cgi.omg.org/techprocess/meetings/schedule/tech2a.html#mod>

The XML Files: Using XML and XSL with IBM WebSphere 3.0 (SG24-5479-00) March 2000

<http://www.redbooks.ibm.com/abstracts/sg245479.html>

XML Schema Primer

<http://www.w3.org/XML/Schema>

Appendix I. Features of WebSphere Servers

	Standard	Advanced	Enterprise
Security Service	Web server security plug-in	Web server security plug-in, Object service plug-in	Policy Director and Component Broker
Delegation	Within security runtime if you are using LTPA	Within security runtime, if you are using LTPA	Through DCE/Kerberos
Credential Mapping	No	No	Yes
EJB Support	None	EJBs, EJServer, RMI, IIOP	EJBs, EJServer, RMI, IIOP
Legacy Connections	None	Connectors, for persistent object data stores	Adaptors, for connection to legacy applications and stores

Appendix J. Software Prerequisites

The following is a recent list of software prerequisites for IBM WebSphere Standard and Advanced Edition Application Servers. The list was taken from the IBM WebSphere Web site. This site is frequently updated. **To make sure you have the correct software versions for your installation, be sure to check the online information!** At the time this information was compiled, the URL for the information was:

http://www-4.ibm.com/software/webservers/appserv/doc/v35/a_as.htm

** included with Application Server

Platform: AIX

Browsers (one of the following)

Netscape Communicator 4.7 Or later version (must support HTML4 and CSS).

Netscape Navigator 4.07 Or later version (must support HTML4 and CSS).

Compilers (for development only)

VisualAge C++ 3.6.6.0 **

Databases (one of the following)

DB2 6.1 ** Must install Fixpack 4. Support is for Enterprise Edition only.

InstantDB 3.1.3 **

Oracle 8i Release 2 8.1.6

Sybase Adaptive Server Enterprise 12.0

Java

Java 2 SDK 1.2.2 ** PTF 7 is recommended for Version 3.5, and required for Service Pack 1.

Operating system

AIX 4.3.3 Requires Maintenance Level 4330-02 for JDK 1.2.2 compatibility.

Web servers (one of the following)

IBM HTTP Server 1.3.12 **

Apache Server 1.3.12

Netscape Enterprise Server 3.6.3

Netscape Enterprise Server 3.5.1

iPlanet Web Server, Enterprise Edition 4.0

Lotus Domino Enterprise Server 5.0.2B

Lotus Domino Enterprise Server 5.0.5 Support for Version 5.0.5 begins with WAS 3.5, Service Pack 1.

Platform: Windows NT

Browsers (one of the following)

Internet Explorer 4.0.1 Service Pack 1 or higher (must support HTML4 and CSS).

Netscape Communicator 4.7 Or later version (must support HTML4 and CSS).

Netscape Navigator 4.07 Or later version (must support HTML4 and CSS).

Compilers (for development only)

Visual C++ 6.0

VisualAge C++ 3.5.7 **

VisualAge for Java Enterprise 3.5 **

VisualAge for Java Professional 3.5

Databases (one of the following)

DB2 6.1 ** Requires Fixpack 4. Support is for Enterprise Edition only.

InstantDB 3.1.3 **

Oracle 8i Release 2 8.1.6

Java

Java 2 SDK 1.2.2 ** PTF 7 is recommended for Version 3.5, and required for Service Pack 1.

Operating system

Windows NT Server 4.0 Requires Service Pack 4, 6A, or later. Avoid Service Packs 5 and 6.

Appendix J. Software Prerequisites

Web servers (one of the following)

IBM HTTP Server 1.3.12 **

Apache Server 1.3.12

Netscape Enterprise Server 3.6.3

Netscape Enterprise Server 3.5.1

Microsoft Internet Information Server 4.0

iPlanet Web Server, Enterprise Edition 4.0

Lotus Domino Enterprise Server 5.0.2B

Lotus Domino Enterprise Server 5.0.5 Support for Version 5.0.5 begins with WAS 3.5 Service Pack 1.

Platform: Windows 2000

Browsers

Internet Explorer 5.0 Or later version (must support HTML4 and CSS).

Databases (one of the following)

DB2 6.1 ** Requires Fixpack 4. Support is for Enterprise Edition only.

InstantDB 3.1.3 **

Java

Java 2 SDK 1.2.2 ** Requires PTF 7.

Operating system (one of the following)

Windows 2000 Server

Windows 2000 Advanced Server

Web servers (one of the following)

IBM HTTP Server 1.3.12 **

Internet Information Server 5.0

Platform: Solaris

Browsers (one of the following)

Netscape Communicator 4.7 Or later version (must support HTML4 and CSS).

Netscape Navigator 4.07 Or later version (must support HTML4 and CSS).

Compilers (for development only)

Sun Visual WorkShop C++ 5.0

Sun WorkShop C/C++ 5.0

Watcom C/C++ 11.0b [information](#) Last release; to be replaced by Code Warrior.

Databases (one of the following)

DB2 6.1 ** Requires Fixpack 4. Support is for Enterprise Edition only.

InstantDB 3.1.3 **

Oracle 8i Release 2 8.1.6

Sybase Adaptive Server Enterprise 12.0

Java

Java 2 SDK 1.2.2 ** Requires PTF _5a_4330104.

Operating system (one of the following)

Solaris 2.6 Requires maintenance level April 2000 or later.

Solaris 7 Requires maintenance level April 2000 or later.

Web servers (one of the following)

IBM HTTP Server 1.3.12 **

Apache Server 1.3.12

Netscape Enterprise Server 3.6.3

Netscape Enterprise Server 3.5.1

iPlanet Web Server, Enterprise Edition 4.0

Lotus Domino Enterprise Server 5.0.2B

Platform: HP-UX

Browsers (one of the following)

Netscape Communicator 4.8 Or later version (must support HTML4 and CSS).

Appendix J. Software Prerequisites

Netscape Navigator 4.08 Or later version (must support HTML4 and CSS).

Compiler (for development only)

HP C++ Softbench 6.5

Databases (one of the following)

DB2 6.1 ** Requires Fixpack 4. Support is for Enterprise Edition only.

InstantDB 3.1.3 **

Oracle 8i Release 2 8.1.6

Java

Java 2 SDK 1.2.2 ** Requires fix level 04e_2.

Operating system

HP-UX 11.0

Web servers (one of the following)

IBM HTTP Server 1.3.12 **

Apache Server 1.3.12

iPlanet Web Server, Enterprise Edition 4.0

Lotus Domino Enterprise Server 5.0.2B

Platform: OS/400

Browsers (one of the following)

Netscape Communicator 4.8 Or later version (must support HTML4 and CSS).

Netscape Navigator 4.08 Or later version (must support HTML4 and CSS).

Databases (one of the following)

DB2 400 ** The database is embedded in the operating system.

Java

Java 2 SDK 1.2.2 **

Operating system

OS/400 4.4

OS/400 4.5

OS/400 5.1

Web servers (one of the following)

Lotus Domino Go OS400/DG1 4.6.2.6