IBM WebSphere Application Server Network Deployment, Version 5.0.2

Troubleshooting

Note
Before using this information, be sure to read the general information under "Trademarks and service marks" on page v.

Compilation date: July 19, 2003

Contents

Trademarks and service marks

The following terms are trademarks of IBM Corporation in the United States, other countries, or both:

- Cloudscape
- Everyplace
- iSeries
- IBM
- Redbooks
- ViaVoice
- WebSphere
- zSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product and service names may be trademarks or service marks of others.

Chapter 1. Troubleshooting or problem determination

As an activity, troubleshooting or problem determination encompasses a wide range of tasks that might need to be performed at any phase in product usage. In addition to built-in preventative measures, the product provides a variety of tools to make problem determination easier.

The troubleshooting section of the documentation helps you understand why your enterprise application, application server, or product installation is not working, and helps you resolve the problem. There are several ways to find information for diagnosing and resolving problems:

- For tips on investigating common problems, organized by task, see Chapter 3, "Troubleshooting by task: what are you trying to do?," on page 5.
- To look up the explanation and recommended response for a particular WebSphere Application Server error message, see Chapter 5, "Message reference," on page 123
- For help finding product error and warning messages, interpreting messages, and configuring product log files, see Chapter 7, "Working with message logs," on page 127.
- Difficult problems might require the use of product tracing, which exposes the low-level flow of control and interactions among product components. For help understanding and using product traces, see Chapter 9, "Working with trace," on page 141.
- For help adding log and trace capability to your own application, see Chapter 10, "Adding logging and tracing to your application," on page 151.
- For help using product utilities to diagnose the problem, see Chapter 11, "Working with troubleshooting tools," on page 191.
- To find out how to look up documented problems, common mistakes, product prerequisites, and other problem-determination information on the IBM WebSphere Application Server public web site, or to obtain technical support, see Chapter 13, "Obtaining help from IBM," on page 211.

Chapter 2. Diagnosing and fixing problems

The purpose of this section is to aid you in understanding why your enterprise application, application server, or WebSphere Application Server is not working and to help you resolve the problem. Unlike performance tuning which focuses on solving problems associated with slow processes and unoptimized performance, problem determination focuses on finding solutions to functional problems.

The kind of problem you are encountering, and how much you already know about it, determine what steps to take to resolve it:

Steps for this task

1. For tips on investigating common problems organized according to tasks within WebSphere Application server, see Chapter 3, “Troubleshooting by task: what are you trying to do?,” on page 5.
2. For tips on how to investigate common kinds of problems based on the component that is causing the problem, see Chapter 4, “Troubleshooting by component: what is not working?,” on page 81.
3. If you already have an error message and want to quickly look up its explanation and recommended response, look up the message by selecting the **Quick reference** view of the WebSphere Application Server Version 5 InfoCenter and expanding **Messages**.
4. For help in knowing where to find error and warning messages, interpreting messages, and configuring log files, see Chapter 7, “Working with message logs,” on page 127.
5. Difficult problems can require the use of tracing, which exposes the low-level flow of control and interactions between components. For help in understanding and using traces, see Chapter 9, “Working with trace,” on page 141..
6. For help in adding log and trace capability to your own application, see Chapter 10, “Adding logging and tracing to your application,” on page 151.
7. For help in using WebSphere Application Server utilities to help you diagnose the problem, see Chapter 11, “Working with troubleshooting tools,” on page 191..

   Some of these tools are bundled with the product, and others are freely downloadable.
8. To find out how to look up documented problems, common mistakes, WebSphere Application Server prerequisites, and other problem-determination information on the WebSphere Application Server public web site, or to obtain technical support from IBM, see Chapter 13, “Obtaining help from IBM,” on page 211.

Chapter 3. Troubleshooting by task: what are you trying to do?

This section provides troubleshooting information based on the task you were trying to accomplish when the problem occurred. To find more information about your problem, select a task category from the list below.

If you do not see a task that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

Troubleshooting installation problems

Select the problem you are having with WebSphere Application Server installation:

- Install completes with errors or warnings, or installation hangs.
- The installation process completes, but the application server will not start.
- The installation process completes, but sample applications, such as the snoop servlet or other applications from the Sample Gallery do not work.

If you have installed WebSphere Embedded Messaging and are having problems uninstalling or reinstalling WebSphere Application Server, see Reinstalling WebSphere Application Server with Embedded Messaging.

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, see ″Troubleshooting the Installation″ in the WebSphere Application Server Version 5 InfoCenter. If that does not provide you with a resolution to your problem, contact IBM support for further assistance

Installation completes with errors or warnings, or hangs (panel appears, but shows no progress)

If the WebSphere Application Server installation program indicates that errors were encountered while installing the product:

- Browse the file main installation log `install_dir/logs/log.txt` for clues.
- Look the command prompt from which the installation panel that hangs was launched, for error messages.
- Look up any error or warning messages in the message reference table.
- For Unix or AIX users, if you have uninstalled WebSphere Application Server prior to reinstalling it, verify that all related packages have been removed by using SMIT or similar tool and looking for packages beginning ″WS″. If found, remove them.
- Review ″Troubleshooting the installation″ in the WebSphere Application Server V5.0.2 InfoCenter..

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

Installation completes but the administrative console does not start

What kind of problem are you having?

- ″Internal Server Error″, ″Page cannot be found″, 404, or similar error trying to view administrative console.
- ″Unable to process login. Please check User ID and password and try again. ″ error when trying to access console page.
- Directory paths in the console are garbled.

If you are able to bring up the browser page, but the console's behavior is inconsistent, error-prone, or unresponsive, try upgrading the browser you are using. Older browsers may not support the administrative console's features.

If none of these steps solves the problem, check to see if the problem has been identified and documented using the links in Chapter 12, "Diagnosing and fixing problems: Resources for learning," on page 209. If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

″Internal Server Error″, ″Page cannot be found″, 404, or similar error trying to view administrative console

If you are unable to view the administrative console, here are some steps to try:

- Verify that the application server which supports the administrative console is up and running.
  - For a ″base″ configuration, the administrative console is deployed by default on ″server1″. Before viewing the administrative console, you must
    - Run the **startServer server1** command for Windows or **./startServer.sh server1** command for Unix from a command prompt in the *install_dir*\bin directory, or
    - Click the ″start application server″ link from the ″first steps″ panel, or
    - Start WebSphere Application Server as a service or from the Start menu, if you are using Windows.
  - If you are using the Deployment Manager (for a multi-node configuration), run the startManager command from the *Network_Deployment_install_dir*\bin directory.
  - View the SystemOut.log file for the application server or deployment manager to verify that the server supporting the administrative console has actually started.
- Check the URL you are using to view the console. By default, it is http://*server_name*:9090/admin.
- If you are browsing the console from a remote machine, try to eliminate connection, address and firewall issues by:
  - Pinging the server machine from a command prompt, using the same server name as in the URL.
- If you have never been able to access the administrative console, verify that the installation was successful.

″Unable to process login. Please check User ID and password and try again. ″ error when trying to access console page

This error indicates that security has been enabled for WebSphere Application Server, and the user ID or password supplied is either invalid or not authorized to access the console.

To access the console,

- If you are the administrator, use the ID defined as the security administrative ID. This ID is stored in the WebSphere Application Server directory structure in the file security.xml.
- If you are not the administrator, ask the administrator to enable your ID for the administrative console.

Directory paths in the console are garbled

If directory paths used for classpaths or resources specified in the Application Assembly Tool, configuration files, or elsewhere, appear garbled in the administrative console, it may be because the Java runtime interprets a backslash (\) as denoting a control character.

To resolve, modify Windows-style classpaths by replacing occurrences of single backslashes to two. For example, change ″c:\MyFiles\MyJsp.jsp″ to ″c:\\MyFiles\\MyJsp.jsp″.

The application server or Deployment Manager does not start or starts with errors

If the WebSphere Application Server installation program completes successfully, but the application server does not start, or starts with errors:

- View the log files, which are located by default in *install_dir*\logs*server_name*\`SystemErr.log` and `SystemOut.log` for clues.
- If there are several applications deployed on an application server or node, it may take some time to start. Browse the SystemOut.log periodically and look at the most recent updates to see if the server is still starting up. On Unix platforms, the **tail -f *installation_path*/logs/`SystemOut.log`** is a convenient way to watch the progress of the server.
- Look for any errors or warnings relating to specific resources with the module, such as Web modules, enterprise beans and messaging resources. If you find any, examine the application server configuration file for that resource's configuration settings. For example, in a base (non-distributed) configuration on Windows systems, browse

  ```
  install_dir\config\cells\BaseApplicationServerCell\
        nodes\host_name\servers\server_name\server.xml
  ```

  and examine the xml tags for that resource's properties. Change its initialState value from ″START″ to ″STOP″. Then restart the server as a test to see if the problem is due to this component.
- Look up any error or warning messages in the message reference table by selecting the Quick Reference view and expanding the ″Messages″ heading.
- If the application server is part of a Network Deployment (multiple server) configuration,
  - Ensure that you have followed the steps for adding the application server to the configuration.
  - Ensure that the configuration is synchronized between the deployment manager and the node. If auto synchronization is running, wait until the synchronization has had a chance to complete. If you are using manual synchronization, request a synchronization to each node in the cluster.
  - Before starting an application server:
    1. Start the Deployment Manager process: ***installation_root*/bin/startManager.sh** or ***installation_root*\bin\startManager.bat**.

2. Complete the one-time step of "federating" the node the application server is running on to the Deployment Manager. This has to be done even if there is only one node, and it is the same physical server as the one on which the DeploymentManager is running. This is done by running the addnode *nodename* utility in the *installation_root*/bin directory of the application server's host.
3. Start the Node Manager process on the nodes hosting the application servers you want to run: ***installation_root*/bin/startNode.sh** or ***installation_root*\bin\startNode.bat**.

- Ensure that the logical name that you have specified to appear on the console for your application server does not contain invalid characters such as: - / \ : * ? " < >.
- If you are unable to start the DeploymentManager after an otherwise successful installation:
  - Look in the file *installation_root*/dmgr/logs/SystemErr.log and SystemOut.log for messages.
  - Where was the product installed? This product is not stand-alone, and depends upon some files which are already installed as part of the base. The Network Deployment product should be installed under the WebSphere Application Server root directory of one of the nodes with the base product, at the same level as the base product. For example, if the base product is in /usr/WebSphere/AppServer, the Network Deployment should be installed into a directory like /usr/WebSphere/NetworkDeployment. Installing the product apart from the base product may result in a error running the startManager command similar to: WSVR0102E: An error occurred stopping, null [class com.ibm.ws.cache.ServerCache].
- If you are using Cloudscape and receive an "ERROR XSDB6: Another instance of Cloudscape may have already booted the database databaseName." error starting application server, consult this topic for more information.
- When using a non-root user ID to run application servers, verify that the non-root user has write access to the *WebSphereRoot* `/AppServer/temp` directory
- When using a non-root user ID to run application servers, verify that the JVM has write access to *WebSphereRoot* `/config/plugin-cfg.xml`

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

Installation completes, but sample applications do not work

If the WebSphere Application Server installation program completes successfully, but the sample applications do not run:

- Browse the application server log files, which are located by default in *install_dir*\logs*server_name*\`SystemErr.log` and `SystemOut.log` for clues.
- Browse the JVM logs of the hosting application server for clues, after attempting to run a Sample application,
- Look up any error or warning messages in the message table by selecting the **Quick reference** view of the WebSphere Application Server Version 5 InfoCenter and expanding the **Messages** heading.
- You can also encounter some security-related problems, such as after turning on security, "MSGS0508E: The JMS Server security service was unable to authenticate userid:" error is displayed in SystemOut.log when starting an application server.

- Review ″Troubleshooting the Installation″ in the WebSphere Application Server Version 5 InfoCenter for more information.

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, check to see if your problem has been identified. See Chapter 12, “Diagnosing and fixing problems: Resources for learning,” on page 209 for information on getting the latest updates. If your problem has not been reported, contact IBM support for further assistance.

Reinstalling WebSphere Application Server with embedded messaging

If WebSphere Application Server embedded messaging is installed, the component can not uninstall cleanly when the product is uninstalled, and can cause problems upon reinstallation.

Some components of the product may not be uninstalled properly, which can cause problems when the product is reinstalled.

Remove messaging objects such as Queue Connection Factories and Queue Destinations from the configuration,using the administrative console page or `wsadmin` command tool, before uninstalling the product.

If these objects are not removed prior to an uninstall, the Messaging component detects that these objects still exist and, by design, will not uninstall itself. The file `mq_uninstall.log` will show messages similar to these:

```
Publish And Subscribe configuration exists>> - Not uninstalling Pub Sub
MQ Server configuration exists>> - not uninstalling MQ and MA88
```

In this case, it may be possible to simply reinstall the product, including embedded messaging. The installation will detect that the messaging component is already installed, and write error messages to indicate this. In this case it is recommended that you proceed with the installation. If the installation is successful, recreate messaging objects in the administrative console, but use different names for messaging objects than in the prior installation. Further steps, such as manually uninstalling embedded messaging in the Windows control panel, or editing the Windows registry, should not be attempted unless you have tried a reinstallation and it cannot complete or does not create a usable WebSphere Application Server.

If the reinstallation is not successful, follow the steps below for manually deleting the Messaging component, then reinstall WebSphere Application Server with messaging.

Even if all JMS resources, such as Queue Connection Factories and Queue Destinations, have been removed, or none have been created, it may still happen that the messaging component will not uninstall completely, and will cause the reinstall of the product to fail. This is indicated by the following message in the `mq_uninstall.log` file:

```
Return code from Publish And Subscribe uninstall was ERROR_INSTALL_FAILURE (1603)
```

Again, the message may be innocuous, that is the messaging component may be still resident and usable, and the recommended resolution is to first try reinstalling WebSphere Application Server with messaging, in the same location as before, ignoring messages indicating that Messaging is already installed, and to proceed normally unless problems are encountered.

If the reinstallation does not complete, or the reinstalled product does not behave correctly, follow these steps to manually remove the Embedded Messaging subcomponent so that the reinstallation will succeed:

Uninstalling embedded messaging on Windows platforms

First, try using the **Add/Remove Programs** application in the Windows Control Panel to remove `IBM WebSphere MQserver log files` and `IBM WebSphere EMPSserver log files`. This ensures that the programs are removed properly.

If the automated removal does not work, follow these steps to manually remove the programs:

1. Delete the product's registry keys by removing the key `HKLM\SOFTWARE\IBM\WebSphereEmbeddedMessagingPublishAndSubscribe`.
2. Delete the product's files.
3. Delete the product's Microsoft Software Installer (MSI) record. Use Microsoft's ″MSI Cleanup Utility″, available from Microsoft's Web site as `msicuu.exe`:
   a. Click on the downloaded `.exe` file to install it. Once the program is installed, it appears on the program menu. When started it lists all the products that MSI knows about.
   b. Select `IBM WebSphere EMPSserver log files` and click **Remove** to remove its MSI record.
4. Reboot and reinstall.

Troubleshooting migration problems

To resolve problems encountered in trying to migrate an application from an older version of WebSphere Application Server to version 5, first determine whether your problems occur using the pre-upgrade tool or the post-upgrade tool.

- Errors using the WASPreUpgrade tool.
- Errors using the WASPostUpgrade tool.
- For other kinds of migration problems, such as an application imported from another version of WebSphere Application Server that will not start, look up the related problem under Troubleshooting by task: what are you trying to do?, based on the problem you are having in this version.

If none of these steps fixes your problem,

- For general tips on migration problems, see (the migration utility).
- Review the topic Migration and its subtopics, which address migrating specific kinds of components.
- Check to see if the problem has been identified and documented by looking at the ((hints and tips, technotes, and fixes)).
- If you don't find your problem listed there contact IBM support.

Errors using the WASPreUpgrade tool

What kind of error are you encountering?

- ″MIGR0125E: The call to XMLConfig was not successful″
- ″MIGR0108E: The specified WebSphere directory does not contain WebSphere version that can be upgraded.″
- ″not found″ or ″no such file or directory″ message

Errors using the WASPostUpgrade tool

What kind of error are you encountering?

- ″not found″ or ″no such file or directory″ message
- ″MIGR0253E: The backup directory migration_backup_directory does not exist
- ″MIGR0102E: Invalid Command Line. MIGR0105E: You must specify the primary node name.″
- ″MIGR0116E: The backup directory [migration_backup_directory] does not contain the required xml data file.″
- ″MIGR0108E: The specified WebSphere directory does not contain WebSphere version that can be upgraded″

″MIGR0125E: The call to XMLConfig was not successful″ error when trying to run WASPreUpgrade

The WASPreUpgrade tool saves selected files from the WebSphere Application Server release 3.5.x and release 4.x `bin` directories. It also exports the existing application server configuration from the repository.

If you are migrating from WebSphere Application Server Release 4.0.x Advanced Edition, the WASPreUpgrade command calls the `XMLConfigserver log files` command to export the existing application server configuration from the repository. If errors occur during this part of the WASPreUpgrade command, you might have to apply fixes to the installation to successfully complete the export step. Contact IBM support for the latest fixes that might be applicable.

″MIGR0108E: The specified WebSphere directory does not contain WebSphere version that can be upgraded.″

Possible reasons for this error follow:

- If WebSphere Application Server Release 4.0.x is installed, you might not have run the WASPreUpgrade tool from the `bin` directory of the version 5 installation root.
  - If you see the following displayed when the WASPreUpgrade tool was run: ″IBM WebSphere Application Server, Release 4.0″, you are running the WebSphere Application Server Release 4.0 migration utility, not the version 5 migration utility.
  - The resolution is to alter your environment path or change the current directory so that you can launch the WebSphere Application Server version 5 WASPreUpgrade program.
- WebSphere Application Server version 5 might have installed onto the same root directory as the earlier version.
  - Confirm this situation by browsing the older version's directory structure to see whether it contains new 5.0 directories (such as `WebSphere\AppServer\logs\ffdc`).
  - The resolution is to uninstall all versions of WebSphere Application Server, then reinstall and reconfigure the older version, and then install WebSphere Application Server version 5 into a different root directory than the previous one.
- An invalid directory might have been specified when launching the WASPostUpgrade tool, or the WASPreUpgrade tool has not been run.

"not found" or "no such file or directory" message is returned from the WASPostUpgrade or WASPreUpgrade tool

This problem can occur if you are trying to run the WASPostUpgrade tool or the WASPreUpgrade tool from a directory other than `install_dir\bin`. The resolution is to ensure that the WASPostUpgrade or WASPreUpgrade `.bat` or `.sh` file resides in the `install_dir\bin` directory, and to launch it from that location.

"MIGR0253E: The backup directory migration_backup_directory does not exist." error returned from the WASPostUpgrade tool

Possible reasons for this error:

- The WASPreUpgrade tool was not run prior to the WASPostUpgrade tool. To verify this, check to see if the backup directory specified in the error message exists. If not, run the WASPreUpgrade `.bat` or `.sh` file, and then retry the WASPostUpgrade tool.
- You might have specified an invalid backup directory. For example, the directory might have been a subdirectory of the V3.5.x or V4.0.x tree, which was deleted after the WASPreUpgrade tool was run and the older version of the product was uninstalled, but before the WASPostUpgrade tool was run.
  - Determine if the full directory structure specified in the error message exists. If possible, rerun the WASPreUpgrade tool, specifying the correct full migration backup directory.
  - If the backup directory does not exist, and the older version it came from is gone, you must rebuild the older version from a backup repository or XML configuration file and rerun the WASPreUpgrade tool.

"MIGR0102E: Invalid Command Line. MIGR0105E: You must specify the primary node name."

The most likely cause of this error is that If release 4.0.x of the WebSphere Application Server is installed, the user might not have run the WASPostUpgrade tool from the `bin` directory of the WebSphere Application Server verion 5 installation root.

If you received the following messages when the WASPostUpgrade tool was run:

- IBM WebSphere Application Server, Release 4.0
- and

```
MIGR0002I: java com.ibm.websphere.migration.postupgrade.WASPostUpgrade
        <backupDirectoryName>
        -adminNodeName <primary node name>
        [-nameServiceHost <hostName> [ -nameServicePort <portNumber>]]
        [-substitute <"key1=value1[;key2=value2;[...]]">]
        In input xml file, the key(s) should appear as $key$ for substitution.")
        [-import <xml data file>]
        [-traceString <trace specification> [-traceFile <filename>]]}"
```

  this indicates that the release 4.0 migration tool was run.

To correct this problem, run the WASPostUpgrade command from the `bin` directory of the WebSphere Application Server version 5 installation root.

"MIGR0116E: The backup directory [migration_backup_directory] does not contain the required xml data file." error returned from the WASPostUpgrade tool.

Possible reasons for this error:

- If release 4.0.x of WebSphere Application Server is installed, you might not have run the WASPostUpgrade tool from the `bin` directory of the version 5.0 installation root.
  - If ″IBM WebSphere Application Server, Release 4.0″ is displayed when launching the WASPostUpgrade program, then the wrong version of the program is being executed.
  - To resolve this problem, run the WASPostUpgrade command from the `bin` directory of the 5.0 installation root.

Troubleshooting code deployment and installation problems

Select the problem you are having with deploying or installing developed code for WebSphere Application Server:

- Errors deploying enterprise beans
- Errors or problems deploying, installing, or promoting applications and databases

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

Errors deploying enterprise beans

What kind of error are you seeing?

- ConnectionFac E J2CA0102E: Invalid EJB component: Cannot use an EJB module with version 1.1 using The Relational Resource Adapter
- WSVR0040E: addEjbModule failed for MyApp-EJB.jar [class com.ibm.ws.runtime.component.DeployedModuleImpl] java.lang.NoClassDefFoundError: com/ibm/ejs/ras/Tr

If none of these errors match the ones you are seeing:

- Browse the server log files for the server containing the application for clues.
- Look up any error or warning messages in the message table.
- If the application server is part of a Network Deployment (multiple-server) configuration, ensure that you have followed the steps for adding the application server to the configuration.
- If the problems began after WebSphere Application Server security was enabled, view the topic Errors and access problems after enabling security in the WebSphere Application Server Version 5 InfoCenter.

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

WSVR0040E: addEjbModule failed for MyApp-EJB.jar [class com.ibm.ws.runtime.component.DeployedModuleImpl] java.lang.NoClassDefFoundError: com/ibm/ejs/ras/Tr

Possible causes of this error include:

- Security permissions are not given for the application in the `installation_root\properties\server.policy` file.

  To confirm that this is the problem, check the server.policy file to see if the security permissions are given for application.

To correct the problem, give permissions for application in the server.policyserver log files file. For example:

```
//purchaseOrder permission
grant codeBase "file:${was.install.root}/installedApps/myApp.ear/-"{
 permission java.security.AllPermission;
 };
```

where myApp.earserver log files is the application name. Imaginary Buffer Line

For details on how to use the policy tool to configure the server.policy file, see ″Configuring server.policy files″ in the WebSphere Application Server V5.0.2 InfoCenter..

- A was.policyserver log files file does not exist in the application/META-INF directory, while deploying the application on to the server.

  To confirm that this is the problem, if was.policy exists in application\META-INF directory, then check for syntax errors and make sure the application ear name is given correctly.To correct this problem, create a was.policy file in the EAR of the application containing the problem enterprise bean, under the [application]/META-INF directory with the following contents:

  ```
  // WebSphere Application Server Security Policy for the application you are running
  grant codeBase "file:myApp.ear" {
  permission java.security.AllPermission;
  };
  .
  ```

  For details on how to use the policy tool to configure the was.policy file, see Configuring was.policy files in the WebSphere Application Server InfoCenter.

Errors or problems deploying, installing, or promoting applications

What kind of problem are you having?

- I installed my application using wsadmin, but it does not show up under Applications-Manage Applications.
- I get a ″java.lang.RuntimeException: Failed_saving_bytes_to_wor_ERROR_″ in the Application Assembly Tool (AAT), administrative console or wsadmin
- I get a WASX7015E error running wsadmin command ″$AdminApp installInteractive″ or ″$AdminApp install″..
- A DDL generated by Application Assembly tool throws an SQL error on target platform.
- ADMA0004E: Validation error in task Specifying the Default Datasource for EJB 1.x Modules″ returned when installing application in administrative console or wsadmin.
- ″No valid target is specified in ObjectName<object> for module <module>″ from installation.
- addNode -includeapps option does not appear to upload all applications to the Deployment Manager.
- ″Timeout!!!″ error displays when attempting to install an enterprise application in the administrative console.
- During application installation, the call to EJB deploy throws an exception

If none of these steps fixes your problem,

- Ensure that the logical name (the name you have identified to appear on the console) for your application, enterprise bean module or other resource does not contain invalid characters such as these: - / \ : * ? " < > |.
- If the application was installed using the wsadmin $AdminApp install command with the -local flag, either restart the server or rerun the command without the -local flag.

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, check to see if the problem is identified and documented by looking at available online support including hints and tips, technotes, and fixes. If the problem has not been identified, contact IBM support for further assistance.

I installed my application using wsadmin, but it does not show up under Applications->Manage Applications

You may have installed the application but have not saved the configuration afterwards. This can be confirmed by:

- Verifying that the application has its subdirectory under install_dir/installedApps.
- Running the $AdminApp list command and verifying that the application is not among those displayed.
  - In the bin directory, run wsadmin.bat or wsadmin.sh.
  - From the wsadmin prompt, enter $AdminApp list and verify that the problem application is not among those displayed.

To resolve, reinstall your application via wsadmin, then run the command $AdminConfigsave in wsadmin before exiting wsadmin.

"java.lang.RuntimeException: Failed_saving_bytes_to_wor_ERROR_" in AAT, admin console or wsadmin

If you see this error when attempting to generate deployed code in the AAT, installing an application or module in the administrative console, or using the wsadmin tool to install an application or module, the system's temporary file path length may have been exceeded. This is typically an issue only on Windows platforms.

To verify that this is the problem, check your system's TEMP and TMP environment variables. If they are long, they are adding path length to the file names accessed by the EJBDeploy tool.

To resolve the problem:

1. Stop all WebSphere Application Server processes and close all DOS prompts.
2. Set the TMP and TEMP environment variables to something short, for example C:\TMP and C:\TEMP.
3. Re-install the application.

If this still doesn't work, try rebooting and re-deploy or reinstall the application.

WASX7015E error running wsadmin command "$AdminApp installInteractive" or "$AdminApp install"

This problem has two possible causes:

1. If the full text of the error is similar to:

   ```
   WASX7015E: Exception running command: "$AdminApp installInteractive
   C:/Documents and Settings/myUserName/Desktop/MyApp/myapp.ear";
   exception information:  com.ibm.bsf.BSFException: error while
   eval'ing Jacl expression: can't find method "installInteractive"
   with 3 argument(s) for class "com.ibm.ws.scripting.AdminAppClient"
   ```

   then the file and path name have been incorrectly specified. In this case, since the path included spaces, it was interpreted as multiple parameters by the wsadmin program.

   To resolve this problem, enter the path of the `.ear` file correctly. In this case, by enclosing it in double quotes: `$AdminApp installInteractive "C:\Documents and Settings\myUserName\Desktop\MyApps\myapp.ear"`.

2. If the full text of the error is similar to:

   ```
   WASX7015E: Exception running command:
   "$AdminApp installInteractive c:\MyApps\myapp.ear ";
   exception information: com.ibm.ws.scripting.ScriptingException:
   WASX7115E: Cannot read input file
   "c:\WebSphere\AppServer\bin\MyAppsmyapp.ear"
   ```

   then the application path is incorrectly specified. In this case, you must use UNIX-style "forward-slash" (/) separators in the path.

DDL generated by Application Assembly tool throws SQL error on target platform

If you receive SQL errors in attempting to execute Data Definition Language statements generated by the Application Assembly Tool on a different platform, for example if you are deploying a CMP enterprise bean designed on Windows onto a Unix server, here are some things to try:

- Browse the DDL statements for dependencies on specific user IDs and passwords, and correct as necessary.
- Browse the DDL statements for dependencies on specific server names, and correct as necessary.
- Refer to the vendor's message reference for causes and suggested actions regarding specific SQL errors. For IBM DB2, these may be viewed online at

  ```
  http://www.ibm.com/cgi-bin/db2www/data/
      db2/udb/winos2unix/support/index.d2w/report
  ```

- If you receive an error similar to

  ```
  SQL0104N  An unexpected token "CREATE TABLE AGENT  (COMM DOUBLE,  PERCENT  DOUBLE,  P"
  was found following "        ".  Expected tokens may include:  " ".  SQLSTATE=42601
  ```

  After executing a DDL file created on Windows on a UNIX platform, the problem may be due to a difference in file formats. To resolve this problem:

  – For UNIX platforms other than Linux, edit the DDL in the vi editor, removing the Ctl-M character at the beginning of each line.

  – For Linux, regenerate the deployment code for the application `EAR` on a Linux platform.

"ADMA0004E: Validation error in task Specifying the Default Datasource for EJB 1.x Modules" returned when installing application in admin console or wsadmin

If you see an error like the following when trying to install an application through the administrative console or the wsadmin command prompt:

```
AppDeploymentException: [ADMA0014E: Validation failed.
ADMA0004E: Validation error in task Specifying the Default
Datasource for EJB 1.x Modules  JNDI name is not
specified for module beannameBean Jar with URI filename.jar,
META-INF/ejb-jar.xml. You have not specified the
data source for each CMP bean belonging to this module.
Either specify the data source for each CMP beans or
specify the default data source for the entire module.]
```

one possible cause is that in WebSphere Application Server Version 4.0, it was mandatory to have a data source defined for each CMP bean in each JAR. In Version 5, you can specify either a data source for a CMP bean or a default data source for all CMP beans in the JAR. Thus during installation interaction (such as the installation wizard in the Administrative console), the data source fields are optional but the validation performed at the end of the install checks to see at least one of the above is specified.

To correct this problem, step through the installation again, and specify either a default datasource or a datasource for each CMP-type enterprise bean. If you are using the wsadmin tool, either:

- use the `$AdminApp installInteractive filename server log files` command in order to be prompted for datasources during the installation, or to provide them in a response file.
- specify datasources as an option to the `$AdminApp installserver log files` command. For details on the syntax, see (with wsadmin).

"No valid target is specified in ObjectName<anObject> for module <module_name>" from install

This error can happen in a clustered environment if the target cell, node, server or cluster into which the application is to be installed is incorrectly specified. For example, it can occur if the target is misspelled.

To correct this problem, check the target names against the actual WebSphere Application Server topology and reenter them with corrections.

addNode -includeapps option does not appear to upload all applications to the Deployment Manager

This error can occur when some or all applications on the target node have already been uploaded to the deployment manager. The addNode program detects which applications are already installed and does not upload them again.

To confirm that this is the cause of the problem, use the administrative console to browse the Deployment Manager configuration and see what applications are already installed.

"Timeout!!!" error displays when attempting to install an enterprise application in the administrative console

This error can happen if you attempt to install an enterprise application that has not been deployed.

To correct this problem:

- Open the ear file file_name.ear in AAT and then select **File ->Generate code for deployment...**. This will create a file with a name like Deployed_file_name.ear.
- In the administrative console, install the deployed ear file.

During application installation, the call to EJB deploy throws an exception

When you specify that EJB deploy be run during application installation and if installation fails with the error command line too long, the problem is that the deployment command generated during installation exceeds the character limit for a command line on the Windows platform. This problem occurs only on Windows platforms.

To work around this problem, you can reduce the length of the EAR file name, reduce the length of the JAR file name within the EAR file, reduce the class path or other options specified for deployment, or change the %TEMP% location of the Windows system to make its path shorter.

Troubleshooting testing and first time run problems

Select the problem you are having with testing or the first run of deployed code for WebSphere Application Server:

- The application server will not start, or starts with errors.
- The application will not start, or starts with errors.
- A Web resource, such as a JSP, servlet, HTML file, or image, does not display.
- Cannot access a datasource.
- Cannot access an enterprise bean from a servlet, JSP file, stand-alone program, or other client.
- Cannot access an object hosted by WebSphere Application Server, such as an enterprise bean or connection pool, from a servlet, JSP file, stand-alone program, or other client.
- I have errors and access problems after enabling security.
- I have errors after enabling Secure Sockets Layer (SSL), or SSL-related error messages.
- I have problems with messaging.
- I get errors when trying to send a SOAP request.
- A WebSphere Application Server Client program does not work.

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

The application server or Deployment Manager does not start or starts with errors

If the WebSphere Application Server installation program completes successfully, but the application server does not start, or starts with errors:

- View the log files, which are located by default in *install_dir*\logs*server_name*\SystemErr.log and SystemOut.log for clues.
- If there are several applications deployed on an application server or node, it may take some time to start. Browse the SystemOut.log periodically and look at the most recent updates to see if the server is still starting up. On Unix platforms, the **tail -f *installation_path*/logs/SystemOut.log** is a convenient way to watch the progress of the server.

- Look for any errors or warnings relating to specific resources with the module, such as Web modules, enterprise beans and messaging resources. If you find any, examine the application server configuration file for that resource's configuration settings. For example, in a base (non-distributed) configuration on Windows systems, browse

  ```
  install_dir\config\cells\BaseApplicationServerCell\
        nodes\host_name\servers\server_name\server.xml
  ```

  and examine the xml tags for that resource's properties. Change its initialState value from ″START″ to ″STOP″. Then restart the server as a test to see if the problem is due to this component.
- Look up any error or warning messages in the message reference table by selecting the Quick Reference view and expanding the ″Messages″ heading.
- If the application server is part of a Network Deployment (multiple server) configuration,
  - Ensure that you have followed the steps for adding the application server to the configuration.
  - Ensure that the configuration is synchronized between the deployment manager and the node. If auto synchronization is running, wait until the synchronization has had a chance to complete. If you are using manual synchronization, request a synchronization to each node in the cluster.
  - Before starting an application server:
    1. Start the Deployment Manager process: ***installation_root*/bin/startManager.sh** or ***installation_root*\bin\startManager.bat**.
    2. Complete the one-time step of ″federating″ the node the application server is running on to the Deployment Manager. This has to be done even if there is only one node, and it is the same physical server as the one on which the DeploymentManager is running. This is done by running the addnode *nodename* utility in the *installation_root*/bin directory of the application server's host.
    3. Start the Node Manager process on the nodes hosting the application servers you want to run: ***installation_root*/bin/startNode.sh** or ***installation_root*\bin\startNode.bat**.
- Ensure that the logical name that you have specified to appear on the console for your application server does not contain invalid characters such as: - / \ : * ? ″ < >.
- If you are unable to start the DeploymentManager after an otherwise successful installation:
  - Look in the file *installation_root*/dmgr/logs/SystemErr.log and SystemOut.log for messages.
  - Where was the product installed? This product is not stand-alone, and depends upon some files which are already installed as part of the base. The Network Deployment product should be installed under the WebSphere Application Server root directory of one of the nodes with the base product, at the same level as the base product. For example, if the base product is in /usr/WebSphere/AppServer, the Network Deployment should be installed into a directory like /usr/WebSphere/NetworkDeployment. Installing the product apart from the base product may result in a error running the startManager command similar to: WSVR0102E: An error occurred stopping, null [class com.ibm.ws.cache.ServerCache].

- If you are using Cloudscape and receive an ″ERROR XSDB6: Another instance of Cloudscape may have already booted the database databaseName.″ error starting application server, consult this topic for more information.
- When using a non-root user ID to run application servers, verify that the non-root user has write access to the *WebSphereRoot* `/AppServer/temp` directory
- When using a non-root user ID to run application servers, verify that the JVM has write access to *WebSphereRoot* `/config/plugin-cfg.xml`

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

The application does not start or starts with errors

What kind of error do you see when you start an application?

- java.lang.ClassNotFoundException: <classname>Bean_AdderServiceHome_04f0e027Bean
- ConnectionFac E J2CA0102E: Invalid EJB component: Cannot use an EJB module with version 1.1 using The Relational Resource Adapter
- NMSV0605E: ″A Reference object looked up from the context...″ error when starting an application.
- other Name Server (″NMSV...″) errors.

If none of these errors match the one you see:

- (the log files) of the application server for this application for clues. By default, these files are: `install_dir/logs/ server_name/SystemErr.log` and `SystemOut.log.`
- Look up any error or warning messages in the message reference table by selecting the Quick Reference view and expanding the ″Messages″ heading.

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

java.lang.ClassNotFoundException: <classname>Bean_AdderServiceHome_04f0e027Bean

An exception similar to this happens is you try to start an undeployed application containing enterprise beans, or containing undeployed enterprise bean modules.

enterprise bean modules created in tools like Eclipse or the Application Assembly Tool (AAT) intentionally have incomplete configuration information. Deploying these modules completes the configuration by reading the module's deployment descriptor and completing platform- or installation-dependent settings and adding related classes to the enterprise bean jar file.

To avoid this problem, use one of the following steps:

- Open the undeployed .ear file containing the enterprise bean, or the stand-alone undeployed .EJB .jar file, in the AAT and run the **File -> Generate code for deployment** option. Then uninstall the application or EJB module in the administrative console and install the deployed version created by the AAT, or
- If you are using the **wsadmin $AdminApp install** command, uninstall it and then reinstall using the -EJBDeploy option. Be sure to follow the install command with the **$AdminConfig save** command.

ConnectionFac E J2CA0102E: Invalid EJB component: Cannot use an EJB module with version 1.1 using The Relational Resource Adapter

This error occurs when an enterprise bean developed to the EJB 1.1 specification is deployed with a WebSphere Application Server V5 J2C-compliant data source, which is the default data source. By default, persistent enterprise beans created under WebSphere Application Server V4.0's Application Assembly tool fulfill the EJB 1.1 specification. In order to run on WebSphere Application Server V5, they must be associated with a WebSphere Application Server V4.0-type data source.

To resolve this problem, you must either modify the application's mapping of enterprise beans to associate 1.x Container Managed Persistence (CMP) beans to associate them with a V4.0 data source or delete the existing data source and create a V4.0 data source with the same name.

To modify the application's mapping of enterprise beans, in the WebSphere Application Server administrative console, select the properties for the problem application and use **map resource references to resources** or **Map data sources for all 1.x CMP beans** to switch the data source the enterprise bean uses, then save the configuration and restart the application.

To delete the existing data source and create a V4.0 data source with the same name:

- In the Administrative Console, select **Resources->Manage JDBC Providers->***JDBC_provider_name*->**Data sources**.
- Delete the data source associated with the EJB 1.1 module.
- Select **Resources->Manage JDBC Providers->***JDBC_provider_name*->**Data sources (Version 4)**.
- Create the data source for the EJB 1.1 module.
- Save the configuration and restart the application.

NMSV0605E: ″A Reference object looked up from the context...″ error when starting an application

If the full text of the error is similar to:

```
[7/17/02 15:20:52:093 CDT] 5ae5a5e2 UrlContextHel W NMSV0605E:
A Reference object looked up from the context
     "java:" with the name "comp/PM/WebSphereCMPConnectionFactory"
         was sent to the JNDI Naming Manager
     and an exception resulted. Reference data follows:
     Reference Factory Class Name:
         com.ibm.ws.naming.util.IndirectJndiLookupObjectFactory
     Reference Factory Class Location URLs:
     Reference Class Name: java.lang.Object
     Type: JndiLookupInfo
     Content: JndiLookupInfo: ; jndiName="eis/jdbc/MyDatasource_CMP";
         providerURL=""; initialContextFactory=""
```

then the problem might be that the data source intended to support a CMP enterprise bean has not been correctly associated with the enterprise bean.

To resolve this problem:

- Select the **Use this Data Source in container managed persistence (CMP)** checkbox in the data source's ″General Properties″ panel of the administrative console.

- Ensure that the JNDI Name given in Administrative Console under **Resources -> Manage JDBC Provider -> DataSource -> JNDI Name** for DataSource matches the JNDI Name given for CMP or BMP Resource Bindings at the time of Assembling the application in AAT, or
- Check the JNDI Name for CMP or BMP Resource Bindings specified in the code by J2EE Application Developer. One way to do this is to open the deployed .ear folder in the AAT, and look for the JNDI Name for your Entity Beans under CMP or BMP Resource Bindings.

Web resource (JSP file, servlet, HTML file, image) does not display

What kind of error do you see when you start an application?

- Graphics do not appear on jsp or servlet output.
- SRVE0026E: [Servlet Error]-[Unable to compile class for JSP error on JSP.
- After modifying and saving a JSP, the change does not show up in the browser (the old JSP displays).
- Message similar to ″Message: /jspname.jsp(9,0) Include: Mandatory attribute page missing″ displays when trying to access JSP.
- The Java source generated from a JSP is not retained in the temp directory (only the classfile is found).
- The JSP Batch Compiler fails with the message ″Enterprise Application [application name you typed in] not found.″
- Non-English browser input is garbled.
- Scroll bars do not appear around items in the browser window.

Otherwise, if you are not able to display a resource in your browser follow these steps:

- Verify that your HTTP server is healthy by accessing the URL `http://server_name` from a browser and seeing whether the ″Welcome page″ appears. This indicates whether the HTTP server is up and running, regardless of the state of WebSphere Application Server.
- If the HTTP server ″Welcome page″ does not appear, that is, if you get a browser message such as ″page cannot be displayed″ or something similar, try to diagnose your Web server problem.
- If the HTTP server appears to be functioning, the problem is:
  - The Application Server may not be serving the target resource. To see if this is the case, try accessing the resource directly through the Application Server instead of through the HTTP server.
  - If you cannot access the resource directly through the Application Server:
    - Verify that the URL used to access the resource is correct.
    - If the URL is incorrect and it is created as a link from another JSP file, servlet, or HTML file:
      - After clicking the link, try correcting it by hand in the browser's URL field and reloading, to confirm that the problem is a malformed URL. If this is the problem, correct the URL in the ″from″ HTML file, servlet or jsp file.
    - If the URL appears to be correct, but the resource cannot be accessed directly through the Application Server, verify the health of the hosting Application Server and Web module:

    - View the hosting Application Server and Web module in the administrative console to verify they are up and running.
    - Copy a simple HTML or JSP file (such as `SimpleJsp.jsp` in the WebSphere Application Server directory structure) to your Web module document root, and try to access it. If this works, the problem is with your resource. View the log of your Application Server to find out why your resource cannot be found or served
- If the resource can be accessed directly through the Application Server, but not through an otherwise healthy HTTP server, the problem lies with the HTTP plug-in — the component that communicates between the HTTP server and the WebSphere Application Server.
- If JSP and servlet output is served, but not static resources such as `.html` and image files, see the steps for enabling file serving.
- If some kinds of resources display correctly, but you cannot display a servlet by its class name:
  - Ensure that the servlet is in a directory in the Web module classpath, such as in the `/Web_module_name.war/WEB-INF/classes directory`.
  - Ensure that you specify the full class name of the servlet, including its package name, in the URL.
  - Ensure that `"/servlet"` precedes the class name in the URL. For example, example: if the root context of a Web module is `"myapp"`, and the servlet is `com.mycom.welcomeServlet`, then the URL should read:

    `http://<hostname>/myapp/servlet/com.mycom.welcomeServlet`
  - Ensure that serving servlets by classname is enabled for the hosting Web module by opening the source Web module in the Application Assembly Tool and browse the "serve servlets by classname" setting in the IBM Extensions property page. If necessary, enable this flag and redeploy the Web module.
  - For servlets or other resources served by mapped URLs, the URL is `http://hostname/web module context root/ mappedURL`.

If none of these steps fixes your problem, check to see if the problem has been identified and documented by looking at the (support (hints and tips, technotes, and fixes)). If you do not find your problem listed there contact IBM support.

Diagnosing Web server problems

If you are unable to view the welcome page of your HTTP server, first determine if the server is operating properly.

On Windows systems, look in the Services panel for the service corresponding to your HTTP server, and verify that the state is "Started". If not, start it. If the service does not start, try starting it manually from the command prompt. If you are using IBM HTTP Server, the command is `IHS_install_dir\apache` .

On Unix systems, execute the command `ps -ef | grep httpdserver log files`. There should be several processes running with a name of `"httpd"`. If not, start your HTTP server manually. If you are using IBM HTTP Server, the command is `IHS_install_dir/bin/apachectl start`.

If the HTTP server will not start:

- Examine the HTTP server error log for clues.
- Try restoring the HTTP server to its configuration prior to installing WebSphere Application Server and restarting it. If you are using IBM HTTP Server:

  - rename the file `IHS_install_dir\httpd.conf`.
  - copy the file `httpd.conf.default` to `httpd.conf`.
  - If Apache is running, stop and restart it.
- For the Sun ONE (iPlanet) Web server, the configuration file to restore is `obj.conf` for Sun ONE V4.1 and both `obj.conf` and `magnus.conf` for Sun ONE V6.0 and later.
- For IIS, try removing the WebSphere Application Server plug-in through the IIS administrative GUI.
-

  If restoring the HTTP server default configuration works, manually review the configuration file that has WebSphere Application Server updates to verify directory and file names for WebSphere Application Server files. If you cannot manually correct the configuration, you can uninstall and reinstall WebSphere Application Server to create a clean HTTP configuration file.

  If restoring the default configuration file does not help, contact technical support for the Web server you are using. If you are using IBM HTTP Server with a WebSphere Application Server purchase, support is included - first check online support (hints and tips, technotes, and fixes). If you do not find your problem listed there contact IBM support.

Accessing a web resource through the application server (bypassing the HTTP server)

Starting with WebSphere Application Server version 4.0, the built-in Application Server contained in each Application Server allows you to access Web pages directly, bypassing the HTTP server. It is not recommended to serve a production Web site in this way, but it provides a good diagnostic tool when it is not clear whether a problem resides in the HTTP server, WebSphere Application Server, or the HTTP plug-in.

To access a a Web resource through the Application Server:

- Find out the port of the HTTP service in the target Application Server.
  - In the WebSphere Administrative Console, select **Servers->Manage Application Servers**.
  - Select the target server, then under Additional Properties select **Web Container**.
  - Under the Additional Properties of the Web Container, select **HTTP Transports**. You will see the ports listed for virtual hosts served by the Application Server.
  - There may be more than one port listed. In the default Application Server (server1), for example, 9090 is the port reserved for administrative requests, and 9443 and 9043 are used for SSL-encrypted requests. To simply test the sample "`snoop`" servlet, for example, you would use the default application port 9080, unless it has been changed.
- Using the HTTP transport port number of the Application Server, access the resource from a browser. For example, if the port is 9080, the URL would be `http://hostname:9080\myAppContext\myJSP.jsp`.
- If you are still unable to access the resource, ensure that the HTTP transport port is in the "Host Alias" list:
  1. Select **Application Servers>Your_ApplicationServer>Web Container>HTTP Transports** to check the Default virtual host and the HTTP transport ports used by this Application Server.

2. Select **Environment>Manage Virtual Hosts>default host>Host Aliases** to check if the HTTP transport port exists. Add an entry if necessary. For example, if the HTTP port for your application is server is 9080, add a host alias of *:9082.

HTTP server and Application Server are working separately, but requests are not passing from HTTP server to Application Server

If your HTTP server appears to be functioning correctly, and the Application Server also works on its own, but browser requests sent to the HTTP server for pages are not being served, this indicates a problem in the WebSphere Application Server plug-in.

If this is the case:

- Determine whether the HTTP server is attempting to serve the requested resource itself, rather than forwarding it to the WebSphere Application Server.
  - Browse the HTTP server access log (`IHS install root\logs\access.log` for IBM HTTP Server). It may indicate that it could not find the file in its own document root directory.
  - browse the plug-in log file as described below.
- The file `install_dir/config/plugin-cfg.xml` determines which requests sent to the HTTP server are forwarded to the WebSphere Application Server, and to which Application Server. You may need to refresh this file:
  - In the WebSphere Application Server administrative console, expand the Environment tree control.
  - Select **Update WebSphere Plugin**.
  - Stop and restart the HTTP server and retry the Web request.
- Browse the file `install_dir/logs/http_plugin.log` for clues to the problem. Make sure the timestamps with the most recent Plugin Information stanza, which is printed out when the plug-in is loaded, correspond to the time the Webserver was started.
- Turn on plug-in tracing by setting the `LogLevel` attribute in the `install_dir/config/plugin-cfg.xml` file to `Trace` and reloading the request, then browsing the `install_dir/logs/http_plugin.log` file. You should be able to see the plug-in attempting to match the request URI with the various URI definitions for the routes in the `plugin-cfg.xml`. You should be able to see what rules the plug-in is not matching against and then figure out if you need to add additional ones. If you just recently installed the application you may need to manually regenerate the plug-in configuration in order to pick up the new URIs related to the new application.
- For further details on troubleshooting plug-in-related problems, see the topic "HTTP plug-in component troubleshooting tips" on page 87.

File serving problems (html, images, etc)

If text output appears on your JSP- or servlet-supported Web page, but image files do not:

- Ensure that your files are in the right place: the **document root** directory of your Web application WebSphere Application Server follows the J2EE standard, which means that the document root is the `Web_module_name.war` directory of your deployed Web application. Typically this directory will be found in the `installation_root/installedApps/ nodename/appname.ear` or `installation_root/installedApps/ nodename/ appnameNetwork.ear` directory.

If the files are in a subdirectory of the document root, verify that the reference to the file reflects that. That is, if `invoices.html` is stored in Windows directory `Web_module_name.war\invoices`, then links from other pages in the Web application to display it should read `"invoices\invoices.html"`, not `"invoices.html"`.

- Ensure that your Web application is configured to enable file serving (i.e., display of static resources like image and `.html` files):
  - View the file serving property of the hosting Web module by browsing the source `.war` file in the Application Assembly Tool (AAT). If necessary, update the property and re-deploy the module.
  - Edit the `fileServingEnabledserver log files` property in the deployed Web application `ibm-web-ext.xmi` configuration file, typically found in the `install_root/config/cells/ nodename` or `nodenameNetwork/applications/ application name/deployments/application name/Webmodule name/web-inf` directory.

Graphics do not appear on jsp or servlet output

If text output appears on your JSP- or -servlet-supported Web page, but image files do not:

- Ensure that your graphic files are in the right place: the **document root** directory of your Web application WebSphere Application Server 5 follows the J2EE standard, which means that the document root is the `Web_module_name.war` directory of your deployed Web application. Typically this directory will be found in the `installation_root/installedApps/nodename/ appname.ear` or `installation_root/installedApps/ nodename/appnameNetwork.ear` directory.

  If the graphics files are in a subdirectory of the document root, verify that the reference to the graphic reflects that; e.g., if `banner.gif` is stored in Windows directory `Web_module_name.war/images`, the tag to display it should read: `<img SRC="images/banner.gif">server log files`, not `<img SRC="banner.gif">`.
- Ensure that your Web application is configured to enable file serving (i.e., display of static resources like image and `.html` files).
  - View the file serving property of the hosting Web module by browsing the source `.war` file in the AAT. If necessary, update the property and re-deploy the module. Or
  - Edit the `fileServingEnabledserver log files` property in the deployed Web application `ibm-web-ext.xmi` configuration file, typically found in the `install_root/config/cells/ nodename` or `nodenameNetwork/applications/ application name/deployments/application name/Webmodule name/web-inf` directory.
  - After following one of the above steps:
    - In the administrative console, expand the **Environment** tree control .
    - Click the link **Update WebSphere Plugin**.
    - Stop and restart the HTTP server and retry the Web request.

SRVE0026E: [Servlet Error]-[Unable to compile class for JSP

If this error appears in a browser when trying to access a new or modified .jsp file for the first time, the most likely cause is that the JSP Java source failed (was incorrect) during the javac compilation phase.

To confirm that this is the problem, check the (SystemErr.log) for a compiler error message, such as:

```
C:\WASROOT\temp\ ... test.war\_myJsp.java:14:
Duplicate variable declaration: int myInt was int myInt
     int myInt = 122; String myString = "number is 122";
static int myStaticInt=22; int myInt=121;
                                   ^
```

If this is the problem, fix the problem in the JSP source, save the source and re-request the JSP.

If this error occurs when trying to serve a JSP that was copied from another system where it ran successfully, then there is something different about the new server environment that prevents the JSP from running.

Browse the text of the error for a statement like:

```
Undefined variable or class name: MyClass
```

This error indicates that a supporting class or jar file has not been copied to the target server, or is not on the classpath. To resolve, find the file `MyClass.class`, and place it on the Web module `WEB-INF/classes` directory, or place its containing `.jar` file in the Web module `WEB-INF/lib` directory.

Verify that the URL used to access the resource is correct

- For a JSP file, `html` file, or image file: `http://host_name/Web_module_context_root/ subdir under doc root, if any/filename.ext server log files`. The document root for a web application is the `application_name.WAR` directory of the installed application.
  - For example, to access `myJsp.jsp`, located in `c:\WebSphere\ApplicationServer\installedApps\myEntApp.ear\myWebApp.war\invoices` on `myhost.mydomain.com`, and assuming the context root for the myWebApp Web module is `"myApp"`, the URL would be `http://myhost.mydomain.com/myApp/invoices/myJsp.jsp`.
  - JSP serving is enabled by default. File serving for `html` and image files must be enabled as a property of the Web module, in the Application Assembly Tool, or by setting the `fileServingEnabledserver log files` property to `"true"` in the `ibm-web-ext.xmi`file of the installed Web application and restarting the application.
- For servlets served by class name, the URL is `http://hostname/Web_module_context_root/servlet/packageName.className`.
  - For example, to access myCom.myServlet.class, located in c:\WebSphere\ApplicationServer\installedApps\myEntApp.ear\ myWebApp.war\WEB-INF\classes, and assuming the context root for the myWebApp module is "myApp", the URL would be `http://myhost.mydomain.com/myApp/servlet/myCom.MyServletserver log files`.
- Serving servlets by classname must be enabled as a property of the Web module, and is enabled by default. File serving for html and image files must be enabled as a property of the Web application, in the Application Assembly Tool, or by setting the `fileServingEnabledserver log files` property to `"true"` in the `ibm-web-ext.xmiserver log files` file of the installed Web application and restarting the application.

Correct the URL in the "from" html file, servlet or jsp

An HREF with no leading "/" inherits the calling resource context. For example:

- an HREF in `http://[hostname]/myapp/servlet/MyServlet` to `"ServletB"` resolves to `"http://hostname/myapp/servlet/ServletB"`
- an HREF in `http://[hostname]/myapp/servlet/MyServlet` to `"servlet/ServletB"` resolves to `"http://hostname/myapp/servlet/servlet/ServletB"` (an error)
- an HREF in `http://[hostname]/myapp/servlet/MyServlet` to `"/ServletB"` resolves to `"http://hostname/ServletB"` (an error, if `ServletB` requires the same context root as `MyServlet`)

After modifying and saving a JSP, the change does not show up in the browser (the old JSP displays)

The most likely cause of this error is that the Web application is not configured for servlet reloading, or the reload interval is too high.

To correct this problem, in the Application Assembly Tool, check the **Reloading Enabled** flag and the **Reload Interval** value in the IBM Extensions for the the Web module in question. Turn Reloading on, or if it is already on then set the Reload Interval lower.

Message like "Message: /jspname.jsp(9,0) Include: Mandatory attribute page missing" appears when attempting to browse JSP

The most likely cause of this error is that the JSP file failed during the translation to Java phase. Specifically, a JSPdirective, in this case an Include statement, was incorrect or referred to a file that could not be found.

To correct this problem, fix the problem in the JSP source, save the source and re-request the JSP.

The Java source generated from a JSP is not retained in the temp directory (only the classfile is found)

The most likely cause of this error is that the JSP Processor is not configured to keep generated Java source.

To correct this problem, in the Application Assembly Tool, check the **JSP Attributes** under **Assembly Property Extensions** for the Web module in question. Make sure the attribute **keepgenerated** is there and is set to true. If not, set this attribute and restart the Web application. To see the results of this operation, you will have to delete the classfile from the `temp` directory in order to force the JSP Processor to retranslate the JSP source into Java.

The JSP Batch Compiler fails with the message "Enterprise Application [application name you typed in] not found."

The most likely cause of this error is that the full Enterprise Application path and name, starting with the `.ear` subdirectory that resides in the `install_root\config\cells\ node_nameNetwork\applications` directory is expected as an argument to the JspBatchCompiler tool, not just the display name. For example:

- `"JspBatchCompiler -enterpriseapp.name sampleApp.ear/deployments/sampleApp"` is correct, as opposed to
- `"JspBatchCompiler -enterpriseapp.name sampleApp"`, which is incorrect.

Non-English browser input is garbled

If non-English-character-set browser input is apparently garbled after being read by a servlet or JSP, ensure that the request parameters are encoded according to the expected chararacter set before being read. For example, if the site is Chinese, the target `.jsp` should have a line:

```
req.setCharacterEncoding("gb2312");
```

before any req.getParameter() calls.

Note: This problem especially affects servlets and `jsps` ported from earlier versions of WebSphere Application Server, which converted characters automatically based upon the locale of the WebSphere Application Server.

Scroll bars do not appear around items in the browser window

In some browsers, tree or list type items that extend beyond their allotted windows do not have scroll bars to allow you to see the entire list.

To correct this problem, right click on the browser window and select **Reload** from the pop-up menu.

Cannot access a data source

What kind of database are you trying to access?

- Oracle
- DB2
- SQL Server
- Cloudscape
- Sybase
- My problem was not described under the topic for my database, or might not be DBM specific.

If none of these errors match the one you see:

1. Browse the log files of the application server that contains the application, for clues. By default these files are `install_root/server_name/SystemErr.log` and `SystemOut.log`.
2. Browse the Helper Class property of the data source to verify that it is correct and that it is on the WebSphere Application Server classpath. Mysterious errors or behavior might be the result of a missing or misnamed Helper Class name. If WebSphere Application Server is not able to load the specified class, it uses a default helper class that might not function correctly with your database manager.
3. Verify that the Java Naming and Directory Interface (JNDI) name of the data source matches the name used by the client attempting to access it. If error messages indicate that the problem might be naming-related, such as referring to the **name server** or **naming service**, or including error IDs beginning with **NMSV**, look at the Naming related problems and component topics.
4. Enable tracing for the resource adapter using the trace specification, RRA=all=enabled. Follow the instructions for dumping and browsing the trace output, to narrow the origin of the problem.

If none of these steps fixes your problem, see if the problem has been identified and documented in available online support (hints and tips, technotes, and fixes). If none of the online resources listed in the topic describes your problem, contact IBM support.

What kind of error do you see when you try to access your Oracle-based datasource

- ″DSRA8100E: Unable to get a {0} from the DataSource. Explanation: See the linkedException for more information.″
- Invalid Oracle URL specified
- ″DSRA0080E: An exception was received by the Data Store Adapter. See original exception message: ORA-00600″ when connecting to or using an Oracle data source.
- ″Error while trying to retrieve text for error″ error when connecting to an Oracle data source.
- java.lang.UnsatisfiedLinkError connecting to an Oracle data source.
- java.lang.NullPointerException or ″internal error: oracle.jdbc.oci8.OCIEnv″ connecting to an Oracle data source.
- WSVR0016W: Classpath entry, ${ORACLE_JDBC_DRIVER_PATH}/classes12.zip, in Resource, Oracle JDBC Thin Driver, located at cells/BaseApplicationServerCell/nodes/wasrtp/resources.xml has an invalid variable.

What kind of problem are you having accessing your DB2 database?

- SQL0805N Package ″package name″ was not found.
- SQLException, with ErrorCode -99,999 and SQLState 58004, with java ″StaleConnectionException: COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver] CLI0119E Unexpected system failure. SQLSTATE=58004″ using WAS40-type data source.
- DSRA0023E: The DataSource implementation class ″COM.ibm.db2.jdbc.DB2XADataSource″ could not be found. when trying to access a data source based on a DB2 database.
- SQL0805N Package ″NULLID.SQLLC300″ was not found. SQLSTATE=51002.
- SQL0567N ″DB2ADMIN″ is not a valid authorization ID. SQLSTATE=42602.
- CLI0119E System error. SQLSTATE=58004 - DSRA8100 : Unable to get a XAconnection, or DSRA0011E: Exception: COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver] CLI0119E Unexpected system failure. SQLSTATE=58004.
- COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver][DB2/NT] SQL0911N The current transaction has been rolled back because of a deadlock or timeout. Reason code ″2″. SQLSTATE=40001.
- (Unix)java.sql.SQLException: java.lang.UnsatisfiedLinkError: Can’t find library db2jdbc (libdb2jdbc.a or .so) in java.library.path.
- ″COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource″ could not be found for data source (data_source).

What kind of problem are you having accessing your SQL Server database?

- ERROR CODE: 20001 and SQL STATE: HY000.
- Application fails with message stating ″Cannot find stored procedure...″

What kind of problem are you having accessing your Cloudscape database?

- Unexpected IOException wrapped in SQLException, accessing Cloudscape database.
- ″Select for update″ on one row causes table to become locked, triggering a deadlock condition.
- ″ERROR XSDB6: Another instance of Cloudscape might have already booted the database databaseName.″ error starting appplication server.

Note: Cloudscape errorCodes (2000, 3000, 4000) indicate levels of severity, not specific error conditions. In diagnosing Cloudscape problems, pay attention to the given sqlState value.

What kind of problem are you having accessing your Sybase database?

- SET CHAINED command not allowed within multi-statement transaction.
- ″Sybase Error 7713: Stored Procedure can only be executed in unchained transaction mode″ error.
- ″JZ0XS: The server does not support XA-style transactions. Please verify that the transaction feature is enabled and licensed on this server.″
- A Container Managed Persistence (CMP) enterprise bean is causing exceptions.

What kind of general data access problem do you have?

- ″ObjectNotFoundException″, ″NameNotFoundException″, or other jndi-related error when the client application attempts to use the data source.
- ″IllegalConnectionUseException″
- WTRN0062E: An illegal attempt to enlist multiple one phase capable resources has occurred.
- ConnectionWaitTimeoutException.
- com.ibm.websphere.ce.cm.StaleConnectionException: [IBM][CLI Driver] SQL1013N The database alias name or database name ″NULL″ could not be found. SQLSTATE=42705
- java.sql.SQLException: java.lang.UnsatisfiedLinkError:
- ″J2CA0030E: Method enlist caught java.lang.IllegalStateException″ wrapped in error ″WTRN0063E: An illegal attempt to enlist a one phase capable resource with existing two phase capable resources has occurred″ when attempting to execute a transaction.

DSRA8100E: Unable to get a {0} from the DataSource. Explanation: See the linkedException for more information.

When using oracle thin driver, Oracle throws ″java.sql.SQLException: invalid arguments in call″ if no username or password is specified when getting a connection. If you see this while running WebSphere Application Server, the alias is not set.

To remove the exception, define the alias on the data source.

Invalid Oracle URL specified

This error might be caused by an incorrectly specified URL on the URL property of the target data source.

Examine the URL property for the data source object in the administrative console. For the 8i OCI driver, ensure that **oci8** is used in the URL. For the 9i OCI driver, you can use either **oci8** or **oci**.

Examples of Oracle URLs:

- For the thin driver: jdbc:oracle:thin:@hostname.rchland.ibm.com:1521:IBM
- For the thick (OCI) driver: jdbc:oracle:oci8:@tnsname1

″DSRA0080E: An exception was received by the Data Store Adapter. See original exception message: ORA-00600″ when connecting to or using an Oracle data

source ″DSRA0080E: An exception was received by the Data Store Adapter. See original exception message: ORA-00600″ when connecting to or using an Oracle data source ″DSRA0080E: An exception was received by the Data Store Adapter. See original exception message: ORA-00600″ when connecting to or using an Oracle data source

A possible reason for this exception is that the version of the Oracle JDBC driver being used is older than the Oracle database. It is possible that more than one version of the Oracle JDBC driver has been configured on the WebSphere Application Server.

To confirm that this is the cause of the problem, examine the version of the JDBC driver. Sometimes you can determine the version by looking at the classpath to determine what directory the driver is in.

If you cannot determine the version this way, use the following program to determine the version. Before running the program, set the classpath to the location of your JDBC driver files.

```
import java.sql.*;
     import oracle.jdbc.driver.*;
     class JDBCVersion
     {
         public static void main (String args[])
         throws SQLException
         {
             // Load the Oracle JDBC driver
             DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
             // Get a connection to a database
             Connection conn = DriverManager.getConnection
                 ("jdbc:oracle:thin:@appaloosa:1521:app1","sys","change_on_install");
             // Create Oracle DatabaseMetaData object
             DatabaseMetaData meta = conn.getMetaData();
             // gets driver info:
             System.out.println("JDBC driver version is " + meta.getDriverVersion());
         }
     }
```

If the driver and the database are at different versions, replace the JDBC driver with the correct version. If multiple drivers are configured, remove any that are at the incorrect level.

″Error while trying to retrieve text for error″ error when connecting to an Oracle data source

The most likely cause is that the Oracle 8i OCI driver is being used with an ORACLE_HOME property that is either not set or is set incorrectly.

To correct the error, examine the user profile that WebSphere Application Server is running under to verify that the $ORACLE_HOME environment variable is set correctly.

″java.lang.UnsatisfiedLinkError:″ connecting to an Oracle data source

The problem might be that the environment variable LIBPATH is not set or is set incorrectly, if your data source throws an **UnsatisfiedLinkError**, and the full exception indicates that the problem is related to an Oracle module, as in the following examples.

- Example of invalid LIBPATH for the 8i driver:

```
Exception in thread "main" java.lang.UnsatisfiedLinkError:
/usr/WebSphere/AppServer/java/jre/bin/libocijdbc8.so:  load ENOENT on shared library(s)
/usr/WebSphere/AppServer/java/jre/bin/libocijdbc8.so libclntsh.a
```

- Example of invalid LIBPATH for for the 9i driver:

```
Exception in thread "main" java.lang.UnsatisfiedLinkError:
no ocijdbc9  (libocijdbc9.a or .so) in java.library.path
at java.lang.ClassLoader.loadLibrary(ClassLoader.java(Compiled Code))
at java.lang.Runtime.loadLibrary0(Runtime.java:780)
```

To correct the problem, examine the user profile that WebSphere Application Server is running under to verify that the LIBPATH environment variable includes Oracle libraries. Scan for the file `lobocijdbc8.so` to find the right directory.

java.lang.NullPointerException referencing 8i classes, or ″ internal error: oracle.jdbc.oci8. OCIEnv″ connecting to an Oracle data source

The problem might be that the 9i OCI driver is being used on an AIX 32 bit machine, the LIBPATH is set correctly, but the ORACLE_HOME is not set or is set incorrectly, if you encounter an exception similar to either of the following, when your application attempts to connect to an Oracle data source:

- Exception example for java.lang.NullPointerException:

```
Exception in thread "main" java.lang.NullPointerException
at oracle.jdbc.oci8.OCIDBAccess.check_error(OCIDBAccess.java:1743)
at oracle.jdbc.oci8.OCIEnv.getEnvHandle(OCIEnv.java:69)
at oracle.jdbc.oci8.OCIDBAccess.logon(OCIDBAccess.java:452)
at oracle.jdbc.driver.OracleConnection. <init>(OracleConnection.java:287)
```

- Exception example for java.sql.SQLException:

```
Exception in thread "main" java.sql.SQLException:
internal error: oracle.jdbc.oci8. OCIEnv@568b1d21
at oracle.jdbc.dbaccess.DBError.throwSqlException(DBError.java:184)
at oracle.jdbc.dbaccess.DBError.throwSqlException(DBError.java:226)
at oracle.jdbc.oci8.OCIEnv.getEnvHandle(OCIEnv.java:79)
```

To correct the problem, examine the user profile that WebSphere Application Server is running under to verify that it has the `$ORACLE_HOME` environment variable set correctly, and that the `$LIBPATH` includes `$ORACLE_HOME/lib`.

WSVR0016W: Classpath entry, ${ORACLE_JDBC_DRIVER_PATH}/classes12.zip, in Resource, Oracle JDBC Thin Driver, located at cells/BaseApplicationServerCell/nodes/wasrtp/resources.xml has an invalid variable

This error occurs when no environment variable is defined for the property, ORACLE_JDBC_DRIVER_PATH.

Verify that this is the problem in the administrative console. Go to **Environment > Manage WebSphere Variables** to verify whether the variable `ORACLE_JDBC_DRIVER_PATH` is defined.

To correct the problem, click **New** and define the variable. For example, name : **ORACLE_JDBC_DRIVER_PATH** , value : **c:\oracle\jdbc\lib** Use a value that names the directory in your operating system and directory structure that contains the `classes12.zip` file.

SQL0805N Package ″<package-name>″ was not found

Possible reasons for these exceptions are:

- If the package name is NULLID.SQLLC300, see SQL0805N Package "NULLID.SQLLC300" was not found. SQLSTATE=51002. for the reason.
- You are attempting to use an XA-enabled JDBC driver on a DB2 database that is not XA-ready.

To correct the problem on a DB2/UDB database, run this one-time procedure, using the db2cmd interface while connected to the database in question:

1. **DB2 bind @db2ubind.lst blocking all grant public**
2. **DB2 bind @db2cli.lst blocking all grant public**

The db2ubind.lst and db2cli.lst files are in the bnd directory of your DB2 install_root. Run the commands from that directory.

SQLException, with ErrorCode -99,999 and SQLState 58004, with java "StaleConnectionException: COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver] CLI0119E Unexpected system failure. SQLSTATE=58004", when using WAS40-type data source

An unexpected system failure usually occurs when running in XA mode (two-phase commit). Among the many possible causes are:

- An invalid username or password was provided.
- The database name is incorrect.
- Some DB2 packages are corrupted.

To determine whether you have a username or password problem, look in the db2diag.log file to view the actual error message and SQL code. A message like the following, with an SQLCODE of -1403, indicates an invalid user ID or password:

```
2002-07-26-14.19.32.762905   Instance:db2inst1   Node:000
PID:9086(java)   Appid:*LOCAL.db2inst1.020726191932
XA DTP Support  sqlxa_open   Probe:101
DIA4701E Database "POLICY2" could not be opened for distributed transaction processing.
String Title: XA Interface SQLCA  PID:9086 Node:000
SQLCODE = -1403
```

To resolve these problems:

1. Correct your username and password. If you specify your password on the GUI (for 40 Datasource), ensure that the username and password you specify on the bean are correct. The username and password you specify on the bean overwrite whatever you specify when creating the data source.
2. Use the correct database name.
3. Rebind the packages (in the bnd directory) as follows:

   ```
   db2connect to dbname
   c:\SQLLIB\bnd>DB2 bind @db2ubind.lst blocking all grant public
   c:\SQLLIB\bnd>DB2 bind @db2cli.lst blocking all grant public
   ```

4. Ensure that the file \WebSphere\AppServer\properties\wsj2cdpm.properties has the right userid and password.

Error message "java.lang.reflect.InvocationTargetException: com.ibm.ws.exception.WsException: DSRA0023E: The DataSource implementation class "COM.ibm.db2.jdbc.DB2XADataSource" could not be found." when trying to access a DB2 database

One possible reason for this exception is that a user is attempting to use a JDBC 2.0 DataSource, but DB2 is not JDBC 2.0 enabled. This frequently happens with new installations of DB2 because DB2 provides separate drivers for JDBC 1.X and 2.0, with the same physical file name. By default, the JDBC 1.X driver is on the classpath.

To confirm that this is the problem:

- On Windows systems, look for the file **inuse** in the `java12` directory in your DB2 install_root. If it is not there, you are using the JDBC 1.x driver.
- On UNIX systems, check the classpath for your data source. If it does not point to the `db2java.zip` file in the `java12` directory, you are using the JDBC 1.x driver.

To correct this problem:

- On Windows systems, stop DB2. Run `usejdbc2.bat` from the `java12` directory in your DB2 install_root. Run this from a command line to verify that it completes successfully.
- On UNIX systems, change the classpath for your data source to point to the `db2java.zip` file in the `java12` directory of your DB2 install_root.

SQL0805N Package ″NULLID.SQLLC300″ was not found. SQLSTATE=51002

Some possible causes of this error are:

- The underlying database was dropped and recreated.
- DB2 was ugpraded, and its packages are not rebound correctly.

To resolve this problem, rebind the DB2 packages by running the the **db2cli.lst** script found in the `bnd` directory. For example:**db2>@db2cli.lst**.

SQL0567N ″DB2ADMIN ″ is not a valid authorization ID. SQLSTATE=42602

If you encounter this error when attempting to access a DB2/UDB data source:

1. Verify that your username and password in the data source properties in the admin console, are correct.
2. Ensure that the userid and password do not contain blank characters (before, in between, or after).

CLI0119E System error. SQLSTATE=58004 - DSRA8100 : Unable to get a XAconnection or DSRA0011E: Exception: COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver] CLI0119E Unexpected system failure. SQLSTATE=5800

If you encounter this error when attempting to access a DB2/UDB data source:

1. Check your username and password ″custom properties″ in the data source properties page in the admin console. Ensure that they are correct.
2. Ensure the userid and password do not containe any blank characters (before, in between, or after).
3. Check that the `WAS.policy` file exists for the application, for example, `D:\WebSphere\AppServer\installedApps\markSection.ear\META-INF\was.policy`.
4. View the entire exception listing for an underlying SQL error, and look it up using the DBM vendor message reference.

If you encounter this error while running DB2 on Red Hat Linux, the `max queues system wide` parameter is too low to allow DB2 to acquire the necessary resources to complete the transaction. When this is the problem, exception DSRA8100E can be preceded by exceptions J2CA0046E and DSRA0010E.

To correct this problem, edit the file `/proc/sys/kernal/msgmni` to increase the value of the `max queues system wide` parameter to a value greater than 128.

COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver][DB2/NT] SQL0911N The current transaction has been rolled back because of a deadlock or timeout. Reason code ″2″. SQLSTATE=40001

This is probably an application-caused DB2 deadlock, particularly if you see an error similar to the following when accessing a DB2 data source:

```
ERROR CODE: -911
COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver][DB2/NT] SQL0911N
The current transaction has been rolled back because of a deadlock or timeout.
Reason code "2".  SQLSTATE=40001
```

To diagnose the problem:

1. Execute these DB2 commands:
   a. **db2 update monitor switches using LOCK ON**
   b. **db2 get snapshot for LOCKS on dbName > *directory_name*\lock_snapshot.log**

   The `directory_name\lock_snapshot.log` now has the DB2 lock information.
2. Turn off the lock monitor by executing: **db2 update monitor switches using LOCK OFF**

To verify that you have a deadlock:

1. Look for an application handle that has a lock-wait status, then look for the **ID of agent holding lock** to verify the ID of the agent.
2. Go to that handle to verify it has a lock-wait status, and the ID of the agent holding the lock for it. If it is the same agent ID as the previous one, then you know that you have a circular lock (deadlock).

To resolve the problem:

1. Examine your application and use a less restrictive isolation level if no concurrency access is needed.
2. Use caution when moving to a lesser **accessIntent**, which can result in data integrity problems.
3. For DB2/UDB Version 7.2 and earlier releases, you can set the DB2_RR_TO_RS flag from the DB2 command line window to eliminate unnecessary deadlocks, such as when the accessIntent defined on the bean method is too restrictive, for example, PessmisticUpdate. The DB@_RR_TO_RS setting has two impacts:
   - If RR is your chosen isolation level, it is effectively downgraded to RS.
   - If you choose another isolation level, and the DB2_RR_TO_RS setting is on, a scan skips over rows that have been deleted but not committed, even though the row might qualify for the scan. The skipping behavior affects the RR, Read Stability (RS), and Cursor Stability (CS) isolation levels.

   For example, consider the scenario where transaction A deletes the row with column1=10 and transaction B does a scan where column1>8 and column1<12. With DB2_RR_TO_RS off, transaction B waits for transaction A to commit or

rollback. If transaction A rolls back, the row with column1=10 is included in the result set of the transaction B query. With DB2_RR_TO_RS on, transaction B does not wait for transaction A to commit or rollback. Transaction B immediately receives query results that do not include the deleted row. Setting DB2_RR_TO_RS effectively changes locking behavior, thus avoiding deadlocks.

java.sql.SQLException: java.lang.UnsatisfiedLinkError: Can't find library db2jdbc (libdb2jdbc.a or .so) in java.library.path

"COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource" could not be found for data source ([data-source-name])"

This error usually occurs when the classpath of the DB2 JDBC driver is set correctly to `${DB2_JDBC_DRIVER_PATH}/db2java.zip` but the environment variable DB2_JDBC_DRIVER_PATH is not set.

Confirm that this is the problem on the **Manage WebSphere Variables** panel. Select **Environment** to verify that there is no entry for the variable DB2_JDBC_DRIVER_PATH.

To correct this problem, add the variable DB2_JDBC_DRIVER_PATH with **value** equal to the directory path containing the `db2java.zip` file.

ERROR CODE: 20001 and SQL STATE: HY000 accessing SQLServer database

The problem might be that the Distributed Transaction Coordinator service is not started, if you see an error similar to the following when attempting to access an SQL Server database:

```
ERROR CODE: 20001
SQL STATE:  HY000
java.sql.SQLException: [Microsoft][SQLServer JDBC Driver][SQLServer]xa_open (0) returns -3
at com.microsoft.jdbc.base.BaseExceptions.createException(Unknown Source) ...
at com.microsoft.jdbcx.sqlserver.SQLServerDataSource.getXAConnection(Unknown Source) ...
```

To confirm that this is the problem, in the Windows **Control Panel > Services** (or the **Control Panel > Administrative Tools > Services**) window, verify whether the service **Distributed Transaction Coordinator** or **DTC** is started. If not, it might be the cause of the problem.

To resolve this problem, start the Distributed Transaction Coordinator service.

Application fails with message stating "Cannot find stored procedure..." accessing an SQLServer database

One possible cause for this error is that the Stored Procedures for JTA feature was not installed on the Microsoft SQL Server.

To correct the problem, repeat the installation for the Stored Procedures for JTA feature, according to the ConnectJDBC installation guide.

Unexpected IOException wrapped in SQLException, accessing Cloudscape database

This problem can occur because Cloudscape databases use a large number of files. Some operating systems, such as Sun Solaris, limit the number of files an application can open at one time. If the default is a low number, such as 64, you can get this exception.

If your operating system lets you configure the number of file descriptors, you can correct the problem by setting the number to a high value, such as 1024.

"select for update" causes table lock and deadlock when accessing Cloudscape

If a **select for update** operation on one row locks the entire table, which creates a deadlock condition, the cause can be that you have not defined indexes on that table. Lack of an index on the columns you use in the **where** clause can cause Cloudscape to create a table lock rather than a row level lock.

To resolve this problem, create an index on the affected table.

ERROR XSDB6: Another instance of Cloudscape may have already booted the database "database"

This problem is caused by the fact that Cloudscape 5.0.X allows only one JVM to access the database instance at a time.

To resolve this problem:

1. Ensure that you do not have other JDBC client programs, such as **ij** or **cview** running on that database instance, when WebSphere Application Server is running.
2. Ensure that you do not use the same instance of the database for more than one data source.

"SET CHAINED command not allowed within multi-statement transaction." exception accessing Sybase

The reason for the error might be that:

- You are attempting to set autocommit to **on** in a 2 phase transaction, which is not permitted.
- You have an incorrectly configured DSM license.

To verify that one of these problems is the cause, look for an error similar to the following when attempting to use a Sybase data source:

```
[7/30/02 9:44:06:191 CDT] 3ab306e5 SybaseDataSto d The sqlState is: ZZZZZ
[7/30/02 9:44:06:191 CDT] 3ab306e5 GenericDataSt > findMappingClass for exception
com.sybase.jdbc2.jdbc.SybSQLException: SET CHAINED command not allowed
within multi-statement transaction.
```

To resolve the problem of attempting to set autocommit **on** in a 2 phase transaction, perform either of these actions:

- Do not modify the autocommit value.
- Use a single phase data source.

To resolve the problem of an incorrectly configured DSM license, correct the **Adaptive Server Enterprise DTM option authorization code**. This is the license code supplied by your Sybase dealer. You can enter it into the `license.dat` file in the Sybase directory structure.

"Sybase Error 7713: Stored Procedure can only be executed in unchained transaction mode" error

This error occurs when either:

- The JDBC attempts to put the connection in **autocommit(true)** mode.
- A stored procedure is not created in a compatible mode.

To fix the **autocommit(true)** mode problem, let the application change the connection to chained mode using **Connection.setAutoCommit(false)**, or use a **set chained on** language command.

To resolve the stored procedure problem, use this command, **sp_procxmode** *procedure_name*, "**anymode**".

"JZ0XS: The server does not support XA-style transactions. Please verify that the transaction feature is enabled and licensed on this server."

This error occurs when XA-style transactions are attempted on a server that does not have Distributed Transaction Management (DTM) installed.

To correct this problem, use the instructions in the Sybase Manual titled: *Using Adaptive Server Distributed Transaction Management Features* to enable Distributed Transaction Management (DTM). The main steps in this procedure are:

1. Install the DTM option.
2. Check the `license.dat` file to verify that the DTM option was installed.
3. Restart the license manager.
4. Enable DTM in ISQL.
5. Restart the ASE service.

A Container Managed Persistence (CMP) enterprise bean is causing exceptions

This error is caused by improper use of reserved words. Reserved words cannot be used as column names.

To correct this problem, rename the variable to remove the reserved word. You can find a list of reserved words in the *Sybase Adaptive Server Enterprise Reference Manual; Volume 1: Building Blocks*, Chapter 4. This manual is available online at: http://manuals.sybase.com/onlinebooks/group-as/asg1250e/refman.

IllegalConnectionUseException

One possible reason for this error is that a connection obtained from a WAS40DataSource is being used on more than one thread. This is a violation of the J2EE 1.3 programming model, and an exception is generated when it is detected on the server. This problem occurs for users accessing a data source through servlets or Bean Managed Persistence (BMP) type enterprise beans.

To confirm that this is the problem, examine the code for sharing of connections. Code can inadvertently cause sharing by not following the programming model recommendations, for example by storing a connection in an instance variable in a servlet, which can cause the connection to be used on multiple threads at the same time.

WTRN0062E: An illegal attempt to enlist multiple one phase capable resources has occurred

Possible causes of this error include:

- An attempt to share a single phase connection, when each **getConnection** method has different connection properties; such as the AccessIntent. This causes the connection to be created as non-shareable.
- An attempt to have more than one unshareable connection participate in a global transaction, when the data source is not an XA resource.
- An attempt to have a one phase resource participate in a global transaction while an XA resource or another one phase resource has already participated in this global transaction.
  - Within the scope of a global transaction you try to get a connection more than once and at least one of the resource-refs you are using specifies that the connection is unshareable, and the data source is not configured to support 2 Phase Commit transactions. It does not support an XAResource. If you do not use a resource-ref, you default to unshareable connections.
  - Within the scope of a global transaction you try to get a connection more than once and at least one of the resource-refs you are using specifies that the connection is shareable and the data source is not configured to support two phase Commit transactions. That is, it does not support an XAResource. In addition, even though you specify that connections should be shareable, each getConnection request is made with different connection properties (such as IsolationLevel or AccessIntent). In this case, the connections are not shareable, and multiple connections are handed back.
  - Multiple components (Servlets, Session Beans, BMP Entity Beans, or CMP Entity Beans) are accessed within a global transaction. All use the same DataSource, all specify shareable connections on their resource-refs, and you expect them to all share the same connection. If the properties are different, as stated above, you get multiple connections. AccessIntent settings on CMP beans change their properties. To share a connection, the AccessIntent setting must be the same. For more information about CMP beans sharing a connection with non-CMP components, see the *Data access application programming interface support* and *Example: Accessing data using IBM extended APIs to share connections between container-managed and bean-managed persistence beans* topics in the DataAccess section of the InfoCenter.

To correct this error:

- Check what your client code passes in with its **getConnection** requests, to ensure they are consistent with each other.
- Check the connection sharing scope from the resource binding, using the AAT.
  - If you are running an unshareable connection scope, ensure that your data source is an XA data source.
  - If you are running a shareable connection scope, ensure that all connection properties, including AccessIntent and other properties (such as userid), are sharable.
- Check the JDBC provider **implementation** class from the **Manage JDBC resource** panel of the administrative console to ensure that it is a class that supports XA-type transactions.

ConnectionWaitTimeoutException accessing a data source or resource adapter

If your application receives a com.ibm.websphere.ce.cm.ConnectionWaitTimeoutException or com.ibm.websphere.ce.j2c.ConnectionWaitTimeoutException when attempting to access a WebSphere Application Server data source or JCA-compliant resource adapter, respectively, some possible causes are:

- The maximum number of connections for a given pool is set too low. The demand for concurrent use of connections is greater then the configured maximum for the connection pool. One indication that this is the problem is that you receive these exceptions regularly, but your CPU utilization is not high. This indicates that there are too few connections available to keep the threads in the server busy.
- Connection Wait Time is set too low. Current demand for connections is high enough such that sometimes there is not an available connection for short periods of time. If your connection wait timeout value is too low, you may timeout shortly before a user returns a connection back to the pool. Adjusting the connection wait time may give you some relief. One indication that this is the problem is that you are using near the maximum number of connections for an extended period and receiving this error regularly.
- You are not closing some connections or are returning connections back to the pool at a very slow rate. This can easily happen when using unshareable connections, when you forget to close them, or you close them long after you are finished using them, thus keeping the connection from being returned to the pool for reuse. The pool soon becomes empty and all applications get ConnectionWaitTimeoutExceptions. One indication that this is the problem is that the connection pool has become starved and you receive this error on most requests.
- You are driving more load than the server or backend system have resources to handle. In this case you must determine which resources you need more of and upgrade configurations or hardware to address the need. One indication that this is the problem is that the application or database server CPU is nearly 100% busy.

To correct these problems, modify an application to use fewer connections or properly close the connections, change the pool settings of MaxConnections or ConnnectionWaitTimeout, or adjust resources and their configuration.

com.ibm.websphere.ce.cm.StaleConnectionException: [IBM][CLI Driver] SQL1013N The database alias name or database name ″NULL″ could not be found. SQLSTATE=42705

This error occurs when a data source has been defined but the databaseName attribute and corresponding value have not been added to the ″custom properties″.

To add the the databaseName property:

1. Expand the **Resources->Manage JDBC Providers** link in the administrative console.
2. Select the JDBC Provider which supports the problem data source.
3. Select **Data Sources** and then select the problem data source.
4. Under **additional properties** select **Custom Properties**.
5. Select the **databaseName** property, or add one if it does not exist, and enter the actual database name as the value.
6. Click **Apply** or **OK**, and then **Save** from the action bar.
7. Try to access the data source again.

java.sql.SQLException: java.lang.UnsatisfiedLinkError:

This error indicates that the directory containing the binary libraries which support a database are not included in the LIBPATH environment variable for the environment in which the WebSphere Application Server is started.

The path containing the DBM vendor's libraries vary by dbm. One way to find them is by scanning the for missing library specified in the error message. Then the LIBPATH variable can be corrected to include the missing directory, either in the `.profile` of the account from which WebSphere Application Server is executed, or by adding a statement in a `.sh` file which then executes the "`startServer`" program.

"J2CA0030E: Method enlist caught java.lang.IllegalStateException" wrapped in error "WTRN0063E: An illegal attempt to enlist a one phase capable resource with existing two phase capable resources has occurred" when attempting to execute a transaction.

This error can occur when Last Participant Support (LPS) is missing or disabled. LPS allows a one-phase capable resource and a two-phase capable resource to be enlisted within the same transaction.

LPS is only available if the following are true:

- WebSphere Application Server Programming Model Extensions (PME), which is included in the Application Server Enterprise product) is installed.
- The option "Additional Enterprise Extensions" was enabled when PME was installed. If you perform a typical installation, this is be enabled by default. If you perform a custom installation, you have the option to disable this function, which would disable LPS.
- The application enlisting the one phase resource has been deployed with the **Accept heuristic hazard** option enabled. This is done through the Application Assembly Tool. To enable this option in the Application Assembly Tool:
  1. Load the `EAR` file into the Application Assembly Tool.
  2. If the `EAR` file is actually a JTEE1.2 `EAR` then it must be upgraded to a JTEE1.3 `EAR` by selecting `File-> Convert EARserver log files` from the Application Assembly Tool.
  3. Select the `EAR` file in the left-hand panel of the Application Assembly Tool.
  4. Select the **WAS Enterprise** tab in bottom right-hand window panel of the Application Assembly Tool.
  5. Ensure that the **Accept heuristic hazard** option is selected.
  6. Save the `EAR` file.

Cannot access an enterprise bean from a servlet, JSP file, stand-alone program, or other client

What kind of error are you seeing?

- javax.naming.NameNotFoundException: Name name not found in context "local" message when access is attempted
- BeanNotReentrantException is thrown
- CSITransactionRolledbackException / TransactionRolledbackException is thrown
- Call fails, Stack trace beginning EJSContainer E Bean method threw exception [exception_name] found in JVM log file.

- Call fails, ObjectNotFoundException or ObjectNotFoundLocalException when accessing stateful session EJB found in JVM log file.
- Attempt to start CMP EJB module fails with javax.naming.NameNotFoundException: dataSourceName
- Transaction [tran ID] has timed out after 120 seconds error accessing EJB.
- Symptom: CNTR0001W: A Stateful SessionBean could not be passivated
- Symptom: org.omg.CORBA.BAD_PARAM: Servant is not of the expected type. minor code: 4942F21E completed: No returned to client program when attempting to execute an EJB method

If the client is remote to the enterprise bean, which means, running in a different application server or as a stand-alone client, browse thelogs of the application server hosting the enterprise bean as well as log files of the client.

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, perform these steps:

1. If the problem appears to be Name-Service related, which means that you see a NameNotFoundException, or a message ID beginning with NMSV, see these topics for more information:
   - Cannot access an object hosted by WebSphere Application Server from a servlet, JSP, or other client
   - "Naming Services component troubleshooting tips"
2. Check to see if the problem has been identified and documented using the links in Chapter 12, "Diagnosing and fixing problems: Resources for learning," on page 209.

If you still cannot fix your problem, contact IBM support for further assistance.

ObjectNotFoundException or ObjectNotFoundLocalException when accessing stateful session EJB

A possible cause of this problem is that the stateful session bean timed out and was removed by the container. This event must be coded for, according to the EJB 2.0 specification available at http://java.sun.com/products/ejb/docs.html, section 7.6.2, Dealing with exceptions).

Stack trace beginning "EJSContainer E Bean method threw exception [exception_name]" found in JVM log file

If the "exception name" indicates an exception thrown by an IBM class, that is it begins "com.ibm...", then search for the exception name within the WebSphere Application Server Version 5 InfoCenter, and in the online help as described below. If "exception name" indicates an exception thrown by your application, contact the application developer to determine what might have caused it.

javax.naming.NameNotFoundException: Name name not found in context "local"

A possible reason for this is exception is that the enterprise bean is not local (not running in the same Java Virtual Machine [JVM] or Application Server) to the client JSP, servlet, Java application, or other enterprise bean, yet the call is to one of the enterprise bean's "local" interface methods. If access worked in a development environment but not when deployed to WebSphere Application Server, for example, it could be that the enterprise bean and its client were in the same JVM in development, but after deployment they are in separate processes.

To resolve this problem, contact the developer of the enterprise bean and determine whether the client call is to a method in the enterprise bean's local interface. If so, have the client code changed to call a remote interface method, or promote the local method into the remote interface.

References to enterprise beans with local interfaces are bound in a name space local to the server process with the URL scheme of `local:`. To obtain a dump of a server `local:` name space, use the name space dump utility described in the article "Name space dump utility for java: and local: name spaces "

BeanNotReentrantException is thrown

This problem can be caused by client code (typically a servlet or JSP) attempting to call the same stateful SessionBean from two different client threads. This situation often arises when the an application stores the reference to the stateful session bean in a static variable, uses a global (static) JSP variable to refer to the stateful SessionBean reference, or stores the stateful SessionBean reference in the HTTP session object and then has the client browser issue a new request to the servlet or JSP before the previous request has completed.

To resolve this problem, ask the developer of the client code to review their code for these conditions.

CSITransactionRolledbackException / TransactionRolledbackException is thrown

These are high-level exceptions thrown by an enterprise bean's container, and indicate that an enterprise bean call could not be successfully completed. When this exception is thrown, browse the JVM logs to determine the underlying cause.

Some possible causes are:

- The enterprise bean may be throwing an exception that was not declared as part of its method signature. The container is required to roll back the transaction in this case. Common causes of this situation are where the enterprise bean or code that it calls throws a NullPointerException, ArrayIndexOutOfBoundsException, or other Java "runtime" exception, or where a BMP bean encounters a JDBC error. The resolution is to investigate the enterprise bean code and resolve the underlying exception, or to add the exception to the problem method's signature.
- A transaction may have attempted to do additional work after being placed in a "`Marked Rollback`", "`RollingBack`", or "`RolledBack`" state. Transactions cannot continue to do work after they have been set to one of these states. Often this occurs because the transaction has timed out which, in turn, often occurs because of a database deadlock. The resolution is to work with the application's database managements tools or administrator to determine whether database transactions called by the enterprise bean are timing out.
- A transaction may fail on commit due to "dangling work". This could be due to "local" transactions. The local transaction encountered some "dangling work" during commit. The default "action" for local transactions when they encounter an "Unresolved Action" is to "rollback". This can be adjusted to "commit" in the Application Assembly Tool. In the AAT, open the enterprise bean `.jar` file (or the `EAR` file containing the enterprise bean) and select the "Session Beans" or "Entity Beans" object in the component tree on the left. The "Unresolved Action" property is on the "IBM Extensions" tab of the container properties.

Attempt to start EJB module fails with ″javax.naming.NameNotFoundException dataSourceName_CMP″exception

The possible causes of this problem are:

- When the DataSource resource was configured, Container Managed Persistence was not selected.
  - To confirm that this is the problem, in the administrative console, browse the properties of the data source given in the NameNotFoundException. On the Configuration panel, look for a checkbox labeled **Container Managed Persistence**.
  - To correct this problem, select checkbox for **Container Managed Persistence**.
- If Container Managed Persistence is selected, it is possible that the CMP DataSource could not be bound into the namespace.
  - Look for additional naming warnings or errors in the status bar, and in the hosting application server's (JVM logs). Check any further naming-exception problems that you find by looking at the topic Cannot access an object hosted by WebSphere Application Server (enterprise bean, connection pool, etc) from a servlet, JSP, stand-alone program , or other client.

Transaction [tran ID] has timed out after 120 seconds accessing EJB

This error can happen when a client executes a transaction on a CMP or BMP enterprise bean.

- The default timeout value for enterprise bean transactions is 120 seconds. After this time, the transaction times out and the connection is closed.
- If the transaction legitimately takes longer than the specified timeout period, go to **Manage Application Servers ->** *server_name*, select the **Transaction Service properties** page, and look at the property **Total transaction lifetime timeout**. Increase this value if necessary and save the configuration.

Symptom:CNTR0001W: A Stateful SessionBean could not be passivated

This error can occur when a Connection Object being used in the bean has not been closed or nulled out.

To confirm that this is the problem, look for an exception stack in the (JVM log) for the EJB Container which hosts the enterprise bean, which looks similar to:

```
[time EDT] <ThreadID> StatefulPassi W CNTR0001W: A Stateful SessionBean could not
be passivated: StatefulBeanO(BeanId(XXX#YYY.jar#ZZZZ, <ThreadID>),
state = PASSIVATING) com.ibm.ejs.container.passivator.StatefulPassivator@<ThreadID>
java.io.NotSerializableException: com.ibm.ws.rsadapter.jdbc.WSJdbcConnection
 at java.io.ObjectOutputStream.outputObject((Compiled Code))
 at java.io.ObjectOutputStream.writeObject(ObjectOutputStream.java(Compiled Code))
 at java.io.ObjectOutputStream.outputClassFields((Compiled Code))
 at java.io.ObjectOutputStream.defaultWriteObject((Compiled Code))
 at java.io.ObjectOutputStream.outputObject((Compiled Code))
 at java.io.ObjectOutputStream.writeObject(ObjectOutputStream.java(Compiled Code))
 at com.ibm.ejs.container.passivator.StatefulPassivator.passivate((Compiled Code))
 at com.ibm.ejs.container.StatefulBeanO.passivate((Compiled Code)
 at com.ibm.ejs.container.activator.StatefulASActivationStrategy.atUnitOfWorkEnd
          ((Compiled Code))
 at com.ibm.ejs.container.activator.Activator.unitOfWorkEnd((Compiled Code))
 at com.ibm.ejs.container.ContainerAS.afterCompletion((Compiled Code)
```

where XXX,YYY,ZZZ is the Bean's name, and <ThreadID> is the thread ID for that run.

To correct this problem, the application must close all connections and set the reference to null for all connections. Typically this is done in the `ejbPassivate()` method of the bean. See the enterprise bean specification mandating this requirement, specifically section 7.4 in the EJB specification version 2.0. Also, note that the bean must be coded to reacquire these connections when the bean is reactivated. Otherwise, there will be NullPointerExceptions when the application tries to reuse the connections.

Symptom: org.omg.CORBA.BAD_PARAM: Servant is not of the expected type. minor code: 4942F21E completed: No

This error can be returned to a client program when the program attempts to execute an EJB method.

Typically this is caused by a mismatch between the interface definition and implementation of the client and server installations, respectively.

Another possible cause is when an application server is set up to use a SINGLE class loading scheme. If an application is uninstalled while the application server remains active, the classes of the uninstalled application are still class-loaded in the application server. If you change the application, and redeploy and reinstall it on the application server, the previously loaded classes are now back level. The back level classes cause a code version mismatch between the client and the server.

To correct this problem:

1. Change the application server class loading scheme to MULTIPLEserver log files.
2. Stop and restart the application server and try the operation again.
3. Make sure the client and server code version are the same.

Cannot look up an object hosted by WebSphere Application Server from a servlet, JSP file, or other client

To resolve problems encountered when a servlet, JSP file, stand-alone application or other client attempts to access an enterprise bean, ConnectionPool, or other named object hosted by WebSphere Application Server, you must first verify that the target server can be accessed from the client:

- From a command prompt on the client's server, enter "`ping server_name`" and verify connectivity.
- Use the WebSphere Application Server administrative console to verify that the target resource's application server and, if applicable, EJB module or Web module, is started.

Continue only if there is no problem with connectivity and the target resource appears to be running.

What kind of error are you seeing?

- NameNotFoundException from JNDI lookup operation
- CannotInstantiateObjectException from JNDI lookup operation
- Message NMSV0610I appears in the server's log file, indicating that some Naming exception has occurred
- OperationNotSupportedException from JNDI Context operation.
- "WSVR0046E: Failed to bind" error, with Original exception: "org.omg.CosNaming.NamingContextPackage.AlreadyBound".

- ConfigurationException from ″new InitialContext″ operation or from a JNDI Context operation with a URL name.
- ServiceUnavailableException from ″new InitialContext″ operation.
- CommunicationException thrown from a ″new InitialContext″ operation.
- NMSV0605E: A Reference object looked up from the context...

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

NameNotFoundException from JNDI lookup operation

If you encounter this exception in trying to access an enterprise bean, data source, messaging resource, or other resource:

- Browse the properties of the target object in the administrative console, and verify that the jndi name it specifies matches the JNDI name the client is using.
- If you are looking up an object that resides on a server different from the one from which the initial context was obtained, you must use the fully qualified name.
  - If access is from another server object such as a servlet accessing an enterprise bean and you are using the default context, not specifying the fully qualified JNDI name, you may get this error if the object is being hosted on a different server.
  - If access is from a stand-alone client, it may be that the object you are attempting access is on a server different from the server from which you obtained the initial context.

To correct this problem, use the fully-qualified JNDIname:

- If the object is in a single server: `cell/nodes/ nodeName/servers/serverName/jndiName`. Objects are not supported in this release.
- If the object is on a server cluster: `cell/clusters/ clusterName/jndiName`.

CannotInstantiateObjectException from JNDI lookup operation

If you encounter this exception in trying to access an enterprise bean, data source, messaging resource, or other resource, possible causes include:

- A serialized Java object is being looked up, but the necessary classes required to deserialize it are not in the runtime environment.
- A Reference object is being looked up, and the associated factory used to process it as part of the lookup processing is failing.

To determine the precise cause of the problem:

- Look in the logs of the server hosting the target resource. Look for exceptions immediately preceding the CannotInstantiateObjectException. If it is a java.lang.NoClassDefFoundError or java.lang.ClassNotFoundException, make sure the class referenced in the error message can be located by the class loader.
- Print out the stack trace for the root cause and look for the factory class. It will be called by `javax.naming.NamingManager.getObjectInstance()`. The reason for the failure will depend on the factory implementation, and may require you to contact the developer of the factory class.

Message NMSV0610I appears in the server's log file, indicating that some Naming exception has occurred

This error is informational only and is provided in case the exception is related to an actual problem. Most of the time, it is not. If it is, the log file should contain adjacent entries to provide context.

- If no problems are being experienced, ignore this message. Also ignore the message if the problem you are experiencing does not appear to be related to the exception being reported and if there are no other adjacent error messages in the log.
- If a problem is being experienced, look in the log for underlying error messages.
- The information provided in message NMSV0610I can provide valuable debug data for other adjacent error messages posted in response to the Naming exception that occurred.

OperationNotSupportedException from JNDI Context operation

This error has two possible causes:

- An update operation, such as a bind, is being performed with a name that starts with "`java:comp/env`". This context and its subcontexts are read-only contexts.
- A Context bind or rebind operation of a non-CORBA object is being performed on a remote name space that does not belong to WebSphere Application Server. Only CORBA objects can be bound to these CosNaming name spaces.

To determine which of these errors is causing the problem, check the full exception message.

WSVR0046E: Failed to bind, ejb/jndiName: ejb/jndiName. Original exception : org.omg.CosNaming.NamingContextPackage.AlreadyBound

This error occurs two enterprise bean server applications were installed on the same server such that a binding name conflict occurred. That is, a jndiName value is the same in the two applications' deployment descriptors. The error will surface during server startup when the second application using that jndiName value is started.

To verify that this is the problem, examine the deployment descriptors for all enterprise bean server applications running in the server in search for a jndiName that is specified in more than one enterprise bean application.

To correct the problem, change any duplicate jndiName values to ensure that each enterprise bean in the server process is bound with a different name.

ConfigurationException from "new InitialContext" operation or from a JNDI Context operation with a URL name

If you are attempting to obtain an initial JNDI context, a configuration exception can occur because an invalid JNDI property value was passed to the InitialContext constructor. This includes JNDI properties set in the System properties or in some jndi.properties file visible to the class loader in effect. A malformed provider URL is the most likely property to be incorrect. If the JNDI client is being run as a thin client such that the `CLASSPATH` is set to include all of the individual jar files required, make sure the `.jar` file containing the properties file `com/ibm/websphere/naming/jndiprovider.properties` is in the `CLASSPATH`.

If the exception is occurring from a JNDI Context call with a name in the form of a URL, the current JNDI configuration may not be set up properly so that the required factory class name cannot be determined, or the factory may not be

visible to the class loader currently in effect. If the name is a Java: URL, the JNDI client must be running in a J2EE client or server environment. That is, the client must be running in a container.

Check the exception message to verify the cause.

If the exception is being thrown from the InitialContext constructor, correct the property setting or the `CLASSPATH`.

If the exception is being thrown from a JNDI Context method, make sure the property `java.naming.factory.url.pkgs` includes the package name for the factory required for the URL scheme in the name. URL names with the Java scheme can only be used while running in a container.

ServiceUnavailableException from ″new InitialContext″ operation

This exception indicates that some unexpected problem occurred while attempting to contact the name server to obtain an initial context. The ServiceUnavailableException, like all NamingException objects, can be queried for a root cause. Check the root cause for more information. It is possible that some of the problems described for CommunicationExceptions may also result in a ServiceUnavailableException.

Since this exception is triggered by an unexpected error, there is no probable cause to confirm. If the root cause exception does not indicate what the probable cause is, investigate the possible causes listed for CommunicationExceptions.

CommunicationException thrown from a ″new InitialContext″ operation

The name server identified by the provider URL cannot be contacted to obtain the initial JNDI context. There are many possible causes for this problem, including:

- The host name or port in the provider URL is incorrect.
- The host name cannot be resolved into an IP address by the domain name server, or the IP address does not match the IP address which the server is actually running under.
- A firewall on the client or server is preventing the port specified in the provider URL from being used.

To correct this problem:

- Make sure the provider URL and the network configurations on the client and server machines are correct.
- Make sure the host name can be resolved into an IP address which can be reached by the client machine. You can do this using the ping command.
- If you are running a firewall, make sure that use of the port specified in the provider URL will be allowed.

Errors or access problems after enabling security

What kind of error are you seeing?

- I cannot access part or all of administrative console or use wsadmin after enabling security
- I cannot access a Web page after enabling security
- The client cannot access an enterprise bean after enabling security
- The client never gets prompted when accessing a secured enterprise bean

- I cannot stop an application server, node manager, or node after enabling security
- Error Message: SECJ0314E: Current Java 2 Security policy reported a potential violation
- ″MSGS0508E: The JMS Server security service was unable to authenticate userid:″ error displayed in SystemOut.log when starting an application server
- ″SECJ0237E: One or more vital LTPAServerObject configuration attributes are null or not available″ after enabling security and starting application server.
- AccessControlException is reported in SystemOut.log.
- After enabling single sign-on, I cannot log on to the administrative console.

For general tips on diagnosing and resolving security-related problems, see the topic ″Troubleshooting the security component″ in the WebSphere Application Server InfoCenter.

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

Can’t access part or all of admin console or use wsadmin after enabling security

- If you cannot access the administrative console, or view or update certain objects, look in the log of the the application server which hosts the administrative console page for a related error message.
- You may not have authorized your ID for administrative tasks. This is indicated by errors such as:
  - **[8/2/02 10:36:49:722 CDT] 4365c0d9 RoleBasedAuth A SECJ0305A: Role based authorization check failed for security name MyServer/myUserId, accessId MyServer/S-1-5-21-882015564-4266526380-2569651501-1005 while invoking method getProcessType on resource Server and module Server.**
  - Exception message: ″**ADMN0022E: Access denied for the getProcessType operation on Server MBean**″
  - **When running the command: wsadmin -username j2ee -password j2ee: WASX7246E: Cannot establish ″SOAP″ connection to host ″BIRKT20″ because of an authentication failure. Please ensure that user and password are correct on the command line or in a properties file.**
- To grant an ID administrative authority:
  - From the Administrative Console, select **System Administration -> Console Users** and validate that the ID is a member. If it is not, add the ID with at least monitor access privileges, for read-only access.

Can’t access a web page after enabling security

When secured resources cannot be accessed, causes include:

- Authentication errors - WebSphere Application Server security cannot identify the ID of the person or process. Symptoms of authentication errors include:
  - Netscape browser:
    - ″Authorization failed. Retry?″ message displayed after an attempt to login.
    - Allows any number of attempts to retry login and displays ″Error 401″ message when ″cancel″ is pressed to stop retry.
    - Typical browser message: ″Error 401: Basic realm=’Default Realm’″.
  - Internet Explorer browser:
    - Login prompt displayed again after an attempt to login.

- Allows 3 attempts to retry login.
- Displays ″Error 401″ message after 3 unsuccessful retries.

- Authorization errors - security has identified the requesting person or process as not authorized to access the secured resource. Symptoms of authorization errors include:
  - Netscape browser: ″Error 403: AuthorizationFailed″ message is displayed.
  - Internet Explorer:
    - ″You are not authorized to view this page″ message is displayed.
    - ″HTTP 403 Forbidden″ error is also displayed.
- SSL errors - WebSphere Application Server security uses Secure Socket Layer (SSL) technology internally to secure and encrypt its own communication, and misconfiguration of the internal SSL settings can cause problems. Also you might have enabled SSL encryption for your own Web application or enterprise bean client traffic which, if misconfigured, can cause problems regardless of whether WebSphere Application Server security is enabled. SSL related problems are often indicated by error messages which contain a statement such as:
  - ERROR: Could not get the initial context or unable to look up the starting context. Exiting. followed by:
  - javax.net.ssl.SSLHandshakeException

Authentication error accessing a Web page

Possible causes for authentication errors include:

Username or passwords invalid
Check the username and password and make sure they are correct.

Security configuration error : User registry type is not set correctly.
Check the user registry property in global security settings in the administrative console. Ensure that it is the intended user registry.

Internal program error
If the client application is a Java stand-alone program, it might not be gathering or sending credential information correctly.

If the user registry configuration, user ID, and password appear to be correct, use the WebSphere Application Server to determine the cause of the problem. To enable security trace, use the com.ibm.ws.security.*=all=enabled trace specification.

Authorization error accessing a Web page

If a user who should have access to a resource does not, there is probably a missing configuration step. Review the steps for securing and granting access to resources.

Specifically:
- Check required roles for the accessed Web resource.
- Check the authorization table to make sure that the user, or the groups to which the user belongs, is assigned to one of the required roles.
- You can view required roles for the Web resource in the deployment descriptor of the Web resource.
- You can also view the authorization table for the application that contains the Web resource, using the administrative console.

- Test with a user who is granted the required roles, to see if the user can access the problem resources.
- If the problem user is required to be granted one or more of the required roles, use the administrative console to assign that user to required roles. Then stop and restart the application.

If the user is granted required roles, but still fails to access the secured resources, enable security trace, using com.ibm.ws.security.*=all=enabled as the trace specification. Collect trace information for further resolution.

Cannot access an enterprise bean after enabling security

If client access to an enterprise bean fails after security is enabled:

- Review the steps for securing and granting access to resources.
- Browse the server's logs for errors relating to enterprise bean access and security. Look up any errors in the message table.
- Errors similar to **Authorization failed for /UNAUTHENTICATED while invoking *resource* securityName:/UNAUTHENTICATED;accessId:UNAUTHENTICATED not granted any of the required roles *roles*** indicate that:
  - an unprotected servlet or JSP accessed a protected enterprise bean. When unprotected servlet is accessed, the user is not prompted to login and hence the servlet runs as UNAUTHENTICATED. When it makes a call to an enterprise bean that is protected it will fail.

    To resolve this problem, secure the servlet that is accessing the secured enterprise bean. Make sure the servlet's `runAs` property is set to an ID that can access the enterprise bean.
  - An unauthenticated Java client program is accessing an enterprise bean resource that is protected. This can happen if the file read by the `sas.client.props` properties file used by the client program does not have the `securityEnabled` flag set to `true`.

    To resolve this problem, make sure that the `sas.client.props` file on the client side has its securityEnabled flag set to true.
- Errors similar to **Authorization failed for *valid_user* while invoking *resource* securityName:/username;accessId:xxxxxx not granted any of the required roles *roles*** indicate that a client attempted to access a secured enterprise bean resource, and the supplied user ID is not assigned the required roles for that enterprise bean.
  - Check the required roles for the enterprise bean resource being accessed. Required roles for the enterprise bean resource can be viewed in the deployment descriptor of the Web resource.
  - Check the authorization table and make sure that the user or the group that the user belongs to is assigned one of the required roles. The authorization table for the application that contains the enterprise bean resource can also be viewed using administrative console.

org.omg.CORBA.NO_PERMISSION exceptions returned when programmatically logging on in order to access a secured EJB indicate an authentication exception has occurred on the server. Typically the CORBA exception is triggered by an underlying `com.ibm.WebSphereSecurity.AuthenticationFailedException`. To determine the actual cause of the authentication exception, the full trace stack must be examined.

- Begin by viewing the text in the exception, following ″WSSecurityContext.acceptSecContext(), reason:″. Typically, it describes the failure without further analysis.
- If this does not describe enough of the problem, look up the CORBA minor code. Those codes are listed in the (components reference) in this document.

For example, the following exception:

org.omg.CORBA.NO_PERMISSION: Caught WSSecurityContextException in WSSecurityContext.acceptSecContext(), reason: Major Code[0] Minor Code[0] Message[Exception caught invoking authenticateBasicAuthData from SecurityServer for user jdoe. Reason: com.ibm.WebSphereSecurity.AuthenticationFailedException] minor code: 49424300 completed: No at com.ibm.ISecurityLocalObjectBaseL13Impl.PrincipalAuthFailReason. map_auth_fail_to_minor_code(PrincipalAuthFailReason.java:83) indicates a CORBA minor code of 49424300. The explanation of this error in the CORBA minor code table reads:

```
authentication failed error.
```

In other words, in this case the user ID or password supplied by the client program is probably invalid.

CORBA INITIALIZE exception with **JSAS1477W: SECURITY CLIENT/SERVER CONFIG MISMATCH** error embedded, received by client program from server.

This error indicates that the server's security configuration differs from the client in some fundamental way. The full exception message will list the specific mismatches. For example, the following exception lists three:

```
Exception received: org.omg.CORBA.INITIALIZE:

 JSAS1477W: SECURITY CLIENT/SERVER CONFIG MISMATCH:
The client security configuration (sas.client.props or
 outbound settings in GUI) does not support the server
security configuration for the following reasons:
ERROR 1: JSAS0607E: The client requires SSL Confidentiality
but the server does not support it.
ERROR 2: JSAS0610E: The server requires SSL Integrity
but the client does not support it.
ERROR 3: JSAS0612E: The client requires client
(e.g., userid/password or token), but the server does not support it.
  minor code: 0 completed: No at com.ibm.ISecurityLocalObjectBaseL13Impl.
SecurityConnectionInterceptor.getConnectionKey(SecurityConnectionInterceptor
.java:1770)
```

In general, resolving the problem requires a change to the security configuration of either the client or the server. In order to determine which configuration setting is involved, look at the text following the ″JSAS″ error message. For more detailed explanations and instructions, look up the error message in the message reference, found in the ″Quick Reference″ view of the WebSphere Application Server Version 5 InfoCenter.

In these particular cases:

- In the case of ERROR 1, the client is requiring SSL Confidentiality but the server does not support SSL Confidentiality. This can be resolved in two ways. The server either supports it or the client no longer requires it.

- In the case of ERROR 2, the server requires SSL Integrity but the client does not support SSL Integrity. Again, there are two ways this problem can be solved. Get the server to not require integrity or get the client to support integrity.
- Finally, in the case of ERROR 3, the client requires client authentication via userid and password, but the server does not support this type of client authentication. Again, either the client or the server needs to change the configuration. To change the client configuration, modify the `SAS.CLIENT.PROPS` file for a pure client or change the server's outbound configuration in the Security GUI. To change the target server's configuration, modify the inbound configuration in the Security GUI.

Similarly, an exception like `org.omg.CORBA.INITIALIZE: JSAS0477W: SECURITY CLIENT/SERVER CONFIG MISMATCH:` appearing on the server trying to service a client request indicates a security configuration mismatch between client and server. The steps for resolving the problem is the same as for JSAS1477W exceptions described above.

Client program never gets prompted when accessing secured enterprise bean

Even though it appears security is enabled and an enterprise bean is secured, it may happen that the client executes the remote method without getting prompted.

- If the remote method is protected, you should get an authorization failure. Otherwise,
- Execute the method as an unauthenticated user.

Possible reasons for this include:

- The server you are communicating with may not have security enabled. Check with the WebSphere Application Server administrator to ensure that the server security is enabled. This is done in the global security settings from within the Security section of the administrative console.
- The client does not have security enabled in the `sas.client.props` file. Edit the `sas.client.props` to ensure the property `com.ibm.CORBA.securityEnabled=true`.
- The client does not have a ConfigURL specified. Ensure that the property `com.ibm.CORBA.ConfigURL` is specified on the command line of the Java client, using the -D parameter.
- The specified ConfigURL has an invalid URL syntax or the `sas.client.props` pointed to by it cannot be found. Ensure that the property `com.ibm.CORBA.ConfigURLserver log files` is valid, for example, similar to`file:/C:/WebSphere/AppServer/properties/sas.client.props` on Windows systems. Check the Java documentation for a description of URL formatting rules. Also, validate that the file exists at the specified path.
- The client configuration does not support message layer client authentication (userid and password). Ensure that the `sas.client.props` has one of the following properties set to true:
  - `com.ibm.CSI.performClientAuthenticationSupported=true server log files`
  - `com.ibm.CSI.performClientAuthenticationRequired=true. server log files`
- The server configuration does not support message layer client authentication (userid and password). Check with the WebSphere Application Server administrator to ensure that Userid and Password authentication is specified for the Inbound configuration of the server within the System Administration section of the administrative console administration tool.

Cannot stop an application server, node manager, or node after enabling security

If you are using command line utilities to stop WebSphere Application Server processes, you need to apply additional parameters after enabling security, in order to provide authentication and authorization information.

Use the command: **./stopServer.sh -help** to display the parameters that should be used.

You should use the following command options after enabling security:

- ./stopServer.sh hostname -username *name* -password *password*
- ./stopNode.sh -username *name* -password *password*
- ./stopManager.sh -username *name* -password *password*

Error Message: SECJ0314E: Current Java 2 Security policy reported a potential violation on server

If you find errors on your server similar to:

```
Error Message: SECJ0314E: Current Java 2 Security policy reported a potential
violation of Java 2 Security Permission. Please refer to Problem Determination
Guide for further information.{0}Permission\:{1}Code\:{2}{3}Stack Trace\:
{4}Code Base Location\:{5}
```

then The Java Security Manager `checkPermission()` method has reported a `SecurityException`.

The reported exception may be critical to the secure system. Turn on security trace to determine the potential code that may have violated the security policy. Once the violating code is determined, you should verify if the attempted operation is permitted with respect to Java 2 Security, by examining all applicable Java 2 security policy files and the application code itself.

A more detailed report is enabled by either configuring RAS trace into debug mode, or specifying a Java property.

- Please check the (trace enabling) section for instructions on how to configure RAS trace into debug mode, or
- Specify the following property in the **Application Servers > *server name* > ProcessDefinition > Java Virtual Machine** panel from the administrative console in the **Generic JVM arguments** panel:
  - add the runtime flag `java.security.debugserver log files`
  - Valid values:

    access to print all debug information including: required permission, code, stack, and code base location.

    stack to print debug information including: required permission, code, and stack.

    failure to print debug information including: required permission and code.

For a review of Java security policies and what they mean , see the Java 2 Security documentation at `http://java.sun.com/j2se/1.3/docs/guide/security/index.html` .

Note: If the application is running with Java Mail, this message may be benign. You can update the `installed Enterprise Application root/META-INF/was.policy` file to grant the following permissions to the application:

- permission java.io.FilePermission "${user.home}${/}.mailcap", "read";
- permission java.io.FilePermission "${user.home}${/}.mime.types", "read";
- permission java.io.FilePermission "${java.home}${/}lib${/}mailcap", "read";
- permission java.io.FilePermission "${java.home}${/}lib${/}mime.types", "read";

"MSGS0508E: The JMS Server security service was unable to authenticate userid:" error displayed in SystemOut.log when starting an application server

This error may be a result of installing the JMS messaging api sample and then enabling security. The JMS sample is not designed to work with WebSphere Application Server security. If WebSphere Application Server was installed with samples and no additional code was installed which uses messaging, this message may be ignored.

You can verify the installation of the message-driven bean sample by launching the installation program, selecting **Custom**, and browsing the components which are already installed in the **Select the features you like to install** panel. The JMS sample is shown as **Message-Driven Bean Sample**, under **Embedded Messaging**.

You can also verify this by using the administrative console to open the properties of the application server which contains the samples, selecting "MDBSamples" and clicking "uninstall".

If the problem persists, review the section "Java Messaging (JMS) component troubleshooting tips"

SECJ0237E: One or more vital LTPAServerObject configuration attributes are null or not available

The most likely cause of this error is that LTPA is selected as authentication mechanism but the LTPA keys have not been generated. The LTPA keys are used for encrypting the LTPA token.

To resolve this problem:

1. Select **System Administration -> Console users -> LTPA**
2. Enter a password, which can be anything.
3. Enter the same password in "Confirm Password".
4. Click **Apply**.
5. Click **Generate Keys**.
6. Click on **Save**.

AccessControlException is reported in SystemOut.log

The problem is related to the **Java 2 Security** feature of WebSphere Application Server, the API-level security framework that is implemented in WebSphere Application Server Version 5, if you see an exception similar to the following. The error message and number can vary.

```
E SRVE0020E: [Servlet Error]-[validator]: Failed to load servlet:
java.security.AccessControlException: access denied (java.io.FilePermission
C:\WebSphere\AppServer\installedApps\maeda\adminconsole.ear\adminconsole.war
\WEB-INF\validation.xml read)
```

For an explanation of Java 2 security, how and why to enable or disable it, how it relates to policy files, and how to edit policy files, see the (Java 2 Security) topic in the InfoCenter. The topic explains that Java 2 security is not only used by this product, but can also be implemented by business application developers. Administrators might need to involve developers, if this exception is thrown when a client tries to access a resource hosted by WebSphere Application Server.

Possible causes of these errors include:

- Syntax errors in a policy file.
- Syntax errors in permission specifications in the ra.xml file bundled in a .rar file. This case applies to resource adapters that support ″connector″ access to CICS or other resources.
- An application is missing the specified permission in a policy file, or in permission specifications in an ra.xml file bundled in a .rar file
- The classpath is not set correctly. If the classpath is not set correctly in resource.xml file for SPI, permissions cannot be created correctly.
- A library called by an application, or the application itself, is missing a doPrivileged block to allow access to a resource.
- Permission is specified in the wrong policy file.

To resolve these problems:

- Check all of the related policy files to ensure that the permission shown in the exception, for example java.io.FilePermission, is specified.
- Look for a related ParserException in SystemOut.log which reports the details of the syntax error. For example: **SECJ0189E: Caught ParserException** while creating template for Application Policy C:\WAS\config\cells\xxx\nodes\xxx\app.policy. The exception is com.ibm.ws.security.util.ParserException: line 18: expected ';', found 'grant'
- Look for a message similar to: **SECJ0325W: The permission *permission* specified in the policy file is unresolved.**
- Check the call stack to determine which method does not have the permission. Identify what this method's classpath is. If it is hard to identify the method, enable **Java2 security Report**.
  - Configuring RAS trace by specifying `com.ibm.ws.security.core.*=all=enabled`, or specifying a Java `property.java.security.debug` . Valid values for property java.security.debug are:

    access to print all debug information including: required permission, code, stack, and code base location.

    stack to print debug information including: required permission, code, and stack.

    failure
    to print debug information including: required permission and code.
  - The report shows:

    Permission
    the missing permission.

    Code which method has the problem.

    Stack Trace
    where the access violation occurred.

CodeBaseLocation
the detail of each stack frame.

Usually, Permission and Code should be enough to identify the problem. The following is an example of a report:

```
Permission:
      C:\WebSphere\AppServer\logs\server1\SystemOut_02.08.20_11.19.53.log :
      access denied (java.io.FilePermission
      C:\WebSphere\AppServer\logs\server1\SystemOut_02.08.20_11.19.53.log delete)

Code:
     com.ibm.ejs.ras.RasTestHelper$7  in
     {file:/C:/WebSphere/AppServer/installedApps/maeda/JrasFVTApp.ear/RasLib.jar}

Stack Trace:

java.security.AccessControlException: access denied
(java.io.FilePermission
C:\WebSphere\AppServer\logs\server1\SystemOut_02.08.20_11.19.53.log delete)
        at java.security.AccessControlContext.
               checkPermission(AccessControlContext.java(Compiled Code))
        at java.security.AccessController.
               checkPermission(AccessController.java(Compiled Code))
        at java.lang.SecurityManager.
               checkPermission(SecurityManager.java(Compiled Code))
                                .
Code Base Location:

com.ibm.ws.security.core.SecurityManager :
file:/C:/WebSphere/AppServer/lib/securityimpl.jar
  ClassLoader: com.ibm.ws.bootstrap.ExtClassLoader
  Permissions granted to CodeSource
  (file:/C:/WebSphere/AppServer/lib/securityimpl.jar<no certificates>
  {
    (java.util.PropertyPermission java.vendor read);
    (java.util.PropertyPermission java.specification.version read);
    (java.util.PropertyPermission line.separator read);
    (java.util.PropertyPermission java.class.version read);
    (java.util.PropertyPermission java.specification.name read);
    (java.util.PropertyPermission java.vendor.url read);
    (java.util.PropertyPermission java.vm.version read);
    (java.util.PropertyPermission os.name read);
    (java.util.PropertyPermission os.arch read);
   }
   ( This list continues.)
```

- If the method is SPI, check the **resources.xml** file to ensure that the classpath is correct.
- In order to confirm that all of the policy files are loaded correctly, or what permission each classpath is granted, **enable the trace with `com.ibm.ws.security.policy.*=all=enabled`** . All of the loaded permission will be listed in **trace.log**. Search for `app.policy`,`was.policy` and `ra.xml`. In order to check the permission list for a classpath, search for `Effective Policy for <classpath>server log files`.
- If there is any syntax error in the policy file or ra.xml file, correct it with (policytool). Please avoid editing the policy manually, since it can cause syntax errors.
- If a permission is listed as **Unresolved** it does not take effect. Please make sure that the specified permission name is correct.
- If the classpath specified in `resource.xml` file is not correct, correct it.

- If a required permission does not exist in policy files or the ra.xml file, examine the application code to see if this permission needs to be added. If so, add it to proper policy file or ra.xml file.
- If the permission should not be granted outside of the specific method that is accessing this resource, the code needs to be modified to use a doPrivileged block.
- If this permission does exist in a policy file or ra.xml file and they were loaded correctly, but the classpath still does not have the permission in its list, the location of the permission may not be correct. Please read (Java 2 Security) in the WebSphere Application Server Version 5 InfoCenter carefully to determine in which policy file or ra.xml file that permission should be specified.

Note: If the application is running with Java Mail, this message may be benign. You can update the installed Enterprise Application root/META-INF/was.policy file to grant the following permissions to the application:

- permission java.io.FilePermission "${user.home}${/}.mailcap", "read";
- permission java.io.FilePermission "${user.home}${/}.mime.types", "read";
- permission java.io.FilePermission "${java.home}${/}lib${/}mailcap", "read";
- permission java.io.FilePermission "${java.home}${/}lib${/}mime.types", "read";

After enabling single sign-on, I cannot log on to the administrative console

This problem occurs when single sign-on (SSO) is enabled, and you attempt to access the administrative console using the short name of the server, for example http://myserver:9090/admin. The server will accept your userID and password, but returns you to the sign-on page instead of the administrative console.

To correct this problem, use the fully-qualified hostname of the server, for example http://myserver.mynetwork.mycompany.com:9090/admin.

Errors after enabling Secure Sockets Layer, or Secure Sockets Layer-related error messages

If you are unable to access resources using a Secure Sockets Layer (SSL) type URL (beginning with "https:"), or encounter error messages which indicate SSL problems, ensure that your HTTP server has been configured correctly for SSL by browsing the welcome page of the HTTP server using SSL by entering the URL **https://** *hostname*.

If the page works with HTTP, but not HTTPS, the problem is in the HTTP server.

- Refer to the documentation for your HTTP server for instructions on correctly enabling SSL. If you are using the IBM HTTP Server or Apache, go to:http://www.ibm.com/software/webservers/httpservers/library.html. Select the link *Frequently Asked Questions*, and the topic *SSL*.
- If you are using the **IKeyman** (IBM Key Management) tool to create certificates and keys, remember to "stash" the password to a file when creating the KDB file with the IBM Key Management Tool.
  1. Go to the directory where the KDB file was created, and check to see if there is a .sth file. If not,
  2. Open the KDB file with the IBM Key Management Tool, select **Key Database File > Stash Password**.

3. It will display ″The password has been encrypted and saved in the file″.

If the HTTP server handles SSL-encrypted requests successfully, or is not involved (for example, traffic flows from a Java client application directly to an enterprise bean hosted by the WebSphere Application Server, or the problem appears only after enabling WebSphere Application Server security), what kind of error are you seeing?

- javax.net.ssl.SSLHandshakeException - The client and server could not negotiate the desired level of security. Reason: handshake failure
- javax.net.ssl.SSLHandshakeException - The client and server could not negotiate the desired level of security. Reason: unknown certificate
- javax.net.ssl.SSLHandshakeException - The client and server could not negotiate the desired level of security. Reason: bad certificate
- org.omg.CORBA.INTERNAL: EntryNotFoundException or NTRegistryImp E SECJ0070E: No privilege id configured for: error when programmatically creating a credential.

For general tips on diagnosing and resolving security-related problems, see the topic ″Troubleshooting the security component″ in the WebSphere Application Server Version 5 InfoCenter

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

javax.net.ssl.SSLHandshakeException - The client and server could not negotiate the desired level of security. Reason: handshake failure

If you see a Java exception stack similar to: **[Root exception is org.omg.CORBA.TRANSIENT: CAUGHT_EXCEPTION_WHILE_CONFIGURING_SSL_CLIENT_SOCKET: JSSL0080E: javax.net.ssl.SSLHandshakeException - The client and server could not negotiate the desired level of security. Reason: handshake failure:host=MYSERVER,port=1079 minor code: 4942F303 completed: No] at com.ibm.CORBA.transport.TransportConnectionBase.connect (TransportConnectionBase.java:NNN)**, some possible causes are:

- Not having common ciphers between the client and server.
- Not specifying the correct protocol.

To correct these problems:

- Review the SSL settings by browsing the **WebSphere Administrative Console Security Settings -> SSL Configuration Repertoires -> DefaultSSLSettings** (or other named SSL settings), then selecting the **Secure Sockets Layer (SSL)** option from the **Additional Properties** menu. You can also browse the file manually by viewing: *install_dir*/properties/sas.client.props.
- Check the property specified by **com.ibm.ssl.protocol** to determine which protocol is specified.
- Check the cipher types specified by **com.ibm.ssl.enabledCipherSuites**. You may want to add more cipher types to the list. To see which cipher suites are currently enabled, go to the properties page of the SSL settings as described above, and look for the **Cipher Suites** property. To see the list of all possible cipher suites, go to the properties page of the SSL settings as described above, then view the online help for that page. From the help page, click **Configure additional SSL settings**.

- Correct the protocol or cipher problem by using a different client or server protocol and/or cipher selection. Typical protocols are SSL or SSLv3.
- Make the cipher selection 40-bit instead of 128-bit. For CSIv2, set both of these properties to false:
  - **com.ibm.CSI.performMessageConfidentialityRequired=false**
  - **com.ibm.CSI.performMessageConfidentialitySupported=false**

in the sas.client.props file, or set the **security level=medium** in the Administrative Console settings.

javax.net.ssl.SSLHandshakeException: unknown certificate

If you see a Java exception stack similar to: **ERROR: Could not get the initial context or unable to look up the starting context. Exiting. Exception received: javax.naming.ServiceUnavailableException: A communication failure occurred while attempting to obtain an initial context using the provider url: "corbaloc:iiop:localhost:2809". Make sure that the host and port information is correct and that the server identified by the provider url is a running name server. If no port number is specified, the default port number 2809 is used. Other possible causes include the network environment or workstation network configuration. [Root exception is org.omg.CORBA.TRANSIENT: CAUGHT_EXCEPTION_WHILE_CONFIGURING_SSL_CLIENT_SOCKET: JSSL0080E: javax.net.ssl.SSLHandshakeException - The client and server could not negotiate the desired level of security. Reason: unknown certificate:host=MYSERVER,port=1940 minor code: 4942F303 completed: No]**, it may be caused by not having the server's personal certificate in the client truststore.

To correct this problem:

- Check the client truststore to determine if the signer certificate from the server personal certificate is there. For a self-signed server personal certificate, the signer certificate is the public key of the personal certificate. For a CA signed server personal certificate, the signer certificate is the root CA certificate of the CA which signed the personal certificate.
- Add the server signer certificate to the client truststore.

javax.net.ssl.SSLHandshakeException: bad certificate

If you see a Java exception stack similar to **ERROR: Could not get the initial context or unable to look up the starting context. Exiting. Exception received: javax.naming.ServiceUnavailableException: A communication failure occurred while attempting to obtain an initial context using the provider url: "corbaloc:iiop:localhost:2809". Make sure that the host and port information is correct and that the server identified by the provider url is a running name server. If no port number is specified, the default port number 2809 is used.Other possible causes include the network environment or workstation network configuration. [Root exception is org.omg.CORBA.TRANSIENT: CAUGHT_EXCEPTION_WHILE_CONFIGURING_SSL_CLIENT_SOCKET: JSSL0080E: javax.net.ssl.SSLHandshakeException - The client and server could not negotiate the desired level of security. Reason: bad certificate: host=MYSERVER,port=1940 minor code: 4942F303 completed: No]**, it can be caused by having a personal certificate in the client keystore used for SSL mutual authentication but not having extracted the signer certificate into the server truststore so that the server could trust it whenever the SSL handshake is made.

To verify this, check the server truststore to determine if the signer certificate from the client personal certificate is there. For a self-signed client personal certificate, the signer certificate is the public key of the personal certificate. For a CA signed client personal certificate, the signer certificate is the root CA certificate of the CA which signed the personal certificate.

To correct this problem, add the client signer certificate to the server truststore.

org.omg.CORBA.INTERNAL: EntryNotFoundException or NTRegistryImp E SECJ0070E: No privilege id configured for: error when programmatically creating a credential

If you encounter the following exception in a client application attempting to request a credential from a WebSphere Application Server using SSL mutual authentication:

ERROR: Could not get the initial context or unable to look up the starting context. Exiting. Exception received: org.omg.CORBA.INTERNAL: Trace from server: 1198777258 at host MYHOST on port 0 >>org.omg.CORBA.INTERNAL: EntryNotFoundException minor code: 494210B0 completed: No at com.ibm.ISecurityLocalObjectBaseL13Impl. PrincipalAuthFailReason. map_auth_fail_to_minor_code(PrincipalAuthFailReason.java:99)

or a simutaneous error from the WebSphere Application Server that resembles:

[7/31/02 15:38:48:452 CDT] 27318f5 NTRegistryImp E SECJ0070E: No privilege id configured for: testuser

the cause may be that the user id sent by the client to the server is not in the server's user registry.

To confirm that this is the problem, check that an entry exists for the personal certificate which is being sent to the server. Depending on the mechanism used for user registry, look at the native operating system user ID's or LDAP server entries.

To correct this problem, add the user ID to the user registry entry (for example, operating system, LDAP directory, or other custom registry) for the personal certificate identity.

Errors in messaging (JMS API)

What kind of problem are you seeing?

- javax.jms.JMSException: MQJMS2008: failed to open MQ queue in JVM log.
- SVC: jms.BrokerCommandFailedExceptfailed: 3008.

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, see "Messaging (JMS) component troubleshooting tips" on page 91. If you are still unable to resolve the problem, contact IBM support for further assistance.

javax.jms.JMSException: MQJMS2008: failed to open MQ queue in JVM log

This error can occur when the MQ queue name is not defined in the Internal JMS Server Queue Names List. This can occur if a WebSphere Application Server Queue Destination is created, without adding the Queue Name to the internal JMS Server Queue Names List.

To resolve this problem:

- Open the WebSphere Application Server Administrative Console.
- Click **Servers > Manage Application Servers >** *server_name*> **Server Components > JMS Servers**.
- Add the Queue Name to the list.
- Save the changes and restart the server.

SVC: jms.BrokerCommandFailedExceptfailed: 3008

One possible cause of this error is that you have logged on to a Windows 2000 system as an administrator.

To correct this problem, log out and log in again as a user, rather than an administrator.

Errors returned to client trying to send a SOAP request

What kind of problem are you seeing?

- SOAPException: faultCode=SOAP-ENV:Client; msg=Error opening socket; java.net.ConnectException: Connection refused: connect
- javax.security.cert.CertPathBuilderException: No end-entity certificate matching the selection criteria could be found.

If none of these errors match the one you see:

- Browse the target application server log files. (by default file `installation_directory/server_name/SystemErr.log` and `SystemOut.log` (main installation log file) for clues. See ″Viewing the JVM Logs″ for more information.
- Look up any error or warning messages in the message table.
- See the article ″Universal Description, Discover, and Integration, Web Services and SOAP components″ for more information.

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

SOAPException: faultCode=SOAP-ENV:Client; msg=Error opening socket; java.net.ConnectException: Connection refused: connect

The most likely cause of this refused connection is that it was sent to the default port, 80, and an HTTP server is not installed or configured.

To verify this situation, send the message directly to the SOAP port. For example, to `http://hostname:9080server log files`. If this works, there are two ways to resolve the problem:

- Continue specifying port 9080 on SOAP requests.
- If an HTTP server such as the IBM HTTP Server, iis, IPlanet, or others, is not installed, install one and then step through the WebSphere Application Server installation to install the associated plug-in component.
- If an HTTP server is installed:
  - Regenerate the HTTP plug-in configuration in the administrative console by clicking **Environment > Update WebServer Plugin**, and restart the HTTP server.
  - If the problem persists, view the HTTP server access and error logs, as well as the `install_dir/logs/http_plugin.log file` for more information.

javax.security.cert.CertPathBuilderException: No end-entity certificate matching the selection criteria could be found

This error usually indicates that new or updated security keys are needed. The security key files are:

- SOAPclient
- SOAPserver
- sslserver.p12

In an installed application, these files are located in: `install_dir/installedApps/application_name.ear/soapsec.war/key/`. After replacing these files, you must stop and restart the application.

To replace these files in a SOAP-enabled application that has not yet been installed:

- Expand `application_name.ear`.
- Expand `soapsec.war`.
- Replace the security key files in the `key/` directory.
- After you have replaced these files, install the application and restart the server.

Client program does not work

What kind of problem are you seeing?

- ActiveX client fails to display ASP files, or WebSphere Application Server resources (JSP files, servlet, or HTML pages), or both.

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

ActiveX client fails to display ASP files, or WebSphere Application Server resources (JSP files, servlet, or HTML pages),or both

A possible cause of this problem is that both IIS (for serving ASP files) and an HTTP server that supports WebSphere Application Server (such as IBM HTTP Server) are deployed on the same host. This leads to misdirected HTTP traffic if both servers are listening on the same port (such as the default port 80).

To resolve this problem, do one of the following:

- Open the IIS administrative panel, and edit the properties of the default Web server to change the port number to something other than 80;
- Install IIS and the WebSphere Application Server HTTP server on separate servers.

Troubleshooting application run-time and management problems

Select the problem you are having with running or managing deployed code for WebSphere Application Server:

- I have problems bringing up or using the administrative console.
- I have problems starting or using the wsadmin command prompt.
- My Web module or application server dies or hangs.
- I get errors trying to configure and enable security.
- I cannot seem to distribute the workload across clustered servers.

- There are problems setting up the multiserver Deployment Manager environment.
- I cannot uninstall or remove a node or application server.
- I have problems creating or using HTTP sessions.
- I have problems using tracing, logging, logfiles, or other troublshooting features.
- I get errors connecting to the administrative console from a Netscape browser.

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

Installation completes but the administrative console does not start

What kind of problem are you having?

- ″Internal Server Error″, ″Page cannot be found″, 404, or similar error trying to view administrative console.
- ″Unable to process login. Please check User ID and password and try again. ″ error when trying to access console page.
- Directory paths in the console are garbled.

If you are able to bring up the browser page, but the console's behavior is inconsistent, error-prone, or unresponsive, try upgrading the browser you are using. Older browsers may not support the administrative console's features.

If none of these steps solves the problem, check to see if the problem has been identified and documented using the links in . If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

″Internal Server Error″, ″Page cannot be found″, 404, or similar error trying to view administrative console

If you are unable to view the administrative console, here are some steps to try:

- Verify that the application server which supports the administrative console is up and running.
  - For a ″base″ configuration, the administrative console is deployed by default on ″server1″. Before viewing the administrative console, you must
    - Run the **startServer server1** command for Windows or **./startServer.sh server1** command for Unix from a command prompt in the *install_dir*\bin directory, or
    - Click the ″start application server″ link from the ″first steps″ panel, or
    - Start WebSphere Application Server as a service or from the Start menu, if you are using Windows.
  - If you are using the Deployment Manager (for a multi-node configuration), run the startManager command from the *Network_Deployment_install_dir*\bin directory.
  - View the SystemOut.log file for the application server or deployment manager to verify that the server supporting the administrative console has actually started.
- Check the URL you are using to view the console. By default, it is http://*server_name*:9090/admin.

- If you are browsing the console from a remote machine, try to eliminate connection, address and firewall issues by:
  - Pinging the server machine from a command prompt, using the same server name as in the URL.
- If you have never been able to access the administrative console, verify that the installation was successful.

"Unable to process login. Please check User ID and password and try again. " error when trying to access console page

This error indicates that security has been enabled for WebSphere Application Server, and the user ID or password supplied is either invalid or not authorized to access the console.

To access the console,

- If you are the administrator, use the ID defined as the security administrative ID. This ID is stored in the WebSphere Application Server directory structure in the file security.xml.
- If you are not the administrator, ask the administrator to enable your ID for the administrative console.

Directory paths in the console are garbled

If directory paths used for classpaths or resources specified in the Application Assembly Tool, configuration files, or elsewhere, appear garbled in the administrative console, it may be because the Java runtime interprets a backslash (\) as denoting a control character.

To resolve, modify Windows-style classpaths by replacing occurrences of single backslashes to two. For example, change "c:\MyFiles\MyJsp.jsp" to "c:\\MyFiles\\MyJsp.jsp".

Problems starting or using the wsadmin command

What kind of problem are you having?

- "WASX7023E: Error creating "SOAP" connection to host" or similar error trying to launch wsadmin command line utility.
- "com.ibm.bsf.BSFException: error while eval'ing Jacl expression: no such method "<command name>" in class com.ibm.ws.scripting.AdminConfigClient" returned from wsadmin command.
- WASX7022E returned from running "wsadmin -c ..." command, indicating invalid command.
- com.ibm.ws.scripting.ScriptingException: WASX7025E: String "" is malformed; cannot create ObjectName.
- "The input line is too long" error returned from the wsadmin command on a Windows platform.

If you do not see your problem here:

- If you are not able to enter wsadmin command mode, try running **wsadmin -c "$Help wsadmin"** for help in verifying that you are entering the command correctly.
- If you can get the wsadmin command prompt, enter **$Help help** to verify that you are using specific commands correctly.

- wsadmin commands are a superset of Jacl (Java Command Language), which is in turn a Java-based implementation of the Tcl command language. For details on Jacl syntax beyond wsadmin commands, refere to the Tcl developers' site, http://www.tcl.tk. For specific details relating to the Java implementation of Tcl, refer to http://www.tcl.tk/software/java.
- Browse the *install_dir*/logs/wsadmin.traceout file for clues.
  - Keep in mind that wsadmin.traceout is refreshed (existing log records are deleted) whenever a new wsadmin session is started.
  - If the error returned by wsadmin does not seem to apply to the command you entered, for example, you receive WASX7023E, stating that a connection could not be created to host ″myhost,″ but you did not specify ″-host myhost″ on the command line, examine the properties files used by wsadmin to determine what properties are specified. If you do not know what properties files were loaded, look for the WASX7326I messages in the wsadmin.traceout file; there will be one of these messages for each properties file loaded.

If none of these steps fixes your problem, check to see if the problem has been identified and documented by looking at the (support (hints and tips, technotes, and fixes)). If you don't find your problem listed there contact IBM support.

″WASX7023E: Error creating ″SOAP″ connection to host″ or similar error trying to launch wsadmin command line utility

By default, the wsadmin utility attempts to connect to an application server at startup. This is because some commands act upon running application servers. This error indicates that no connection could be established.

To resolve this problem:

- If you are not sure whether an application server is running, start it by entering **startserver** ***servername*** from the command prompt. If the server is already running, you will see an error similar to ″ADMU3027E: An instance of the server is already running″.
- If you are running a Network Deployment configuration, you will first need to start the deployment manager by running ″startManager″ or ″startManager.sh″ from the *install_dir*/bin directory. Then you can launch wsadmin immediately to connect to the deployment manager, or start a node and application server to connect to.
- If an application server is running and you still get this error:
  - If you are running remotely (that is, on a different machine from the one running WebSphere Application Server), you must use the **-host** ***hostname*** option to the wsadmin command to direct wsadmin to the right physical server.
  - If you are using the -host option, try pinging the server machine from the command line from the machine on which you are trying to launch wsadmin to verify there are no issues of connectivity such as firewalls.
  - verify that you are using the right port number to connect to the WebSphere Application Server process:
    - If you are not specifying a port number (using the -port option) when you start wsadmin, wsadmin uses the default port specified in *install_dir*/properties/wsadmin.properties, property name=com.ibm.ws.scripting.port (default value =8879).
    - The port that wsadmin should send on depends on the server process wsadmin is trying to connect to.

For a single-server installation, wsadmin attempts to connect to the application server process by default. To verify the port number:

- Look in the file *install_dir*/config/cells/*node_name*/nodes/*node_name*/serverindex.html for a tag containing the property **serverType="APPLICATION_SERVER"**.
- Look for an entry within that tag with the property **endPointName="SOAP_CONNECTOR_ADDRESS"**.
- Look for a **port** property within that tag. This is the port wsadmin should send on.

In a Network Deployment installation, wsadmin launched from the bin directory on the Network Deployment installation attempts to send requests to the deployment manager by default. To verify the port number:

- Get the hostname of the node on which the Deployment Manager is installed.
- Using that hostname, look in

  install_dir/config/cells/*node_name*Network/nodes/*node_name*Manager/serverindex.html

  for a tag containing the property **serverType="DEPLOYMENT_MANAGER"**.
- Within that tag, look for an entry with a property **endPointName="SOAP_CONNECTOR_ADDRESS"**.
- Within that tag, look for a "port" property. This is the port wsadmin should send on.

"com.ibm.bsf.BSFException: error while eval'ing Jacl expression: no such method "<command name>" in class com.ibm.ws.scripting.AdminConfigClient" returned from wsadmin command.

This error is usually caused by a misspelled command name. Use the **$AdminConfig help** command to get information about what commands are available. Note that command names are case-sensitive.

WASX7022E returned from running "wsadmin -c ..." command, indicating invalid command

If the command following -c appears to be valid, the problem may be caused by the fact that on Unix, using wsadmin -c to invoke a command that includes dollar signs results in the shell attempting to do variable substitution. To confirm that this is the problem, check the command to see if it contains an unescaped dollar sign, for example: **wsadmin -c "$AdminApp install ...."**.

To correct this problem, escape the dollar sign with a backslash. For example: **wsadmin -c "\$AdminApp install ..."**.

com.ibm.ws.scripting.ScriptingException: WASX7025E: String "" is malformed; cannot create ObjectName

One possible cause of this error is that an empty string was specified for an object name. This can happen if you use one scripting statement to create an object name and the next statement to use that name, perhaps in an "invoke" or "getAttribute" command, but you don't check to see if the first statement really returned an object

name. For example the following samples use basic Jacl commands in addition to the wsadmin Jacl extensions to make a sample script:

```
#let's misspell "Server"
set serverName [$AdminControl queryNames type=Server,*]
$AdminControl getAttributes $serverName
```

To correct this error, make sure that object name strings have values before using them. For example:

```
set serverName[$AdminControl queryNames type=Server,*]
if {$serverName == ""} {puts "queryNames returned empty - check query argument"}
else {$AdminControl getAttributes $serverName}
```

For details on Jacl syntax beyond wsadmin commands, refer to the Tcl developers' site, http://www.tcl.tk.

"The input line is too long" error returned from the wsadmin command on a Windows platform

This error indicates that the Windows command line limit of 1024 characters has been exceeded, probably due to a long path name used within the `wsadmin.bat` command. The problem can be avoided by using the Windows `subst` command, which allows you to map an entire path to a virtual drive. To see the syntax of the `subst` command, enter **`help subst`** from a Windows command prompt.

For example if the product resides in `c:\TestEnvironment\Beta\WebSphere\AppServer`, edit the file `c:\TestEnvironment\Beta\WebSphere\AppServer\bin\setupCmdLine.bat` as follows:

```
subst w: c:\TestEnvironment\Beta\WebSphere\AppServer

REM comment out the old line
REM SET WAS_HOME=C:\TestEnvironment\Beta\WebSphere\AppServer

SET WAS_HOME=w:

REM comment out the old line
REM SET JAVA_HOME=C:\TestEnvironment\Beta\WebSphere\AppServer\java

SET JAVA_HOME=w:\java
```

Web module or application server dies or hangs

If an application server dies (its process spontaneously closes), or freezes (its Web modules stop responding to new requests):

- Isolate the problem by installing Web modules on different servers, if possible.
- Read the (resource analyzer)) topic. You can use the performance viewer to determine which resources have reached their maximum capacity, such as Java heap memory (indicating a possible memory leak) and database connections. If a particular resource appears to have reached its maximum capacity, review the application code for a possible cause:
  - If database connections are used and never freed, ensure that application code performs a **close()** on any opened **Connection** object within a **finally{}** block.
  - If there is a steady increase in servlet engine threads in use, review application **synchronized** code blocks for possible deadlock conditions.
  - If there is a steady increase in a JVM heap size, review application code for memory leak opportunities, such as static (class-level) collections, that can cause objects to never get garbage-collected.

- As an alternative to using the performance viewer to detect memory leak problems, enable verbose garbage collection on the application server. This feature adds detailed statements to the JVM error log file of the application server about the amount of available and in-use memory. To set up verbose garbage collection:
  1. Select **Servers > Application Servers >** *server_name* **> Process Definition > Java Virtual Machine**, and enable **Verbose Garbage Collection**.
  2. Stop and restart the application server.
  3. Periodically, or after the application server stops, browse the log file for garbage collection statements. Look for statements beginning with ″allocation failure″. The string indicates that a need for memory allocation has triggered a JVM garbage collection (freeing of unused memory). Allocation failures themselves are normal and not necessarily indicative of a problem. The allocation failure statement is followed by statements showing how many bytes are needed and how many are allocated.

     If there is a steady increase in the total amount of free and used memory (the JVM keeps allocating more memory for itself), or if the JVM becomes unable to allocate as much memory as it needs (indicated by the bytes needed statement), there might be a memory leak.
- If neither the performance viewer or verbose garbage collection output indicates that the application server is running out of memory, one of the following problems might be present:
  – There is a memory leak in application code that you must address. To pinpoint the cause of a memory leak, enable the **RunHProf** function in the (pane) of the problem application server:
    - In the same JVM pane, set the **HProf Arguments** field to a value similar to `depth=20,file=heapdmp.txt`. This value shows exception stacks to a maximum of 20 levels, and saves the heapdump output to the `install_root/bin/heapdmp.txt` file.
    - Save the settings.
    - Stop and restart the application server.
    - Re-enact the scenario or access the resource that causes the hang or crash, if possible. Stop the application server. If this is not possible, wait until the hang or crash happens again and stop the application server.
    - Examine the file into which the heapdump was saved. For example, examine the `install_root/bin/heapdmp.txt` file:
      - Search for the string, ″SITES BEGIN″. This finds the location of a list of Java objects in memory, which shows the amount of memory allocated to the objects.
      - The list of Java objects occurs each time there was a memory allocation in the JVM. There is a record of what type of object the memory instantiated and an identifier of a trace stack, listed elsewhere in the dump, that shows the Java method that made the allocation.
      - The list of Java object is in descending order by number of bytes allocated. Depending on the nature of the leak, the problem class should show up near the top of the list, but this is not always the case. Look throughout the list for large amounts of memory or frequent instances of the same class being instantiated. In the latter case, use the ID in the `trace stack` column to identify allocations occurring repeatedly in the same class and method.
      - Examine the source code indicated in the related trace stacks for the possibility of memory leaks.

  - The default (heap size) of the application server needs to be increased.
  - There is a defect in the WebSphere Application Server product that you must either report, or correct by (installing a fix or FixPak), from a maintenance download. Contact IBM support.
- If an application server spontaneously dies, look for a Java thread dump file. The JVM creates the file in the product directory structure, with a name like `javacore[number].txt`.
- Force an application to create a thread dump (or javacore). Here is the process for forcing a thread dump, which is different from the process in earlier releases of the product:
  1. Using the wsadmin command prompt, get a handle to the problem application server: **wsadmin>set jvm [$AdminControl completeObjectName type=JVM,process=server1,*]**
  2. Generate the thread dump: **wsadmin>$AdminControl invoke $jvm dumpThreads**.
  3. Look for an output file in the installation root directory with a name like `javacore.date.time.id.txt`.
- Browse the thread dump for clues:
  - If the JVM creates the thread dump as it closes (the thread dump is not manually forced), there might be ″error″ or ″exception information″ strings at the beginning of the file. These strings indicate the thread that caused the application server to die.
  - The thread dump contains a snapshot of each thread in the process, starting in the section labeled ″Full thread dump.″
    - Look for threads with a description that contains ″state:R″. Such threads are active and running when the dump is forced, or the process exited.
    - Look for multiple threads in the same Java application code source location. Multiple threads from the same location might indicate a deadlock condition (multiple threads waiting on a monitor) or an infinite loop, and help identify the application code with the problem.

If these steps do not fix your problem, search to see if the problem is known and documented, using the methods identified in the ((hints and tips, technotes, and fixes)) topic. If you find that your problem is not known, contact IBM support to report it.

Errors when trying to configure or enable security

What kind of error are you seeing?

- ″LTPA password not set. validation failed″ message displayed as error in the Administrative Console after enabling global security.
- ″Validation failed for user [userid]. Please try again...″ displayed in the Administrative Console when enabling global security.
- If you have successfully configured security (made changes, saved the configuration, and enabled security with no errors), but are now having problems accessing Web resources or the administrative console, refer to Errors or access problems after enabling security.

For general tips on diagnosing and resolving security-related problems, see the topic ″Troubleshooting the security component″ in the WebSphere Application Server Version 5 InfoCenter

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

"LTPA password not set. validation failed" message displayed as error in the Administrative Console after saving global security settings

This error can be caused if, when configuring WebSphere Application Server security, "LTPA" is selected as the authentication mechanism, and the LTPA password field is not set. To resolve this problem:

- Select Security **Authentication Mechanism > LTPA** in the console left-hand navigation pane.
- Complete the password and confirm password fields.
- Click **OK**.
- Try setting Global Security again.

"Validation failed for user userid. Please try again..." displayed in the Administrative Console after saving global security settings

This typically indicates that a setting in the User Registry configuration is not valid:

- If the user registry is LocalOS, it's likely that either the server userid and password is invalid or the server userid does not have "Act As Part of the Operating System" (for NT) or root authority (for Unix). It needs this authority in order to access the LocalOS user registry to authenticate.
- If the user registry is Lightweight Directory Access Protocol (LDAP):
  - Any of the settings that enable WebSphere Application Server to communicate with LDAP might be invalid, such as the LDAP server's userid, password, host, port, or LDAP filter. When you select **Apply** or **OK** on the Global Security panel, a validation routine connects to the registry just as it would during runtime when security is enabled. This is done in order to verify any configuration problems immediately, instead of waiting until the server restarts.
  - If the BIND DN is required, you must specify a DN instead of a short name.
  - Sometimes the LDAP server might be down during configuration. The best way to check this is to issue a command line search - LdapSearch, to search for the server ID. This way you can determine if the server is running and if the server ID is a valid entry in the LDAP.
- If the user registry is Custom, double check that your implementation is in the classpath. Also, check to see if your implementation is authenticating properly.
- Regardless of registry type, check the User Registries configuration panels to see if you can find a configuration error:
  - Go back to the User Registries configuration panels and retype the password for the server ID.
- See if there is an obvious configuration error. Double check the attributes specified.

After you enable security on HP-UX 11i platforms, the following error in the `native_stdout.log` file occurs, along with a core dump and WebSphere Application Server does not start:

```
Java HotSpot(TM) Server VM warning:
Unexpected Signal 11 occurred under user-defined signal handler 0x7895710a
```

To work around this error, apply the fixes recommended by HP for Java at the following URL:
http://www.hp.com/products1/unix/java/infolibrary/patches.html

Cannot uninstall an application or remove a node or application server

What kind of problem are you having?

- After uninstalling an application through wsadmin tool, the application continues to run and throws ″DocumentIOException″
- The removeNode command does not remove the installed application from the deployment manager
- I cannot display the syntax for the removeNode command.

If none of these steps fixes your problem:

- Make sure that the application and its Web and EJB modules, are in a stopped state before uninstalling.
- If you are uninstalling or installing an application using **wsadmin**, make sure that you are using the **-conntype NONE** option to invoke **wsadmin** and enable local mode. To use the **-conntype NONE** option, stop the hosting application server before uninstalling the application.
- Check to see if the problem has been identified and documented by looking at the ((hints and tips, technotes, and fixes)).
- If you don't find your problem listed there contact IBM support

After uninstalling application through the wsadmin tool, the application throws ″DocumentIOException″

If this exception occurs after the application was uninstalled using wsadmin with the `-conntype NONE` option:

- Restart the server or,
- Rerun the uninstall command without the `-conntype NONE` option.

The removeNode command does not remove the installed application from the deployment manager

If the applications were installed indirectly using the **addNode** program with the **-includeapps** option, then `removeNode` will not uninstall them, since they may be in use by other nodes. These applications must be explicitly uninstalled, for example through the administrative console.

I cannot display the syntax for the removeNode command

Unlike the `addNode` command, the `removeNode` command is valid with no parameters, so executing it will execute the operation, that is, remove the node, without displaying the command syntax.

To see the valid options for removeNode, execute `removeNode -?` or `removeNode -help`.

Problems creating or using HTTP sessions

Note: To view and update the Session Manager settings discussed here, use the administrative console. Select the application server that hosts the problem application, then under **Additional properties**, select **Web Container**, then **Session manager**.

What kind of problem are you having?

- HTTP Sessions are not getting created, or are lost between requests.
- HTTP Sessions are not persistent (session data lost when application server restarts, or not shared across cluster).
- Session is shared across multiple browsers on same client machine.
- Session is not getting invalidated immediately after specified Session timeout interval.
- Unwanted sessions are being created by jsps.

If your problem is not described here, or none of these steps fixes the problem:

- Review ″Troubleshooting the HTTP Session Manager″ in the WebSphere Application Server Version 5 InfoCenter for general steps on debugging Session-manager related problems.
- Review ″Managing HTTP Sessions″ for information on how to configure the Session manager, and best practices for using it.
- Check to see if the problem has been identified and documented by looking at the ((hints and tips, technotes, and fixes)).
- If you don't find your problem listed there contact IBM support.

HTTP Sessions are not getting created, or are lost between requests

By default, the Session Manager uses cookies to store the session ID on the client between requests. Unless you intend to avoid cookie-based session tracking, ensure that cookies are flowing between WebSphere Application Server and the browser:

- Make sure the **Enable cookies** checkbox is checked under the **Session tracking Mechanism** property.
- Make sure cookies are enabled on the browser you are testing from or from which your users are accessing the application.
- Check the Cookie domain specified on the SessionManager (to view or update the cookie settings, in the **Session tracking mechanism->enable cookies** property, click **Modify**).
  - For example, if the cookie domain is set as ″.myCom.com″, resources should be accessed using that domain name, e.g. http://www.myCom.com/myapp/servlet/sessionservlet.
  - If the domain property is set, make sure it begins with a dot (.). Certain versions of Netscape do not accept cookies if domain name doesn't start with a dot. Internet Explorer honors the domain with or without a dot. For example, if the domain name is set to *mycom.com*, change it to *.mycom.com* so that both Netscape and Internet Explorer honor the cookie.
- Check the **Cookie path** specified on the SessionManager. Check whether the problem url is hierarchially below the Cookie path specified. If not correct the Cookie path.
- If the `Cookie maximum age` property is set, ensure that the client (browser) machine's date and time is the same as the server's, including the time zone. If

the client and the server time difference is over the ″Cookie maximum age″ then every access would be a new session, since the cookie will ″expire″ after the access.

- If you have multiple web modules within an enterprise application that track sessions:
  - If you want to have different session settings among web modules in an enterprise application, ensure that each web module specifies a different cookie name or path, or
  - If Web modules within an enterprise application use a common cookie name and path, ensure that the HTTP session settings, such as Cookie maximum age, are the same for all Web modules. Otherwise cookie behavior will be unpredictable, and will depend upon which application creates the session. Note that this does not affect session data, which is maintained separately by Web module.
- Check the cookie flow between browser and server:
  1. On the browser, enable ″cookie prompt″. Hit the servlet and make sure cookie is being prompted.
  2. On the server, enable SessionManager trace. (Enable tracing) for the HTTP Session Manager component, by using the trace specification ″com.ibm.ws.webcontainer.httpsession.*=all=enabled″. After trace is enabled, excericise your session-using servlet or jsp, then follow the instructions for dumping and browsing the trace output .
  3. Access the session servlet from the browser.
  4. The browser will prompt for the cookie; note the jsessionid.
  5. Reload the servlet, note down the cookie if a new cookie is sent.
  6. Check the session trace and look for the session id and trace the request by the thread. Verify that the session is stable across web requests:
     - Look for **getIHttpsession(...)** which is start of session request.
     - Look for **releaseSesson(..)** which is end of servlet request.
- If you are using URL rewriting instead of cookies:
  - Ensure there are no static HTML pages on your application's navigation path.
  - Ensure that your servlets and jsp files are implementing URL rewriting correctly. For details and an example see ″Session tracking options″.
- If you are using SSL as your session tracking mechanism:
  - Ensure that you have SSL enabled on your IBM HTTP Server or iPlanet HTTP server.
  - Review Session tracking with SSL information.
- If you are in a clustered (multiple node) environment, ensure that you have session persistence enabled.

HTTP Sessions are not persistent

If your HTTP sessions are not persistent, that is session data is lost when the application server restarts or is not shared across the cluster:

- Check the Datasource.
- Check the SessionManager's Persistence Settings properties:
  - If you intend to take advantage of Session Persistence, verify that Persistence is set to **Database** or **Memory to Memory Replication**.
  - If you are using **Database-based persistence**:

- Check the jndi name of the datasource specified correctly on SessionManager.
- Specify correct userid and password for accessing the database.

  Note that these settings have to be checked against the properties of an existing Data Source in the admin console. The Session Manager does not automatically create a session database for you.
- The Datasource should be non-JTA, i.e. non XA enabled.
- Check the logs for appropriate database error messages.
- With DB2, for row sizes other than 4k make sure specified row size matches the DB2 page size. Make sure tablespace name is specified correctly.

– If you are using **memory-based persistence**, available in a network-deployment (multiple application server) configuration only:
  - Review [links]Memory-to-memory replication and Configuring for Memory-to-memory replication.
  - Review the **Internal Replication Domains properties** of your Session manager.

Session is shared across multiple browsers on same client machine

This behavior is browser-dependent. It varies between browser vendors, and also may change according to whether a browser is launched as a new process or as a subprocess of an existing browser session (for example by hitting Ctl-N on Windows).

The Cookie maximum age property of the Session Manager also affects this behavior, if cookies are used as the session-tracking mechanism. If the maximum age is set to some positive value, all browser instances share the cookies, which are persisted to file on the client for the specifed maximum age time.

Session is not getting invalidated immediately after specified Session timeout interval

The SessionManager invalidation process thread runs every *x* seconds to invalidate any invalid sessions, where x is determined based on the Session timeout interval specified in the Session manager properties. For the default value of 30 minutes , *x* is around 300 seconds. In this case, it could take up to 5 minutes (300 seconds) beyond the timeout threshold of 30 minutes for a particular session to become invalidated.

Unwanted sessions are being created by jsps

As required by the JavaServer Page specification, jsps by default perform a request.getSession(true), so that a session is created if none exists for the client. To prevent jsps from creating a new session, set the session scope to **false** in the jsp file using the page directive as follows:

```
<% @page session="false" %>
```

JSP source code shown by the Web server

Problem

If you share the document root of the WebSphere Application Server within the Web server document root, a security exposure can result as the Web server might display the JSP source file as plain text.

You can use the WebSphere Web server plug-in set of rules to determine whether a given request will be handled by the WebSphere Application Server. When an incoming request fails to match those rules, the Web server plug-in returns control to the Web server so that the Web server can fulfill the request. In this case, the unknown host header causes the Web server plug-in to return control to the Web server because the rules do not indicate that the WebSphere Application Server should handle it. Therefore, the Web server looks for the request in the Web server document root. Since the JSP source file is stored in the document root of the Web server, the Web server finds the file and displays it as plain text.

Suggested solution

Move the WebSphere Application Server JSP source file outside of the Web server document root. Then, when this request comes in with the unknown host header, the plug-in returns control to the Web server and the JSP source file is not found in the document root. Therefore, the Web server returns a `404 File Not Found` error rather than the JSP source file.

Problems using tracing, logging or other troubleshooting features

What kind of problem are you having?

- Error messages when launching the Log Analyzer
- Netscape browser fails when trying to enable a component trace

Error messages when launching the Log Analyzer

Upon starting the Log Analyzer for the first time or after the Log Analyzer preferences files of the users are deleted, the following message displays in the Log Analyzer shell window:

```
Cannot open input stream for waslogbrsys
```

This message is an informational message. You can disregard the message because it does not affect the execution of the Log Analyzer.

The following error messages might display in the Log Analyzer shell window when you start the Log Analyzer:

```
Cannot open input stream for default
Cannot open input stream for default
Cannot load configuration: default
Cannot open input stream for default
Cannot open input stream for default
Cannot load configuration: default
```

These error messages indicate corrupt or incomplete user preference files.

To resolve this problem, take the following steps:

1. Close the Log Analyzer.
2. Delete all user preference files in the *%USERPROFILE%\logbr* directory on Windows platforms or *$HOME/logbr* directory on UNIX platforms.
3. Restart the Log Analyzer.

Note: Deleting all user preference files removes the preferences of Log Analyzer set by the user in the preferences dialog.

Netscape browser fails when trying to enable a component trace

On systems using AIX, the Netscape browser fails when you try to enable trace on a component.

To work around this problem, do one of the following:

- Disable JavaScript on the browser and continue setting trace.
- Administer the AIX server from a remote machine running another browser and operating system.
- Change the trace manually in the *server.xml* file.

Errors connecting to the administrative console from a Netscape browser

What kind of problem are you having?

- Resizing the Netscape browser results in an error (nresize)
- Resizing the Netscape browser causes an error 404 message (404)
- Netscape screen blanks out while using the administrative console (152339)
- Resizing Netscape Version 4.7 causes errors (resizing47)
- Enabling Netscape Version 4.7 to display double-byte character set correctly (netscape479)
- Using Netscape Version 4.79 on a Solaris Operating Environment causes problems (479)
- Limitations occur when using Netscape with Solaris Operating Environment (nhas)
- Netscape browser must be capable of launching from a terminal window on all UNIX platforms (151829)

If you are able to bring up the browser page, but the console's behavior is inconsistent, error-prone, or unresponsive, try upgrading the browser you are using. Older browsers may not support the administrative console's features.

If none of these steps solves the problem, check to see if the problem has been identified and documented using the links in Chapter 12, "Diagnosing and fixing problems: Resources for learning," on page 209. If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

Resizing the Netscape browser results in an error (nresize)

If you resize your Netscape browser, you could get a ″Data Missing″ error. The error message disappears in 60 seconds.

Resizing the Netscape browser causes an error 404 message (404)

When connecting to the IBM WebSphere Application Server administrative console from a Netscape browser, resizing the browser can cause an error 404 message to occur. This situation occurs because the browser reloads the frame when resizing the window.

To avoid getting the error message, you can refrain from resizing the Netscape browser window, or you can connect to the IBM WebSphere Application Server administrative console using an Internet Explorer browser.

Netscape screen blanks out while using the administrative console (152339)

While working with the right-hand panel of the administrative console to do regular administrative tasks, the browser screen blanks out intermittently.

To work around this problem, do one of the following:

- After the problem occurs, close the Netscape browser, log in again, and continue working.
- Use the Internet Explorer browser from a Windows machine.
- Use Netscape 7.x, Mozilla 1.x, Opera 5, or Konquerer browsers on the platform, depending on which is available. Although there is not formal support for these browsers, they have all been used successfully with the product and in many cases work better than the previous 4.7.x series of Netscape browsers

Resizing Netscape Version 4.7 causes errors (resizing47)

You receive the following error messages when resizing Netscape Version 4.7:

```
Error 0
       An error occurred while processing request:
       http://localhost:9090/admin/upload.do
       message:
       Details
       com.ibm.webshpere.servlet.error.ServletErrorReport:
         at java.lang.Class.newInstance0(Native Method)
         ...
```

After resizing Netscape 4.7, Netscape has to reload the page just as it initially loads the page on the first request. For pages that do not expect POST data, it is not a problem. But for pages that do, Netscape 4.7 cannot retain the data.

Enabling Netscape Version 4.7 to display double-byte character set correctly (netscape479)

When using Netscape on AIX platforms without the translated package, the English version of Netscape is available for all locale environments as the default package. However, the English version of Netscape does not display double-byte character set (DBCS) characters on the browser radio buttons and title bars because the fonts are mismatched.

To work around this problem, you can install the message resource to make the translated version of Netscape available on a DBCS environment. Use the translated version of Netscape to display the corrupted DBCS correctly. Change the locale from English to the expected DBCS locale before starting Netscape. For example, issue the following commands to display the Japanese contents on Ja_JP (AIX Shift JIS locale):

```
$ export LANG=Ja_JP
  $ netscape&
```

Using Netscape Version 4.79 on a Solaris Operating Environment causes problems (479)

Using Netscape Version 4.79 on a Solaris Operating Environment to access the administrative console causes problems with some key text translations with the zh_TW.EUC locale. This situation is not a problem when you use Netscape Version 4.7. The officially supported version of Netscape on a Solaris Operating Environment is Version 4.79, but in this case the workaround is to use Netscape Version 4.7.

Limitations occur when using Netscape with Solaris Operating Environment (nhas)

If you click Troubleshooting > Logs and Trace >*server* > Diagnostic Trace > Modify, the window that pops up allowing you to select the Components and Groups to trace might not display a scroll bar, preventing you from viewing all the components and groups.

The text area displaying the selected components, groups, and trace levels does not have a vertical scroll bar. This omission is a limitation of Netscape on a Solaris Operating Environment.

To work around this problem, refresh the window to show the scroll bar.

Netscape browser must be capable of launching from a terminal window on all UNIX platforms (151829)

To make sure the Netscape browser can launch from a terminal window, edit or create a file called `profile` in the `/etc` directory and add the Netscape directory to the system path. For example: `PATH=$PATH:/opt/Netscape export PATH`

Chapter 4. Troubleshooting by component: what is not working?

This section provides troubleshooting information based on the task you were trying to accomplish when the problem occurred. To find more information about your problem, select a task category from the list below.

If you do not see a task that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

Installation component troubleshooting tips

If you are having problems installing the WebSphere Application Server, follow these steps to resolve the problem:

- If possible, follow the steps outlined in ″Troubleshooting the Installation″ in the WebSphere Application Server Version 5 InfoCenter.
- Browse the relevant log files for clues:
  - The main installation log file: `install_dir/log.txt`.
  - IBM Http Server log: `install_dir/ihs.log`.
  - The log file produced when the default application `.ear` file is installed is: `install_dir/logs/installDefaultApplication.log`.
- Ensure that you have installed the correct level of dependent software, such as operating system version and revision level, by reviewing `http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html`

If none of these steps solves the problem, check to see if the problem is identified and documented using the links in Chapter 12, “Diagnosing and fixing problems: Resources for learning,” on page 209. If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

Migration utility troubleshooting tips

If you are encounter problems migrating an application from a previous version of WebSphere Application Server to Version 5.0:

- Look for these log files and browse them for clues:
  - `install_dir/logs/WASPostUpgrade.time stamp.log`
  - `migration_backup_dir/WASPreUpgrade.time stamp.log`
  - `install_dir/logs/clientupgrade.time stamp.log`
- Look for **MIGR0259I: Completed successfully** or **MIGR0271W: Completed with warnings** in the `migration_backup_dir/WASPreUpgrade.time stamp.log`, `migration_backup_dir/WASPreUpgrade. time stamp.log`, or `install_dir/logs/clientupgrade. time stamp.log`.

  If **MIGR0286E: Completed with errors.** appears, attempt to correct any problems based on the error messages that appear in the log file. After correcting any errors, rerun the command from the bin directory of the product installation root. If the errors persist, rerun the command with trace enabled.
- To generate more detailed messages when running the migration tools, (enable tracing):

  - when running the WASPreUpgrade or WASPostUpgrade tools, add the following strings when you invoke them: **-traceString** ″*=all=enabled″ **-traceFile** *migration_backup_dir/filename*.
  - when running ClientUpgrade, add the following strings to the command line when you invoke it: **-traceString** ″*=all=enabled″ **-traceFile** *install_dir*/**logs**/*filename*.
- Open the Log Analyzer on the service log of the server which is hosting the resource you are trying to access and use it to browse error and warning messages.
- With WebSphere Application Server running, run the **dumpNameSpace** on Windows or **dumpNameSpace.sh** command on Unix, and pipe, redirect, or ″more″ the output so that it can be easily viewed. This command results in a display of all objects in WebSphere Application Server's namespace, including the directory path and object name.
- If the object a client needs to access does not appear, use the administrative console to verify that:
  - The server hosting the target resource is started.
  - The web module or EJB container hosting the target resource is running.
  - The JNDI name of the target resource is properly specified.
- To view detailed information on the runtime behavior of WebSphere Application Server's Naming service, (enable trace) on the following components and review the output:
  - com.ibm.ws.naming.*
  - com.ibm.websphere.naming.*

If none of these steps solves the problem, see ″Troubleshooting migration problems″ in the WebSphere Application Server Version 5 InfoCenter for tips on specific migration problems. If none of these match your problem, check to see if the problem is identified and documented using the links in Chapter 12, "Diagnosing and fixing problems: Resources for learning," on page 209. If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

Administration and administrative console troubleshooting tips

In WebSphere Application Server products, administrative functions are supported by:

- The application server (such as server1) process in the base product
- The deployment manager (dmgr) process in the Network Deployment product

The process must be running to use the administrative console. The **wsadmin** command line utility has a local mode that you can use to perform administrative functions, even when the server process is not running.

If you have problems starting or using the administrative console or wsadmin utility, verify that the supporting server process is started and that it is healthy.

- For the base product, look at these files:
  - `install_root/logs/server/startServer.log` for the message that indicates that the server started successfully: **ADMU3000I: Server server1 open for e-business; process id is** *nnnn*..

  - `install_root/logs/server/SystemOut.log` for the message that indicates that the server started successfully: **WSVR0001I: Server *server* open for e-business**.
- For the Network Deployment product, look at these files:
  - `install_root/logs/dmgr/startServer.log` for the message that indicates that the server started successfully: **ADMU3000I: Server dmgr open for e-business; process id is *nnnn*.**.
  - `install_root/logs/dmgr/SystemOut.log` for this message indicating that the server started successfully: **WSVR0001I: Server dmgr open for e-business**.
- Look up any error messages in these files in the message reference table. Select the **Quick reference** view in the InfoCenter, then click **Messages**.
- A message like **WASX7213I: This scripting client is not connected to a server process** when trying to start wsadmin indicates that either the server process is not running, the host machine where it is running is not accessible, or that the port or server name used by wsadmin is incorrect.
- Verify that you are using the right port number to communicate with the administrative console or wsadmin server using the following steps:
  - Look in the `SystemOut.log` file.
  - The line **ADMC0013I: SOAP connector available at port *nnnn*** indicates the port that the server is using to listen for wsadmin functions.
  - The property **com.ibm.ws.scripting.port** in the `install_root/properties/wsadmin.properties` file controls the port used by wsadmin to send requests to the server. If it is different from the value shown in the `SystemOut.log` file, either change the port number in the `wsadmin.properties` file, or specify the correct port number when starting wsadmin by using the **-port *port_number*** property on the command line.
  - The message **SRVE0171I: Transport http is listening on port *nnnn* (default 9090)** indicates the port the server uses to listen for administrative console requests. If it is different than the one specified in the URL for the administrative console, change the URL in the browser to the correct value. The default value is http://localhost:9090/admin.
- Use the TCP/IP **ping** command to test that the hostname where the application server or deployment manager is executing, is reachable from the system where the browser or wsadmin program are being used. If you are able to ping the hostname, this indicates that there are no firewall or connectivity issues.
- If the host where the application server or deployment manager is running is remote to the machine from which the client browser or wsadmin command is running, ensure that the appropriate hostname parameter is correct:
  - The hostname in the browser URL for the console.
  - The **-host *hostname*** option of the wsadmin command that is used to direct wsadmin to the right server
- Tracing the administrative component: WebSphere Application Server technical support might ask you to trace the administrative component for detailed problem determination. The trace specification for this component is com.ibm.websphere.management.*=all=enabled:com.ibm.ws.management.*=all=enabled″

If none of these steps solves the problem, see if the specific problem you are having is addressed in “Installation completes but the administrative console does not start” on page 5. Check to see if the problem has been identified and documented using the links in the Chapter 12, “Diagnosing and fixing problems: Resources for learning,” on page 209 topic. If you do not see a problem that

resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

Application Assembly Tool troubleshooting tips

If you are having problems installing the WebSphere Application Server Application Assembly Tool (AAT), follow these steps:

- If a problem occurs using this component, the first thing to do is to enable the printing of messages and exceptions to the screen.
  - Modify the `assembly.bat` file located in the bin directory of the product installation. Change the statement `"start javaw"` to just `"java"`.
  - Restart the AAT and a hanging command prompt window will appear through the lifetime of the Java process and display messages and exceptions.
  - Look up any error or warning messages you see in the message reference table.
- With a problem application open in the AAT, use the **Verify** menu command. This command will go through all components of the application and validate them for any XML errors or invalid entries such as missing fields, invalid bean or class references.
- To verify the integrity of an EAR (Enterprise Application Resource) file, expand it manually (outside of the AAT) by running the WebSphere Application Server `install_root\bin\EARExpander.bat` or `EARExpander.sh` file and supplying the name of the EAR file as a parameter. Browse the directory structure of the expanded EAR file to see if contains all the expected files.

  Here is an example using the Windows command prompt: `EARExpander -ear my.ear -expandDir c:\tmp\myear -operation expand`
- Contact the developer of the EAR file or its component files and ensure that they comply with J2EE specification level 1.3 and that any enterprise beans it contains conform to the EJB 2.0 Specification level.

If none of these steps solves the problem, check to see if the problem is identified and documented using the links in Chapter 12, "Diagnosing and fixing problems: Resources for learning," on page 209. If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

Web Container troubleshooting tips

If you are having problems starting a Web module, or accessing resources within a particular Web module:

- View the JVM logs and process logs for the application server which hosts the problem Web modules, and look for messages in the JVM output file which indicate that the web module has started successfully. You should see messages similar to the following:

```
WebContainer  A SRVE0161I: IBM WebSphere Application Server -
Web Container.  Copyright IBM Corp. 1998-2002
WebContainer  A SRVE0169I: Loading Web Module: [module_name]
ApplicationMg A WSVR0221I: Application started: [application_name]
HttpTransport A SRVE0171I: Transport http is listening on port [port_number]
[server_name] open for e-business in [install_root]/log/[server_name]/SystemOut.log
```

- For specific problems that can cause servlets, html files, and jsp files not to be served, see "Web resource (JSP file, servlet, HTML file, image) does not display" on page 22.
- Use the Log Analyzer tool to browse the service log (activity.log) file for clues.

- For a detailed trace of the runtime behavior of the Web container, enable trace for the component com.ibm.ws.webcontainer.*.

If none of these steps fixes your problem, check to see if the problem has been identified and documented by looking at the available online support (hints and tips, technotes, and fixes). If you don't find your problem listed there contact IBM support.

JDBC and data source troubleshooting tips

To see whether your specific problem has been addressed, review the Cannot access a data source topic in the WebSphere Application Server InfoCenter. If you cannot find your problem described there, investigate the problem further by using the following instructions to enable tracing for relevant WebSphere Application Server components.

This topic includes the following sections:

- Trace strings for JDBC data sources
- JDBC trace properties

Trace strings for JDBC data sources

Turn on JDBC tracing by using the following trace strings:

- **com.ibm.ws.database.logwriter** Trace string for databases that use the GenericDataStoreHelper. You can also use this trace string for unsupported databases.
- **com.ibm.ws.db2.logwriter** Trace string for DB2 databases.
- **com.ibm.ws.oracle.logwriter** Trace string for Oracle databases.
- **com.ibm.ws.cloudscape.logwriter** Trace string for Cloudscape databases.
- **com.ibm.ws.informix.logwriter** Trace string for Informix databases.
- **com.ibm.ws.sqlserver.logwriter** Trace string for Microsoft SQL Server databases.
- **com.ibm.ws.sybase.logwriter** Trace string for Sybase databases.

The trace group that includes the trace strings is WAS.database.

JDBC trace properties

Use a back-end database that supports JDBC tracing. Setting trace strings does not result in a trace if the database does not support JDBC tracing. The following databases offer JDBC tracing at this time:

- DB2
- Oracle
- SQL Server

Set the level of trace desired for DB2 Universal database and Oracle as custom properties on the datasource.

- **DB2 Universal JDBC driver provider** Custom properties for DB2 are:
  - **traceLevel** Possible traceLevel values are:
    - TRACE_NONE = 0
    - TRACE_CONNECTION_CALLS = 1
    - TRACE_STATEMENT_CALLS = 2

- TRACE_RESULT_SET_CALLS = 4
- TRACE_DRIVER_CONFIGURATION = 16
- TRACE_CONNECTS = 32
- TRACE_DRDA_FLOWS = 64
- TRACE_RESULT_SET_META_DATA = 128
- TRACE_PARAMETER_META_DATA = 256
- TRACE_DIAGNOSTICS = 512
- TRACE_SQLJ = 1024
- TRACE_ALL = -1

  Note: This trace level provides real data that sets to the PreparedStatement or gets from the ResultSet object.

– **traceFile** Use this property to integrate the DB2 trace with the WebSphere Application Server trace. If you do not set the value, traces are integrated. Otherwise, DB2 traces are directed to the desired file. You can dynamically enable or disable trace. You can run an application and turn on the DB2 trace if there is a problem. Use the run time trace enablement provided with the Application Server by specifying a trace string of com.ibm.ws.db2.logwriter=all=enabled.

• **Oracle JDBC provider** Custom properties for Oracle are:

– **oraclelogCategoryMask** Controls the output category. The default is 47, which is (OracleLog.USER_OPER 1 | OracleLog.PROG_ERROR 2 | OracleLog.ERROR 4 | OracleLog.WARNING 8 | OracleLog.DEBUG1 32).

  Possible values are:

  - OracleLog.USER_OPER 1
  - OracleLog.PROG_ERROR 2
  - OracleLog.ERROR 4
  - OracleLog.WARNING 8
  - OracleLog.FUNCTION 16
  - OracleLog.DEBUG1 32
  - OracleLog.SQL_STR 128

– **oraclelogModuleMask** Controls which modules write debug output. The default is 1, which is OracleLog.MODULE_DRIVER 1.

  Possible values are:

  - OracleLog.MODULE_DRIVER 1,
  - OracleLog.MODULE_DBACCESS 2

– **oraclelogPrintMask** Controls which information to print with each trace message. The default is 62, which is ([OracleLog.FIELD_OBJECT for 9i / OracleLog.FIELD_CONN for 8i] 32 | OracleLog.FIELD_CATEGORY 16 | OracleLog.FIELD_SUBMOD 8 | OracleLog.FIELD_MODULE 4 | OracleLog.FIELD_TIME 2).

  Possible values are:

  - OracleLog.FIELD_TIME 2
  - OracleLog.FIELD_MODULE 4
  - OracleLog.FIELD_SUBMOD 8
  - OracleLog.FIELD_CATEGORY 16
  - OracleLog.FIELD_OBJECT 32
  - OracleLog.FIELD_THREAD 64

Notes for Oracle JDBC tracing:

1. Oracle 9i requires the use of `classes12_g.zip` to display traces. With Oracle8i, the `classes12_g.zip` is optional.
2. You can dynamically enable or disable trace. You can run an application and turn on the Oracle trace if there is a problem. Use the run-time trace enablement provided with the WebSphere Application Server products, by specifying a trace string of com.ibm.ws.oracle.logwriter=all=enabled.

If JDBC tracing does not provide enough information to isolate and fix your problem, check to see if the problem has been identified and documented by looking at the available online support (hints and tips, technotes, and fixes). If you do not find your problem listed there, contact IBM support.

HTTP plug-in component troubleshooting tips

If you are having problems with the HTTP plug-in component - the component which sends requests from your HTTP server, such as IBM HTTP Server, Apache, Domino, iPlanet, or IIS, to the Websphere Application Server, try these steps:

- Review the file `install_dir/logs/http_plugin.log` for clues. Look up any error or warning messages in the message table.
- Review your HTTP server's error and access logs to see if the HTTP server is having a problem:
  - IBM HTTP Server and Apache: `access.log` and `error.log`.
  - Domino web server: `httpd-log` and `httpd-error`.
  - iPlanet: `access` and `error`.
  - IIS: `timedatestamp.log`.

If these files don't reveal the cause of the problem, follow these additional steps.

Plugin Problem Determination Steps

The plug-in provides very readable tracing which can be beneficial in helping to figure out the problem. By setting the **LogLevel** attribute in the `config/plugin-cfg.xml` file to **Trace**, you can follow the request processing to see what is going wrong. At a high level:

1. The plug-in gets a request.
2. The plug-in checks the routes defined in the `plugin-cfg.xml` file.
3. It finds the server group.
4. It finds the server.
5. It picks the transport protocol, usually HTTP.
6. It sends the request.
7. It reads the response.
8. It writes it back to the client.

You can see this very clearly by reading through the trace for a single request:

- The first step is to determine if the plug-in has loaded into the HTTP server successfully.
  - Check to make sure the`http_plugin.log` has been created.
  - If it has, look in it to see if any error messages indicate some sort of failure that took place during plug-in initialization. If no errors are found look for

the following stanza, which indicates that the plug-in started normally. Ensure that the timestamps for the messages correspond to the time you started the Web server:

```
[Thu Jul 11 10:59:15 2002] 0000009e 000000b1 - PLUGIN: ---------System Information-----
[Thu Jul 11 10:59:15 2002] 0000009e 000000b1 - PLUGIN: Bld date: Jul  3 2002, 15:35:09
[Thu Jul 11 10:59:15 2002] 0000009e 000000b1 - PLUGIN: Web server: IIS
[Thu Jul 11 10:59:15 2002] 0000009e 000000b1 - PLUGIN: Hostname = SWEETTJ05
[Thu Jul 11 10:59:15 2002] 0000009e 000000b1 - PLUGIN: OS version 4.0, build 1381,
                                                                  'Service Pack 6'
[Thu Jul 11 10:59:15 2002] 0000009e 000000b1 - PLUGIN: --------------------------------
```

– Some common errors are:

lib_security: loadSecurityLibrary: Failed to load gsk library
The GSK did not get installed or the installation is corrupt. If the GSK did not get installed you can determine this by searching for the file `gsk5ssl.dll` on all drives for Win32 or see if there are any `libgsk5*.so` files in `/usr/lib` on Unix. Try reinstalling the plug-in to see if you can get the GSK to install in order to fix this.

ws_transport: transportInitializeSecurity: Keyring wasn't set
The HTTPS transport defined in the configuration file was prematurely terminated and did not contain the Property definitions for the keyring and stashfile. Check your XML syntax for the line number given in the error messages that follow this one to make sure the Transport element contains definitions for the keyring and stashfiles before it is terminated.

– If the`http_plugin.log` is not created, check the Web server error log to see if any plug-in related error messages have been logged there that indicate why the plug-in is failing to load. Typical causes of this can include failing to correctly configure the plug-in with the Web server environment. Check the documentation for the Web server you are trying to use with the Web server plug-in.

• Determine whether there are network connection problems with the plug-in and the various app servers defined in the configuration. Typically you will see the following message when this is the case:

ws_common: websphereGetStream: Failed to connect to app server, OS err=%d

Where %d is an OS specific error code related to why the connect() call failed. This can happen for a variety of reasons.

– Ping the machines to make sure they are properly connected to the network. If the machines can't be pinged then there is no way the plug-in will be able to contact them. Possible reasons for this include:
  - Firewall policies limiting the traffic from the plug-in to the app server.
  - The machines are not on the same network.

– If you are able to ping the machines then the likely cause of the problem is that the port is not active. This could be because the application server or cluster has not been started or the application server has gone down for some reason. You can test this by hand by trying to telnet into the port that the connect() is failing on. If you cannot telnet into the port the app server is not up and that problem needs to be resolved before the plug-in will be able to connect() successfully.

• Determine whether other activity on the machines where the servers are installed is impairing the server's ability to service a request. Check the processor utilization as measured by the task manager, processor ID, or some other outside tool to see if it:

  - Is not what was expected.
  - Is erratic rather than a constant.
  - Shows that a newly added member of the cluster is not being utilized.
  - Shows that a failing member that has been fixed is not being utilized.
- Check the administrative console to ensure that the application servers are started. View the administrative console for error messages or look in the logs.
- In the administrative console, select the problem application server and view its installed applications to verify that they are started.

If none of these steps solves the problem:

- For specific problems that can cause web pages and their contents not to display, see ″JSP, servlet, html file, image, etc will not display″ in the WebSphere Application Server Version 5 InfoCenter.
- Check to see if the problem has been identified and documented using the links in Chapter 12, “Diagnosing and fixing problems: Resources for learning,” on page 209.
- If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

HTTP session manager troubleshooting tips

If you are having problems creating or using HTTP sessions with your Web application hosted by WebSphere Application Server, here are some steps to take:

- See ″HTTP session aren't getting created or are getting dropped″ in the WebSphere Application Server InfoCenter to see if your specific problem is discussed.
- View the ″JVM logs″ for the application server which hosts the problem application:
  - first, look at messages written while each application is starting. They will be written between the following two messages:

    ```
    Starting application: <application>
    .....................
    Application started: <application>
    ```

  - Within this block, look for any errors or exceptions containing a package name of com.ibm.ws.webcontainer.httpsession. If none are found, this is an indication that the session manager started successfully.
  - Error ″**SRVE0054E: An error occurred while loading session context and Web application**″ indicates that SessionManager didn't start properly for a given application.
  - Look within the logs for any Session Manager related messages. These messages will be in the format SESNxxxxE and SESNxxxxW for errors and warnings, respectively, where xxxx is a number identifying the precise error. Look up the extended error definitions in the Session Manager message table.
- Use the Log Analyzer tool to browse the service log (activity.log) file for clues.
- See ″Managing HTTP Sessions″.
- To dynamically view the number of sessions as a Web application is running, enable performance monitoring for HTTP sessions. This will give you an indication as to whether sessions are actually being created.
  - To learn how to enable HTTP session monitoring, see (Enabling data collection through the administrative console).

  - To learn how to view the http session counters as the application runs, see "Tivoli Performance Viewer (formerly Resource Analyzer)".
- Alternatively, a special servlet can be invoked that displays the current configuration and statistics related to session tracking. This servlet has all the counters that are in performance monitor tool and has some additional counters.
  - Servlet name: **com.ibm.ws.webcontainer.httpsession.IBMTrackerDebug**.
  - It can be invoked from any web module which is enabled to serve by class name. For example, using default_app, **http://localhost:9080/servlet/com.ibm.ws.webcontainer.httpsession.IBMTrackerDebug**.
  - If you are viewing the module via the serve-by-class-name feature, be aware that it may be viewable by anyone who can view the application. You may wish to map a specific, secured URL to the servlet instead and disable the serve-servlets-by-classname feature.
- Enable tracing for the HTTP Session Manager component:
  - Use the trace specification **com.ibm.ws.webcontainer.httpsession.*=all=enabled**. Follow the instructions for dumping and browsing the trace output to narrow the origin of the problem.
  - If you are using persistent sessions based on memory replication, also enable trace for **com.ibm.ws.drs.***.
- If you are using **database-based persistent sessions**, look for problems related to the **data source** the Session Manager relies on to keep session state information. For details on diagnosing database related problems see "Cannot access a data source" on page 29

If none of these steps fixes your problem, check to see if the problem has been identified and documented by looking at the available online support (hints and tips, technotes, and fixes)). If you don't find your problem listed there contact IBM support.

Naming services component troubleshooting tips

"Naming" is a J2EE service which publishes and provides access to resources such as connection pools, enterprise beans, message listeners, etc, to client processes. If you have problems in accessing a resource which otherwise appears to be healthy, the naming service might be involved. To investigate problems with the WebSphere Application Server Naming service:

- Browse"Viewing the JVM logs" on page 128 for the server which is hosting the resource you are trying to access. Messages starting with NMSV are related to the Naming Service.
- Open the "Log Analyzer" on page 195 on the service log of the server which is hosting the resource you are trying to access and use it to browse error and warning messages.
- With WebSphere Application Server running, run the `dumpNameSpace` command for Windows systems, or the `dumpNameSpace.sh` command for Unix systems, and pipe, redirect, or "more" the output so that it is easily viewed. This command results in a display of all objects in the WebSphere Application Server namespace, including the directory path and object name.
- If the object a client needs to access does not appear, use the administrative console to verify that:
  - The server hosting the target resource is started.

  - The Web module or EJB container, if applicable, hosting the target resource is running.
  - The jndi name of the target resource is correct and updated.
  - If the problem resource is remote, that is, not on the same node as the Name Server node, that the jndi name is fully qualified, including the host name. This is especially applicable to Network Deployment configurations
- View detailed information on the run time behavior of the WebSphere Application Server Naming service by enabling trace on the following components and reviewing the output:
  - com.ibm.ws.naming.*
  - com.ibm.websphere.naming.*
- If you see an exception that appears to be CORBA related (″CORBA″ appears as part of the exception name) look for a naming-services-specific CORBA minor code, further down in the exception stack, for information on the real cause of the problem. For a list of naming service exceptions and explanations, see the class com.ibm.websphere.naming.WsnCorbaMinorCodes in the javadoc topic in the WebSphere Application Server Version 5 InfoCenter(../javadoc/ae/index.html).

If none of these steps solve the problem:

- For specific problems that can cause access to named object hosted in WebSphere Application Server to fail, see “Cannot look up an object hosted by WebSphere Application Server from a servlet, JSP file, or other client” on page 46.
- Check to see if the problem has been identified and documented using the links in Chapter 12, “Diagnosing and fixing problems: Resources for learning,” on page 209
- If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

Messaging (JMS) component troubleshooting tips

If you are having problems deploying or executing applications which use the WebSphere Application Server messaging capabilities, review these articles in the WebSphere Application Server InfoCenter:

- ″Troubleshooting WebSphere Messaging″
- ″Troubleshooting message-driven beans″
- ″Troubleshooting transactions″

If none of these steps solves the problem, check to see if the problem has been identified and documented using the links in Chapter 12, “Diagnosing and fixing problems: Resources for learning,” on page 209. If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

Universal Discovery, Description, and Integration, Web Service, and SOAP component troubleshooting tips

If you are having problems deploying or executing applications that use WebSphere Application Server Web Services, Universal Discovery, Description, and Integration (UDDI), or SOAP, try these steps:

- Review the troubleshooting documentation for messaging in the WebSphere Application Server Version 5 InfoCenter:
  - WSIF troubleshooting tips
  - Problem determination for the UDDI
  - Problem determination for the Web Services Gateway
- Investigate the following areas for SOAP-related problems:
  - View the JVM logs for the target application server, and run the Log Analyzer on the server's service log.
  - View the error log of the HTTP server to which the SOAP request is sent.
  - View the run time behavior of the SOAP component in more detail, by enabling trace for `org.apache.soap.*` and `com.ibm.*.soap*`.
  - Browse the Web site http://xml.apache.org/soap/ for FAQs and known SOAP issues.

If none of these steps solves the problem, check to see if the problem has been identified and documented using the links in Chapter 12, "Diagnosing and fixing problems: Resources for learning," on page 209. If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

Enterprise bean and EJB container troubleshooting tips

If you are having problems starting an EJB container, or encounter error messages or exceptions that appear to be generated on by an EJB container, follow these steps to resolve the problem:

- Browse the relevant log files for clues:
  - Use the Administrative Console to verify that the application server which hosts the container is running.
  - Browse the JVM log files for the application server which hosts the container. Look for the message `server server_name open for e-business` in the `SystemOut.log`. If it does not appear, or if you see the message `problems occurred during startup`, browse the `SystemErr.log` for details.
  - Browse the system log files for the application server which hosts the container.
- Use the Log Analyzer tool to browse the service log file for more information.
- Enable tracing for the EJB Container component, by using the following trace specification `EJBContainer=all=enabled`. Follow the instructions for dumping and browsing the trace output to narrow the origin of the problem.

If none of these steps solves the problem, check to see if the problem is identified and documented using the links in Chapter 12, "Diagnosing and fixing problems: Resources for learning," on page 209. If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

Security components troubleshooting tips

This document explains basic resources and steps for diagnosing security related issues in the WebSphere Application Server, including:

- What log files to look at and what to look for in them.
- A general approach to isolating and resolving security problems.
- When and how to enable security-related trace.

- An overview and table of security-related CORBA minor codes.

The following security-related problems are addressed elsewhere in this InfoCenter:

- Errors and access problems after enabling security
- Errors after enabling SSL, or SSL-related error messages
- Errors trying to configure and enable security

If none of these steps solves the problem, check to see if the problem has been identified and documented using the links in Chapter 12, "Diagnosing and fixing problems: Resources for learning," on page 209. If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

Note: for an overview of WebSphere Application Server security components such as SAS and how they work, see "Securing applications and their environment".

Log files

When troubleshooting the security component, browse the JVM logs for the server that hosts the resource you are trying to access. The following is a sample of messages you would expect to see from a server in which the security service has started successfully:

```
SASRas        A JSAS0001I: Security configuration initialized.
SASRas        A JSAS0002I: Authentication protocol: CSIV2/IBM
SASRas        A JSAS0003I: Authentication mechanism: SWAM
SASRas        A JSAS0004I: Principal name: MYHOSTNAME/aServerID
SASRas        A JSAS0005I: SecurityCurrent registered.
SASRas        A JSAS0006I: Security connection interceptor initialized.
SASRas        A JSAS0007I: Client request interceptor registered.
SASRas        A JSAS0008I: Server request interceptor registered.
SASRas        A JSAS0009I: IOR interceptor registered.
NameServerImp I NMSV0720I: Do Security service listener registration.
SecurityCompo A SECJ0242A: Security service is starting
UserRegistryI A SECJ0136I: Custom Registry:com.ibm.ws.security.registry.
      nt.NTLocalDomainRegistryImpl has been initialized
SecurityCompo A SECJ0202A: Admin application initialized successfully
SecurityCompo A SECJ0203A: Naming application initialized successfully
SecurityCompo A SECJ0204A: Rolebased authorizer initialized successfully
SecurityCompo A SECJ0205A: Security Admin mBean registered successfully
SecurityCompo A SECJ0243A: Security service started successfully
SecurityCompo A SECJ0210A: Security enabled true
```

The following is an example of messages from a server which cannot start the security service, in this case because the administrative user ID and password given to communicate with the user registry is wrong, or the user registry itself is down or misconfigured:

```
SASRas        A JSAS0001I: Security configuration initialized.
SASRas        A JSAS0002I: Authentication protocol: CSIV2/IBM
SASRas        A JSAS0003I: Authentication mechanism: SWAM
SASRas        A JSAS0004I: Principal name: MYHOSTNAME/aServerID
SASRas        A JSAS0005I: SecurityCurrent registered.
SASRas        A JSAS0006I: Security connection interceptor initialized.
SASRas        A JSAS0007I: Client request interceptor registered.
SASRas        A JSAS0008I: Server request interceptor registered.
SASRas        A JSAS0009I: IOR interceptor registered.
NameServerImp I NMSV0720I: Do Security service listener registration.
SecurityCompo A SECJ0242A: Security service is starting
UserRegistryI A SECJ0136I: Custom Registry:com.ibm.ws.security.registry.
```

```
      nt.NTLocalDomainRegistryImpl has been initialized
Authenticatio E SECJ4001E: Login failed for badID/<null> javax.security.auth.
       login.LoginException: authentication failed: bad user/password
```

The following is an example of messages from a server for which LDAP has been specified as the security mechanism, but the LDAP keys have not been properly configured:

```
SASRas        A JSAS0001I: Security configuration initialized.
SASRas        A JSAS0002I: Authentication protocol: CSIV2/IBM
SASRas        A JSAS0003I: Authentication mechanism: LTPA
SASRas        A JSAS0004I: Principal name: MYHOSTNAME/anID
SASRas        A JSAS0005I: SecurityCurrent registered.
SASRas        A JSAS0006I: Security connection interceptor initialized.
SASRas        A JSAS0007I: Client request interceptor registered.
SASRas        A JSAS0008I: Server request interceptor registered.
SASRas        A JSAS0009I: IOR interceptor registered.
NameServerImp I NMSV0720I: Do Security service listener registration.
SecurityCompo A SECJ0242A: Security service is starting
UserRegistryI A SECJ0136I: Custom Registry:com.ibm.ws.security.registry.nt.
    NTLocalDomainRegistryImpl has been initialized
SecurityServe E SECJ0237E: One or more vital LTPAServerObject configuration
         attributes are null or not available. The attributes and values are password :
         LTPA password does exist, expiration time 30, private key <null>,
         public key <null>, and shared key <null>.
```

A problem with the SSL configuration might lead to the following message. You should ensure that the keystore location and keystore passwords are valid. Also, ensure the keystore has a valid personal certificate and that the personal certificate public key or CA root has been extracted on put into the truststore.

```
SASRas        A JSAS0001I: Security configuration initialized.
SASRas        A JSAS0002I: Authentication protocol: CSIV2/IBM
SASRas        A JSAS0003I: Authentication mechanism: SWAM
SASRas        A JSAS0004I: Principal name: MYHOSTNAME/aServerId
SASRas        A JSAS0005I: SecurityCurrent registered.
SASRas        A JSAS0006I: Security connection interceptor initialized.
SASRas        A JSAS0007I: Client request interceptor registered.
SASRas        A JSAS0008I: Server request interceptor registered.
SASRas        A JSAS0009I: IOR interceptor registered.
SASRas        E JSAS0026E: [SecurityTaggedComponentAssistorImpl.register]
Exception connecting object to the ORB.  Check the SSL configuration to
ensure that the SSL keyStore and trustStore properties are set properly.
If the problem persists, contact support for assistance. org.omg.CORBA.OBJ_ADAPTER:
ORB_CONNECT_ERROR (5) - couldn't get Server Subcontract  minor code: 4942FB8F
completed: No
```

General approach for troubleshooting security-related issues

When troubleshooting security-related problems, the following questions are very helpful and should be considered:

Does the problem occur when security is disabled?

This is a good litmus test to determine that a problem is security related. However, just because a problem only occurs when security is enabled does not always make it a security problem. More troubleshooting is necessary to ensure the problem is really security-related.

Did security appear to initialize properly?

A lot of security code is visited during initialization. So you will likely see problems there first if the problem is configuration related. The following sequence of messages generated in the SystemOut.log indicate normal code initialization of an application server. This will vary based on the configuration, but the messages are similar:

```
SASRas        A JSAS0001I: Security configuration initialized.
SASRas        A JSAS0002I: Authentication protocol: CSIV2/IBM
SASRas        A JSAS0003I: Authentication mechanism: SWAM
SASRas        A JSAS0004I: Principal name: BIRKT20/pbirk
SASRas        A JSAS0005I: SecurityCurrent registered.
SASRas        A JSAS0006I: Security connection interceptor initialized.
SASRas        A JSAS0007I: Client request interceptor registered.
SASRas        A JSAS0008I: Server request interceptor registered.
SASRas        A JSAS0009I: IOR interceptor registered.
NameServerImp I NMSV0720I: Do Security service listener registration.
SecurityCompo A SECJ0242A: Security service is starting
UserRegistryI A SECJ0136I: Custom Registry:com.ibm.ws.security.registry.
                      nt.NTLocalDomainRegistryImpl has been initialized
SecurityCompo A SECJ0202A: Admin application initialized successfully
SecurityCompo A SECJ0203A: Naming application initialized successfully
SecurityCompo A SECJ0204A: Rolebased authorizer initialized successfully
SecurityCompo A SECJ0205A: Security Admin mBean registered successfully
SecurityCompo A SECJ0243A: Security service started successfully
SecurityCompo A SECJ0210A: Security enabled true
```

Is there a stack trace or exception printed in the SystemOut.log?

A single stack trace tells a lot about the problem. What code initiated the code that failed? What is the failing component? Which class did the failure actually come from? Sometimes the stack trace is all that is needed to solve the problem and it can pinpoint the root cause. Other times, it can only give us a clue, and could actually be misleading. When support analyzes a stack trace, they may request additional trace if it is not clear what the problem is. If it appears to be security related and the solution cannot be determined from the stack trace or problem description, you will be asked to to gather the following trace specification specification: `SASRas=all=enabled:com.ibm.ws.security.*=all=enabled` from all processes involved.

Is this a distributed security problem or a local security problem?

- If the problem is local, that is the code involved does not make a remote method invocation, then troubleshooting is isolated to a single process. It is important to know when a problem is local versus distributed since the behavior of the Orb, among other components, is different between the two. Once a remote method invocation takes place, an entirely different security code path is entered.
- When you know that the problem involves two or more servers, the techniques of troubleshooting change. You will need to trace all servers involved simultaneously so that the trace shows the client and server sides of the problem. Try to make sure the timestamps on all machines match as closely as possible so that you can find the request and reply pair from two different processes. Enable both SAS and Security trace using the trace specification using the trace specification: `SASRas=all=enabled:com.ibm.ws.security.*=all=enabled`.

Is the problem related to authentication or authorization?

Most security problems fall under one of these two categories. Authentication is the process of determing who the caller is. Authorization is the process of validating that the caller has the proper authority to invoke the requested method. When authentication fails, typically this is related to either the authentication protocol, authentication mechanism or user registry. When authorization fails, this is usually related to the application bindings from assembly and/or deployment and to the caller's identity who is accessing the method and the roles required by the method.

Is this a Web or EJB request?

Web requests have a completely different code path than EJB requests. Also, there are different security features for Web requests than for EJB requests, requiring a completely different body of knowledge to resolve. For example, when using the LTPA authentication mechanism, the Single SignOn feature is available for Web requests but not for EJB requests. Web requests involve HTTP header information not required by EJB requests due to the protocol differences. Also, the Web container (or servlet engine) is involved in the entire process. Any of these components could be involved in the problem and all should be considered during troubleshooting, based on the type of request and where the failure occurs.

Secure EJB requests heavily involve the ORB and Naming components since they flow over the RMI/IIOP protocol. In addition, when work flow management (WLM) is enabled, other behavior changes in the code can be observed. All of these components interact closely for security to work properly in this environment. At times, trace in any or all of these components might be necessary to troubleshoot problems in this area. The trace specification to begin with is `SASRas=all=enabled:com.ibm.ws.security.*=all=enabled`. ORB trace is also very beneficial when the SAS/Security trace does not seem to pinpoint the problem.

Does the problem seem to be related to SSL?

The Secure Socket Layer is just that, a totally distinct, separate layer of security. Troubleshooting SSL problems are usually separate from troubleshooting authentication and/or authorization problems. There are many things to consider. Usually, SSL problems are first time setup problems because the configuration can be difficult. Each client must contain the server's signer certificate. During mutual authentication, each server must contain the client's signer certificate. Also, there can be protocol differences (SSLv3 vs. TLS), and listener port problems related to stale IORs (i.e., IORs from a server reflecting the port prior to the server restarting).

For SSL problems, we sometimes request an SSL trace to determine what is happening with the SSL handshake. The SSL handshake is the process which occurs when a client opens a socket to a server. If anything goes wrong with the key exchange, cipher exchange, etc. the handshake will fail and thus the socket is invalid. Tracing JSSE (the SSL implementation used in WebSphere Application Server) involves the following steps:

- Ensure that the client and server processes contain an `ibmjsse-debug.jar` file in the `java/jre/lib/ext` directory. The `ibmjsse-debug.jar` is shipped with the product. You can locate the file under `installation_directory\web\docs\jsse`. Make sure you remove the existing `ibmjsse.jar` file from this directory after putting in the `ibmjsse-debug.jar`. If both exist in the `/ext` directory, the JSSE trace will not be complete.
- Set the following system property on the client and server processes: `-Djavax.net.debug=true`. For the server, add this to the (Generic JVM Arguments) property of the Java virtual machine settings page.
- Turn on ORB trace as well.
- Recreate the problem. The `SystemOut.log` of both processes should contain the JSSE trace. You will find trace similar to the following:

```
SSLConnection: install <com.ibm.sslite.e@3ae78375>
>> handleHandshakeV2 <com.ibm.sslite.e@3ae78375>
>> handshakeV2 type = 1
```

```
>> clientHello: SSLv2.
SSL client version: 3.0
...
...
...
JSSEContext: handleSession[Socket[addr=null,port=0,localport=0]]

<< sendServerHello.
SSL version: 3.0
SSL_RSA_WITH_RC4_128_MD5
HelloRandom
...
...
...
<< sendCertificate.
<< sendServerHelloDone.
>> handleData <com.ibm.sslite.e@3ae78375>
>> handleHandshake <com.ibm.sslite.e@3ae78375>
>> handshakeV3 type = 16
>> clientKeyExchange.
>> handleData <com.ibm.sslite.e@3ae78375>
>> handleChangeCipherSpec <com.ibm.sslite.e@3ae78375>
>> handleData <com.ibm.sslite.e@3ae78375>
>> handleHandshake <com.ibm.sslite.e@3ae78375>
>> handshakeV3 type = 20
>> finished.
<< sendChangeCipherSpec.
<< sendFinished.
```

Tracing security

The classes which implement WebSphere Application Server security are:

- com.ibm.ws.security.*
- com.ibm.websphere.security.*
- com.ibm.WebSphereSecurityImpl.*
- SASRas

To view detailed information on the runtime behavior of security, enabling trace on the following components and review the output:

- `com.ibm.ws.security.*=all=enabled:com.ibm:WebSphereSecurityImpl.*=all =enabled:com.ibm.websphere.security.*=all=enabled`

  This trace statement collects the trace for the security runtime.
- `com.ibm.ws.console.security.*=all=enabled`. This trace statement collects the trace for the security center GUI.
- `SASRas=all=enabled`. This trace statement collects the trace for SAS (low-level authentication logic).

Fine tuning SAS traces:

If a subset of classes need to be traced for the SAS/CSIv2 component, a system property can be specified with the class names comma separated: `com.ibm.CORBA.securityTraceFilter=SecurityConnectionInterceptorImpl, VaultImpl, ...`

Fine tuning Security traces:

If a subset of packages need to be traced, specify a trace specification more detailed than `com.ibm.ws.security.*=all=enabled`. For example, to trace just dynamic policy code, you can specify `com.ibm.ws.security.policy.*=all=enabled`. To disable dynamic policy trace, you can specify `com.ibm.ws.security.policy.*=all=disabled`.

Configuring CSIv2 or SAS Trace Settings

Situations arise where reviewing trace for the CSIv2 or SAS authentication protocols can assist in troubleshooting difficult problems. This section decribes how to enable to CSIv2/SAS trace.

Enabling Client-Side CSIv2/SAS Trace

To enable CSIv2 and SAS trace on a pure client, the following steps need to be taken:

- Edit the file TraceSettings.properties in the **/WebSphere/AppServer/properties directory**.
- In this file, change `traceFileName=` to point to the path in which you want the ouput file created. Make sure you put a double backslash (`\\`) between each subdirectory. For example, `traceFileName=c:\\WebSphere\\AppServer\\logs\\sas_client.log`
- In this file, add the trace specification string: `SASRas=all=enabled`. Any additional trace strings can be added on separate lines.
- Point to this file from within your client application. On the Java command line where you launch the client, add the following system property: `-DtraceSettingsFile=TraceSettings.properties`.

  Note: Do not give the fully qualified path to the `TraceSettings.properties` file. Make sure that the `TraceSettings.properties` file is in your classpath.

Enabling Server-Side CSIv2/SAS Trace

To enable SAS trace in an application server, complete the following:

- Add the trace specification, `SASRas=all=enabled`, to the `server.xml` file or add it to the Trace settings within the WebConsole GUI.
- Typically it is best to also trace the authorization security runtime in addition to the authentication protocol runtime. To do this, use the following two trace specifications in combination: `SASRas=all=enabled:com.ibm.ws.security.*=all=enabled`.
- When troubleshooting a connection type problem, it is beneficial to trace both SAS/CSIv2 and the ORB. To do this, use the following three trace specifications:

  ```
  SASRas=all=enabled:com.ibm.ws.security.*=all=
  enabled:ORBRas=all=enabled
  ```

  .
- In addition to adding these trace specifications, for ORB trace there are a couple of system properties that also need to be set. Go to the ORB settings in the GUI and add the following two properties: `com.ibm.CORBA.Debug=true` and `com.ibm.CORBA.CommTrace=true`.

CSIv2 CORBA Minor Codes

Whatever exceptions might occur within the security code on either the client or server, the eventual exception will become a CORBA exception. So any exception that occurs gets "wrapped" by a CORBA exception, because the CORBA

architecture is used by the security service for its own inter-process communication. CORBA exceptions are generic, and indicate a problem in communication between two components. CORBA minor codes are more specific, and indicate the underlying reason that a component could not complete a request.

The following shows the CORBA Minor codes which a client can expect to receive after executing a security-related request such as authentication. It also includes the CORBA exception type that the minor code would appear in.

The following exception shows an example of a CORBA exception where the minor code is 49424300. From the table below, this minor code indicates Authentication Failure. Typically, a descriptive message is also included in the exception to assist in troubleshooting the problem. Here, the detailed message is ″Exception caught invoking authenticateBasicAuthData from SecurityServer for user jdoe. Reason: com.ibm.WebSphereSecurity.AuthenticationFailedException″ which indicates that the authentication failed for user ″jdoe″.

The completed field in the exception indicates whether the method was completed or not. In the case of a NO_PERMISSION, the method should never get invoked, so it will always be ″completed:No″. Other exceptions which are caught on the server side could have a completed status of ″Maybe″ or ″Yes″.

```
org.omg.CORBA.NO_PERMISSION: Caught WSSecurityContextException
in WSSecurityContext.acceptSecContext(),
reason: Major Code[0] Minor Code[0] Message[Exception caught i
nvoking authenticateBasicAuthData from SecurityServer for user jdoe.
Reason: com.ibm.WebSphereSecurity.AuthenticationFailedException]
minor code: 49424300  completed: No
at com.ibm.ISecurityLocalObjectBaseL13Impl.PrincipalAuthFailReason.
map_auth_fail_to_minor_code(PrincipalAuthFailReason.java:83)
        at com.ibm.ISecurityLocalObjectBaseL13Impl.CSIServerRI.
           receive_request(CSIServerRI.java:1569)
        at com.ibm.rmi.pi.InterceptorManager.iterateReceiveRequest
           (InterceptorManager.java:739)
        at com.ibm.CORBA.iiop.ServerDelegate.dispatch
           (ServerDelegate.java:398)
        at com.ibm.rmi.iiop.ORB.process(ORB.java:313)
        at com.ibm.CORBA.iiop.ORB.process(ORB.java:1581)
        at com.ibm.rmi.iiop.GIOPConnection.doWork(GIOPConnection.java:1827)
        at com.ibm.rmi.iiop.WorkUnitImpl.doWork(WorkUnitImpl.java:81)
        at com.ibm.ejs.oa.pool.PooledThread.run(ThreadPool.java:91)
        at com.ibm.ws.util.CachedThread.run(ThreadPool.java:149)
```

The following table shows the CORBA Minor codes which a client can expect to receive after executing a security-related request such as authentication. It also includes the CORBA exception type that the minor code would appear in.

Minor code name	Minor code value (in hex)	Exception type (all in the package of org.omg.CORBA.*)	Minor code description	Retry performed (when authenticationRetryEnabled=true)

AuthenticationFailed	49424300	NO_PERMISSION	This is a generic authentication failed error. It does not give any details about whether the userid or password is invalid. Some registries can choose to use this type of error code, others might choose to use the next three types which are more specific.	Yes
InvalidUserid	49424301	NO_PERMISSION	This occurs when the registry returns bad userid.	Yes
InvalidPassword	49424302	NO_PERMISSION	This occurs when the registry returns bad password.	Yes
InvalidSecurity Credentials	49424303	NO_PERMISSION	This is a generic error indicating that the credentials are bad for whatever reason. It could be that they don't have the right attributes set.	Yes, if client has BasicAuth credential (token based credential was rejected in the first place).
InvalidRealm	49424304	NO_PERMISSION	This occurs when the REALM in the token received from the client does not match the server's current realm.	No

ValidationFailed	49424305	NO_PERMISSION	A validation failure occurs when a token is sent from the client or server to a target server but the token format or the expiration is invalid.	Yes, if client has BasicAuth credential (token based credential was rejected in the first place).
CredentialToken Expired	49424306	NO_PERMISSION	This is more specific about why the validation failed. In this case, the token has a absolute lifetime, and this lifetime has expired. Therefore, it is no longer a valid token and cannot be used.	Yes, if client has BasicAuth credential (token based credential was rejected in the first place).
InvalidCredential Token	49424307	NO_PERMISSION	This is more specific about why the validation failed. In this case, the token cannot be decrypted or the data within it is not readable.	Yes, if client has BasicAuth credential (token based credential was rejected in the first place).
SessionDoesNotExist	49424308	NO_PERMISSION	This indicates that the CSIv2 session does not exist on the server. Typically, a retry occurs automatically and will successfully create a new session.	Yes

SessionConflicting Evidence	49424309	NO_PERMISSION	This indicates that a session already exists on the server which matches the context_id sent over by the client, however, the information provided by the client for this EstablishContext message is different from the information originally provided to establish the session.	Yes
SessionRejected	4942430A	NO_PERMISSION	This indicates that the session referenced by the client has been previously rejected by the server.	Yes
SecurityServerNot Available	4942430B	NO_PERMISSION	This error occurs when the server cannot contact the security server (whether local or remote) in order to authenticate or validate.	No
InvalidIdentityToken	4942430C	NO_PERMISSION	This error indicates that identity cannot be obtained from the identity token when Identity Assertion is enabled.	No

IdentityServerNot Trusted	4942430D	NO_PERMISSION	This indicates that the server id of the sending server is not on the target server's trusted principal list.	No
InvalidMessage	4942430E	NO_PERMISSION	This indicates that the CSIv2 message format is invalid for the receiving server.	No
AuthenticationNot Supported	49421090	NO_PERMISSION	This error occurs when a mechanism does not support authentication (very rare).	No
InvalidSecurity Mechanism	49421091	NO_PERMISSION	This is used to indicate that the specified security mechanism is not known.	No
CredentialNotAvailable	49421092	NO_PERMISSION	This indicates a credential is not available when it is required.	No
SecurityMechanismNot Supported	49421093	NO_PERMISSION	This error occurs when a security mechanism specified in the CSIv2 token is not implemented on the server.	No

ValidationNot Supported	49421094	NO_PERMISSION	This error occurs when a mechanism does not support validation (such as LocalOS). This error should not occur since the LocalOS credential is not a forwardable credential, therefore, validation should never need to be called on it.	No
CredentialTokenNotSet	49421095	NO_PERMISSION	This is used to indicate the token inside the credential is null.	No
ServerConnectionFailed	494210A0	COMM_FAILURE	This error is used when a connection attempt fails.	Yes (via Orb retry)
CorbaSystemException	494210B0	INTERNAL	This is a generic Corba specific exception in system code.	No
JavaException	494210B1	INTERNAL	This is a generic error that indicated an unexpected Java exception occurred.	No
ValueIsNull	494210B2	INTERNAL	This is used to indicate that a value or parameter passed in was null.	No
EffectivePolicyNot Present	494210B3	INTERNAL	This indicates that an effective policy object for CSIv2 is not present. This object is used to determine what security configuration features have been specified.	No

NullPointerException	494210B4	INTERNAL	This is used to indicate that a NullPointerException was caught in the runtime.	No
ErrorGettingClass Instance	494210B5	INTERNAL	This indicates a problem loading a class dynamically.	No
MalFormedParameters	494210B6	INTERNAL	This indicates parameters are not valid.	No
DuplicateSecurity AttributeType	494210B7	INTERNAL	A duplicate credential attribute has been specified during the set_attributes operation.	No
MethodNotImplemented	494210C0	NO_IMPLEMENT	A method invoked has not been implemented.	No
GSSFormatError	494210C5	BAD_PARAM	This indicates that a GSS encoding or decoding routine has thrown an exception.	No
TagComponentFormat Error	494210C6	BAD_PARAM	This indicates that a tag component cannot be read properly.	No
InvalidSecurityAttribute Type	494210C7	BAD_PARAM	This indicates an attribute type specified during the set_attributes operation is an invalid type.	No
SecurityConfigError	494210CA	INITIALIZE	A problem exists between the client and server configuration.	No

JSP engine troubleshooting tips

If you are having difficulty using the JSP engine, try these steps:

1. Determine whether other resources such as `.html` files or servlets are being requested and displayed correctly. If they are not, the problem probably lies at a deeper level, such as with the HTTP server.
2. If other resources are being displayed correctly, determine whether the JSP engine has started normally:

- Browse the logs of the server hosting the JSP files you are trying to access. A message such as `application_name/Sevlet.LOG: JSP 1.2 Processor: init"` in the `root_dir/logs/ server_name/SystemOut.log` file indicates that the JSP engine has started normally. If the JSP processor fails to load, you may see a message such as `Did not realize init() exception thrown by servlet JSP 1.2 Processor in application_name/Servlet.LOG: JSP 1.2 Processor: init"` in the `root_dir/logs/ server_name/SystemOut.log` file.
- Open the Log Analyzer on the service log of the server which is hosting the jsp you are trying to access and use it to browse error and warning messages.

3. If the JSP engine has started normally, the problem may be with the JSP file itself.
   - Copy a simple JSP file (such as the WebSphere Application Server sample "HelloHTML.jsp") to the Web application's document root and attempt to serve it.
   - If that works, examine the target application server's `SystemOut.log` for invalid JSP directive syntax . Errors similar to the following in a browser indicate this kind of problem: `Message: /jspname.jsp(9,0) Include: Mandatory attribute page missing`. This example indicates that line 9, column 0 of the named JSP is missing a mandatory page attribute. Similar messages are displayed for other syntax errors.
   - Examine the target application server's `SystemErr.log` files for problems with invalid Java syntax. Errors similar to `Message: Unable to compile class for JSP` in a browser indicate this kind of problem.

     The error message output from the Javac compiler will be found in the `SystemErr.log`. It might look like:

     ```
     C:\WASROOT\temp\ ... test.war\_myJsp.java:14:
     Duplicate variable declaration: int myInt was int myInt
          int myInt = 122; String myString = "number is 122";
     static int myStaticInt=22; int myInt=121;
                                        ^ 1 error
     ```

     Correct the error in the JSP file and retry the file.

If none of these steps solves the problem, check to see if the problem is identified and documented using the links in Chapter 12, "Diagnosing and fixing problems: Resources for learning," on page 209. If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

Object request broker component troubleshooting tips

This article describes how to diagnose problems related to the WebSphere Application Server Object Request Broker (ORB) component by explaining:

- How to enable tracing for the ORB component.
- What log files to examine for more information.
- Information on the Java packages containing the ORB Service.
- ORB-related tools.
- Where to find configurable settings.
- A listing of CORBA minor codes generated by this component.

Enabling tracing for the Object Request Broker component

The object request broker (ORB) service is one of the WebSphere Application Server run time services. Tracing of messages sent and received by the ORB is a useful starting point for troubleshooting the ORB service. You can selectively enable or disable tracing of ORB messages for each server in a WebSphere Application Server installation, and for each application client.

This tracing is referred to by WebSphere Application Server support as a *comm trace*, and is different from the general purpose trace facility. The trace facility, which shows the detailed run time behavior of product components, may be used alongside comm trace for other product components, or for the ORB component. The trace string associated with the ORB service is `"ORBRas=all=enabled"`.

You can enable and disable comm tracing using the administrative console or by manually editing the **server.xml** file for the server be trace. You must stop and restart the server for the configuration change to take effect.

For example, using the administrative console:

- Navigate to the desired server by clicking **Servers > Application Servers > server1 > ORB Service**,, and select the ORB tracing checkbox. Click **OK**, and then click **Save** to save your settings. Restart the server for the new settings to take effect. Or,
- Locate the `server.xml` file for the selected server, for example: `install_dir/config/cells/nodename/nodes/nodename/servers/servername/server.xml`.
- Locate the services entry for the ORB service (`xmi:type="orb:ObjectRequestBroker"`) and set **commTraceEnabled="true".**

To enable ORB comm tracing for client applications, you must specify two ORB properties in the command line used to launch the client application:

- If you are using the WebSphere Application Server launcher, launchClient, use the option **-CCD** or
- If you are using the **java** command directly, use the -D option to specify these parameters:
  - `com.ibm.CORBA.Debug=true`
  - `com.ibm.CORBA.CommTrace=true`

Log files and messages associated with Object Request Broker

Messages and trace information for the ORB are captured primarily in two logs:

- The `install_dir/logs/servername/trace.log` file for output from communications tracing and tracing the behavior of the ORBRas component
- The JVM logs for each application server, for WebSphere Application Server error and warning messages

The following message in the `SystemOut.log` file indicates the successful start of the Application Server and its ORB service:

WSVR0001I: Server server1 open for e-business

When communications tracing is enabled, a message similar to the following example in the `install_dir/logs/servername/trace.log` file, indicates that the ORB service has started successfully. The message also shows the start of a listener thread, which is waiting for requests on the specified local port.

com.ibm.ws.orbimpl.transport.WSTransport startListening(ServerConnectionData connectionData) P=693799:O=0:CT a new ListenerThread has been started for ServerSocket[addr=0.0.0.0/0.0.0.0,port=0,localport=1360]

If tracing of the Object Adapter has been enabled (`com.ibm.ejs.oa.*=all=enabled`), the following message in the trace.log indicates that the ORB service has started successfully:

EJSORBImpl < initializeORB

The ORB service is one of the first services started during the WebSphere Application Server initialization process. If it is not properly configured, other components such as naming, security, and node agent, are not likely to start successfully. This is obvious in the JVM logs or `trace.log` of the affected application server.

Java packages containing the Object Request Broker service

The ORB service resides in the following Java packages:

- com.ibm.com.CORBA.*
- com.ibm.rmi.*
- com.ibm.ws.orb.*
- com.ibm.ws.orbimpl.*
- org.omg.CORBA.*
- javax.rmi.CORBA.*

JAR files that contain the previously mentioned packages include:

- `install_dir/java/jre/lib/ext/ibmorb.jar`
- `install_dir/java/jre/lib/ext/iwsorbutil.jar`
- `install_dir/lib/iwsorb.jar`

Tools used with Object Request Broker

The tools used to compile Java remote interfaces to generate language bindings used by the ORB at runtime reside in the following Java packages:

- com.ibm.tools.rmic.*
- com.ibm.idl.*

The JAR file that contains the packages is `install_dir/java/lib/ibmtools.jar`.

Object Request Broker properties

The ORB service requires a number of ORB properties for correct operation. It is not necessary for most users to modify these properties, and it is recommended that only your system administrator modify them when required. Consult IBM Support personnel for assistance. The properties reside in the `orb.properties` file, located at `install_dir/java/jre/lib/orb.properties`.

CORBA minor codes

The CORBA specification defines standard minor exception codes for use by the ORB when a system exception is thrown. In addition, the object management group (OMG) assigns each vendor a unique prefix value for use in

vendor-proprietary minor exception codes. Minor code values assigned to IBM and used by the ORB in the WebSphere Application Server follow. The minor code value is in decimal and hexadecimal formats. The column labeled minor code reason gives a short description of the condition causing the exception. Currently there is no documentation for these errors beyond the minor code reason. If you require technical support from IBM, the minor code helps support engineers determine the source of the problem.

Decimal minor exception codes 1229066320 through 1229124228

Decimal	Hexadecimal	Minor code reason
1229066320	0x49421050	HTTPOUTPUTSTREAM_WRITE
1229066321	0x49421051	COULD_NOT_INSTANTIATE_CLIENT _SSL_SOCKET_FACTORY
1229066322	0x49421052	COULD_NOT_INSTANTIATE_SERVER _SSL_SOCKET_FACTORY
1229066323	0x49421053	CREATE_LISTENER_FAILED_1
1229066324	0x49421054	CREATE_LISTENER_FAILED_2
1229066325	0x49421055	CREATE_LISTENER_FAILED_3
1229066326	0x49421056	CREATE_LISTENER_FAILED_4
1229066327	0x49421057	CREATE_LISTENER_FAILED_5
1229066328	0x49421058	INVALID_CONNECTION_TYPE
1229066329	0x49421059	HTTPINPUTSTREAM_NO_ACTIVEINPUTSTREAM
1229066330	0x4942105a	HTTPOUTPUTSTREAM_NO_ OUTPUTSTREAM
1229066331	0x4942105b	CONNECTIONINTERCEPTOR_INVALID_ CLASSNAME
1229066332	0x4942105c	NO_CONNECTIONDATA_IN_ CONNECTIONDATACARRIER
1229066333	0x4942105d	CLIENT_CONNECTIONDATA_IS_ INVALID_TYPE
1229066334	0x4942105e	SERVER_CONNECTIONDATA_IS_ INVALID_TYPE
1229066335	0x4942105f	NO_OVERLAP_OF_ENABLED_AND_ DESIRED_CIPHER_SUITES
1229066352	0x49421070	CONNECT_FAILURE_ON_SSL_ CLIENT_SOCKET
1229066353	0x49421071	GETCONNECTION_KEY_ RETURNED_FALSE
1229066354	0x49421072	UNABLE_TO_CREATE_ SSL_SOCKET
1229066355	0x49421073	SSLSERVERSOCKET_TARGET_ SUPPORTS_LESS_THAN_1
1229066356	0x49421074	SSLSERVERSOCKET_TARGET_ REQUIRES_LESS_THAN_1
1229066357	0x49421075	SSLSERVERSOCKET_TARGET_ LESS_THAN_TARGET_REQUIRES
1229066358	0x49421076	UNABLE_TO_CREATE_SSL_ SERVER_SOCKET
1229066359	0x49421077	CAUGHT_EXCEPTION_WHILE_ CONFIGURING_SSL_SERVER_SOCKET
1229066360	0x49421078	INVALID_SERVER_CONNECTION_ DATA_TYPE
1229066361	0x49421079	GETSERVERCONNECTIONDATA_ RETURNED_NULL
1229066362	0x4942107a	GET_SSL_SESSION_RETURNED_NULL
1229066363	0x4942107b	GLOBAL_ORB_EXISTS

Decimal	Hexadecimal	Minor code reason
1229123841	0x4942f101	DSIMETHOD_NOTCALLED
1229123842	0x4942f102	BAD_INV_PARAMS
1229123843	0x4942f103	BAD_INV_RESULT
1229123844	0x4942f104	BAD_INV_CTX
1229123879	0x4942f127	PI_NOT_POST_INIT
1229123969	0x4942f181	BAD_OPERATION_EXTRACT_SHORT
1229123970	0x4942f182	BAD_OPERATION_EXTRACT_LONG
1229123971	0x4942f183	BAD_OPERATION_EXTRACT_USHORT
1229123972	0x4942f184	BAD_OPERATION_EXTRACT_ULONG
1229123973	0x4942f185	BAD_OPERATION_EXTRACT_FLOAT
1229123974	0x4942f186	BAD_OPERATION_EXTRACT_ DOUBLE
1229123975	0x4942f187	BAD_OPERATION_EXTRACT_ LONGLONG
1229123976	0x4942f188	BAD_OPERATION_EXTRACT_ ULONGLONG
1229123977	0x4942f189	BAD_OPERATION_EXTRACT_ BOOLEAN
1229123978	0x4942f18a	BAD_OPERATION_EXTRACT_CHAR
1229123979	0x4942f18b	BAD_OPERATION_EXTRACT_OCTET
1229123980	0x4942f18c	BAD_OPERATION_EXTRACT_WCHAR
1229123981	0x4942f18d	BAD_OPERATION_EXTRACT_STRING
1229123982	0x4942f18e	BAD_OPERATION_EXTRACT_WSTRING
1229123983	0x4942f18f	BAD_OPERATION_EXTRACT_ANY
1229123984	0x4942f190	BAD_OPERATION_INSERT_OBJECT_1
1229123985	0x4942f191	BAD_OPERATION_INSERT_OBJECT_2
1229123986	0x4942f192	BAD_OPERATION_EXTRACT_OBJECT_1
1229123987	0x4942f193	BAD_OPERATION_EXTRACT_OBJECT_2
1229123988	0x4942f194	BAD_OPERATION_EXTRACT_TYPECODE
1229123989	0x4942f195	BAD_OPERATION_EXTRACT_PRINCIPAL
1229123990	0x4942f196	BAD_OPERATION_EXTRACT_VALUE
1229123991	0x4942f197	BAD_OPERATION_GET_PRIMITIVE_TC_1
1229123992	0x4942f198	BAD_OPERATION_GET_PRIMITIVE_TC_2
1229123993	0x4942f199	BAD_OPERATION_INVOKE_NULL_PARAM_1
1229123994	0x4942f19a	BAD_OPERATION_INVOKE_NULL_PARAM_2
1229123995	0x4942f19b	BAD_OPERATION_INVOKE_DEFAULT_1
1229123996	0x4942f19c	BAD_OPERATION_INVOKE_DEFAULT_2
1229123997	0x4942f19d	BAD_OPERATION_UNKNOWN_ BOOTSTRAP_METHOD
1229124097	0x4942f201	NULL_PARAM_1
1229124098	0x4942f202	NULL_PARAM_2
1229124099	0x4942f203	NULL_PARAM_3
1229124100	0x4942f204	NULL_PARAM_4
1229124101	0x4942f205	NULL_PARAM_5
1229124102	0x4942f206	NULL_PARAM_6
1229124103	0x4942f207	NULL_PARAM_7
1229124104	0x4942f208	NULL_PARAM_8
1229124105	0x4942f209	NULL_PARAM_9
1229124106	0x4942f20a	NULL_PARAM_10
1229124107	0x4942f20b	NULL_PARAM_11
1229124108	0x4942f20c	NULL_PARAM_12
1229124109	0x4942f20d	NULL_PARAM_13
1229124110	0x4942f20e	NULL_PARAM_14
1229124111	0x4942f20f	NULL_PARAM_15
1229124112	0x4942f210	NULL_PARAM_16

Decimal	Hexadecimal	Minor code reason
1229124113	0x4942f211	NULL_PARAM_17
1229124114	0x4942f212	NULL_PARAM_18
1229124115	0x4942f213	NULL_PARAM_19
1229124116	0x4942f214	NULL_PARAM_20
1229124117	0x4942f215	NULL_IOR_OBJECT
1229124118	0x4942f216	NULL_SC_DATA
1229124126	0x4942f21e	BAD_SERVANT_TYPE
1229124127	0x4942f21f	BAD_EXCEPTION
1229124128	0x4942f220	BAD_MODIFIER_LIST
1229124129	0x4942f221	NULL_PROP_MGR
1229124130	0x4942f222	INVALID_PROPERTY
1229124131	0x4942f223	ORBINITREF_FORMAT
1229124132	0x4942f224	ORBINITREF_MISSING_OBJECTURL
1229124133	0x4942f225	ORBDEFAULTINITREF_FORMAT
1229124134	0x4942f226	ORBDEFAULTINITREF_VALUE
1229124135	0x4942f227	OBJECTKEY_SERVERUUID_LENGTH
1229124136	0x4942f228	OBJECTKEY_SERVERUUID_NULL
1229124139	0x4942f22b	NULL_OBJECT_IOR
1229124225	0x4942f281	TYPECODEIMPL_CTOR_MISUSE_1
1229124226	0x4942f282	TYPECODEIMPL_CTOR_MISUSE_2
1229124227	0x4942f283	TYPECODEIMPL_NULL_INDIRECTTYPE
1229124228	0x4942f284	TYPECODEIMPL_RECURSIVE_ TYPECODES

Decimal minor exception codes 1229124235 through 1229125765

Decimal	Hexadecimal	Minor code reason
1229124235	0x4942f28b	TYPECODEIMPL_KIND_INVALID_1
1229124236	0x4942f28c	TYPECODEIMPL_KIND_INVALID_2
1229124237	0x4942f28d	TYPECODEIMPL_NATIVE_1
1229124238	0x4942f28e	TYPECODEIMPL_NATIVE_2
1229124239	0x4942f28f	TYPECODEIMPL_NATIVE_3
1229124240	0x4942f290	TYPECODEIMPL_KIND_INDIRECT_1
1229124241	0x4942f291	TYPECODEIMPL_KIND_INDIRECT_2
1229124242	0x4942f292	TYPECODEIMPL_NULL_TYPECODE
1229124243	0x4942f293	TYPECODEIMPL_BODY_OF_TYPECODE
1229124244	0x4942f294	TYPECODEIMPL_KIND_RECURSIVE_2
1229124245	0x4942f295	TYPECODEIMPL_COMPLEX_DEFAULT_1
1229124246	0x4942f296	TYPECODEIMPL_COMPLEX_DEFAULT_3
1229124247	0x4942f297	TYPECODEIMPL_INDIRECTION
1229124248	0x4942f298	TYPECODEIMPL_SIMPLE_DEFAULT
1229124249	0x4942f299	TYPECODEIMPL_NOT_CDROS
1229124353	0x4942f301	CONNECT_FAILURE_1
1229124354	0x4942f302	CONNECT_FAILURE_2
1229124355	0x4942f303	CONNECT_FAILURE_3
1229124356	0x4942f304	CONNECT_FAILURE_4
1229124357	0x4942f305	CONN_PURGE_REBIND
1229124358	0x4942f306	CONN_PURGE_ABORT
1229124359	0x4942f307	CONN_NOT_ESTABLISH
1229124360	0x4942f308	CONN_CLOSE_REBIND
1229124368	0x4942f310	WRITE_ERROR_SEND

Decimal	Hexadecimal	Minor code reason
1229124376	0x4942f318	GET_PROPERTIES_ERROR
1229124384	0x4942f320	BOOTSTRAP_SERVER_NOT_AVAIL
1229124392	0x4942f328	INVOKE_ERROR
1229124481	0x4942f381	BAD_HEX_DIGIT
1229124482	0x4942f382	BAD_STRINGIFIED_IOR_LEN
1229124483	0x4942f383	BAD_STRINGIFIED_IOR
1229124485	0x4942f385	BAD_MODIFIER_1
1229124486	0x4942f386	BAD_MODIFIER_2
1229124488	0x4942f388	CODESET_INCOMPATIBLE
1229124490	0x4942f38a	LONG_DOUBLE_NOT_IMPLEMENTED_1
1229124491	0x4942f38b	LONG_DOUBLE_NOT_IMPLEMENTED_2
1229124492	0x4942f38c	LONG_DOUBLE_NOT_IMPLEMENTED_3
1229124496	0x4942f390	COMPLEX_TYPES_NOT_IMPLEMENTED
1229124497	0x4942f391	VALUE_BOX_NOT_IMPLEMENTED
1229124865	0x4942f501	TRANS_NS_CANNOT_CREATE_INITIAL_NC_SYS
1229124866	0x4942f502	TRANS_NS_CANNOT_CREATE_INITIAL_NC
1229124867	0x4942f503	GLOBAL_ORB_EXISTS
1229124868	0x4942f504	PLUGINS_ERROR
1229124993	0x4942f581	BAD_REPLYSTATUS
1229124994	0x4942f582	PEEKSTRING_FAILED
1229124995	0x4942f583	GET_LOCAL_HOST_FAILED
1229124996	0x4942f584	CREATE_LISTENER_FAILED
1229124997	0x4942f585	BAD_LOCATE_REQUEST_STATUS
1229124998	0x4942f586	STRINGIFY_WRITE_ERROR
1229125000	0x4942f588	BAD_GIOP_REQUEST_TYPE_1
1229125001	0x4942f589	BAD_GIOP_REQUEST_TYPE_2
1229125002	0x4942f58a	BAD_GIOP_REQUEST_TYPE_3
1229125003	0x4942f58b	BAD_GIOP_REQUEST_TYPE_4
1229125005	0x4942f58d	NULL_ORB_REFERENCE
1229125006	0x4942f58e	NULL_NAME_REFERENCE
1229125008	0x4942f590	ERROR_UNMARSHALING_USEREXC
1229125009	0x4942f591	SUBCONTRACTREGISTRY_ERROR
1229125010	0x4942f592	LOCATIONFORWARD_ERROR
1229125011	0x4942f593	BAD_READER_THREAD
1229125013	0x4942f595	BAD_REQUEST_ID
1229125014	0x4942f596	BAD_SYSTEMEXCEPTION
1229125015	0x4942f597	BAD_COMPLETION_STATUS
1229125016	0x4942f598	INITIAL_REF_ERROR
1229125017	0x4942f599	NO_CODEC_FACTORY
1229125018	0x4942f59a	BAD_SUBCONTRACT_ID
1229125019	0x4942f59b	BAD_SYSTEMEXCEPTION_2
1229125020	0x4942f59c	NOT_PRIMITIVE_TYPECODE
1229125021	0x4942f59d	BAD_SUBCONTRACT_ID_2
1229125022	0x4942f59e	NAMING_CTX_REBIND_ALREADY_BOUND
1229125023	0x4942f59f	NAMING_CTX_REBINDCTX_ALREADY_BOUND
1229125024	0x4942f5a0	NAMING_CTX_BAD_BINDINGTYPE

Decimal	Hexadecimal	Minor code reason
1229125025	0x4942f5a1	NAMING_CTX_RESOLVE_CANNOT_ NARROW_TO_CTX
1229125032	0x4942f5a8	TRANS_NC_BIND_ALREADY_BOUND
1229125033	0x4942f5a9	TRANS_NC_LIST_GOT_EXC
1229125034	0x4942f5aa	TRANS_NC_NEWCTX_GOT_EXC
1229125035	0x4942f5ab	TRANS_NC_DESTROY_GOT_EXC
1229125042	0x4942f5b2	INVALID_CHAR_CODESET_1
1229125043	0x4942f5b3	INVALID_CHAR_CODESET_2
1229125044	0x4942f5b4	INVALID_WCHAR_CODESET_1
1229125045	0x4942f5b5	INVALID_WCHAR_CODESET_2
1229125046	0x4942f5b6	GET_HOST_ADDR_FAILED
1229125047	0x4942f5b7 x	REACHED_UNREACHABLE_PATH
1229125048	0x4942f5b8	PROFILE_CLONE_FAILED
1229125049	0x4942f5b9	INVALID_LOCATE_REQUEST_STATUS
1229125512	0x4942f788	BAD_CODE_SET
1229125520	0x4942f790	INV_RMI_STUB
1229125521	0x4942f791	INV_LOAD_STUB
1229125522	0x4942f792	INV_OBJ_IMPLEMENTATION
1229125523	0x4942f793	OBJECTKEY_NOMAGIC
1229125524	0x4942f794	OBJECTKEY_NOSCID
1229125525	0x4942f795	OBJECTKEY_NOSERVERID
1229125526	0x4942f796	OBJECTKEY_NOSERVERUUID
1229125527	0x4942f797	OBJECTKEY_SERVERUUIDKEY
1229125762	0x4942f882	UNSPECIFIED_MARSHAL_1
1229125763	0x4942f883	UNSPECIFIED_MARSHAL_2
1229125764	0x4942f884	UNSPECIFIED_MARSHAL_3
1229125765	0x4942f885	UNSPECIFIED_MARSHAL_4

Decimal minor exception codes 1229125766 through 1229125924

Decimal	Hexadecimal	Minor code reason
1229125766	0x4942f886	UNSPECIFIED_MARSHAL_5
1229125767	0x4942f887	UNSPECIFIED_MARSHAL_6
1229125768	0x4942f888	UNSPECIFIED_MARSHAL_7
1229125769	0x4942f889	UNSPECIFIED_MARSHAL_8
1229125770	0x4942f88a	UNSPECIFIED_MARSHAL_9
1229125771	0x4942f88b	UNSPECIFIED_MARSHAL_10
1229125772	0x4942f88c	UNSPECIFIED_MARSHAL_11
1229125773	0x4942f88d	UNSPECIFIED_MARSHAL_12
1229125774	0x4942f88e	UNSPECIFIED_MARSHAL_13
1229125775	0x4942f88f	UNSPECIFIED_MARSHAL_14
1229125776	0x4942f890	UNSPECIFIED_MARSHAL_15
1229125777	0x4942f891	UNSPECIFIED_MARSHAL_16
1229125778	0x4942f892	UNSPECIFIED_MARSHAL_17
1229125779	0x4942f893	UNSPECIFIED_MARSHAL_18
1229125780	0x4942f894	UNSPECIFIED_MARSHAL_19
1229125781	0x4942f895	UNSPECIFIED_MARSHAL_20
1229125782	0x4942f896	UNSPECIFIED_MARSHAL_21
1229125783	0x4942f897	UNSPECIFIED_MARSHAL_22
1229125784	0x4942f898	UNSPECIFIED_MARSHAL_23
1229125785	0x4942f899	UNSPECIFIED_MARSHAL_24
1229125786	0x4942f89a	UNSPECIFIED_MARSHAL_25
1229125787	0x4942f89b	UNSPECIFIED_MARSHAL_26
1229125788	0x4942f89c	UNSPECIFIED_MARSHAL_27

Decimal	Hexadecimal	Minor code reason
1229125789	0x4942f89d	UNSPECIFIED_MARSHAL_28
1229125790	0x4942f89e	UNSPECIFIED_MARSHAL_29
1229125791	0x4942f89f	UNSPECIFIED_MARSHAL_30
1229125792	0x4942f8a0	UNSPECIFIED_MARSHAL_31
1229125793	0x4942f8a1	UNSPECIFIED_MARSHAL_32
1229125794	0x4942f8a2	UNSPECIFIED_MARSHAL_33
1229125795	0x4942f8a3	UNSPECIFIED_MARSHAL_34
1229125796	0x4942f8a4	UNSPECIFIED_MARSHAL_35
1229125797	0x4942f8a5	UNSPECIFIED_MARSHAL_36
1229125798	0x4942f8a6	UNSPECIFIED_MARSHAL_37
1229125799	0x4942f8a7	UNSPECIFIED_MARSHAL_38
1229125800	0x4942f8a8	UNSPECIFIED_MARSHAL_39
1229125801	0x4942f8a9	UNSPECIFIED_MARSHAL_40
1229125802	0x4942f8aa	UNSPECIFIED_MARSHAL_41
1229125803	0x4942f8ab	UNSPECIFIED_MARSHAL_42
1229125804	0x4942f8ac	UNSPECIFIED_MARSHAL_43
1229125805	0x4942f8ad	UNSPECIFIED_MARSHAL_44
1229125806	0x4942f8ae	UNSPECIFIED_MARSHAL_45
1229125807	0x4942f8af	UNSPECIFIED_MARSHAL_46
1229125808	0x4942f8b0	UNSPECIFIED_MARSHAL_47
1229125809	0x4942f8b1	UNSPECIFIED_MARSHAL_48
1229125810	0x4942f8b2	UNSPECIFIED_MARSHAL_49
1229125811	0x4942f8b3	UNSPECIFIED_MARSHAL_50
1229125812	0x4942f8b4	UNSPECIFIED_MARSHAL_51
1229125813	0x4942f8b5	UNSPECIFIED_MARSHAL_52
1229125818	0x4942f8ba	UNSPECIFIED_MARSHAL_57
1229125819	0x4942f8bb	UNSPECIFIED_MARSHAL_58
1229125820	0x4942f8bc	UNSPECIFIED_MARSHAL_59
1229125821	0x4942f8bd	UNSPECIFIED_MARSHAL_60
1229125822	0x4942f8be	UNSPECIFIED_MARSHAL_61
1229125823	0x4942f8bf	UNSPECIFIED_MARSHAL_62
1229125824	0x4942f8c0	UNSPECIFIED_MARSHAL_63
1229125825	0x4942f8c1	READ_OBJECT_EXCEPTION_1
1229125826	0x4942f8c2	READ_OBJECT_EXCEPTION_2
1229125828	0x4942f8c4	UNSUPPORTED_IDLTYPE
1229125842	0x4942f8d2	DSI_RESULT_EXCEPTION
1229125844	0x4942f8d4	IIOPINPUTSTREAM_GROW
1229125847	0x4942f8d7	NO_CHAR_CONVERTER_1
1229125848	0x4942f8d8	NO_CHAR_CONVERTER_2
1229125849	0x4942f8d9	CHARACTER_MALFORMED_1
1229125850	0x4942f8da	CHARACTER_MALFORMED_2
1229125851	0x4942f8db	CHARACTER_MALFORMED_3
1229125852	0x4942f8dc	CHARACTER_MALFORMED_4
1229125854	0x4942f8de	INCORRECT_CHUNK_LENGTH
1229125856	0x4942f8e0	CHUNK_OVERFLOW
1229125858	0x4942f8e2	CANNOT_GROW
1229125859	0x4942f8e3	CODESET_ALREADY_SET
1229125860	0x4942f8e4	REQUEST_CANCELLED
1229125861	0x4942f8e5	WRITE_TO_STREAM_1
1229125862	0x4942f8e6	WRITE_TO_STREAM_2
1229125863	0x4942f8e7	WRITE_TO_STREAM_3
1229125864	0x4942f8e8	WRITE_TO_STREAM_4
1229125889	0x4942f901	DSI_NOT_IMPLEMENTED

Decimal	Hexadecimal	Minor code reason
1229125890	0x4942f902	GETINTERFACE_NOT_IMPLEMENTED
1229125891	0x4942f903	SEND_DEFERRED_NOTIMPLEMENTED
1229125893	0x4942f905	ARGUMENTS_NOTIMPLEMENTED
1229125894	0x4942f906	RESULT_NOTIMPLEMENTED
1229125895	0x4942f907	EXCEPTIONS_NOTIMPLEMENTED
1229125896	0x4942f908	CONTEXTLIST_NOTIMPLEMENTED
1229125902	0x4942f90e	CREATE_OBJ_REF_BYTE_NOTIMPLEMENTED
1229125903	0x4942f90f	CREATE_OBJ_REF_IOR_ NOTIMPLEMENTED
1229125904	0x4942f910	GET_KEY_NOTIMPLEMENTED
1229125905	0x4942f911	GET_IMPL_ID_NOTIMPLEMENTED
1229125906	0x4942f912	GET_SERVANT_NOTIMPLEMENTED
1229125907	0x4942f913	SET_ORB_NOTIMPLEMENTED
1229125908	0x4942f914	SET_ID_NOTIMPLEMENTED
1229125909	0x4942f915	GET_CLIENT_SUBCONTRACT_ NOTIMPLEMENTED
1229125913	0x4942f919	CONTEXTIMPL_NOTIMPLEMENTED
1229125914	0x4942f91a	CONTEXT_NAME_NOTIMPLEMENTED
1229125915	0x4942f91b	PARENT_NOTIMPLEMENTED
1229125916	0x4942f91c	CREATE_CHILD_NOTIMPLEMENTED
1229125917	0x4942f91d	SET_ONE_VALUE_NOTIMPLEMENTED
1229125918	0x4942f91e	SET_VALUES_NOTIMPLEMENTED
1229125919	0x4942f91f	DELETE_VALUES_NOTIMPLEMENTED
1229125920	0x4942f920	GET_VALUES_NOTIMPLEMENTED
1229125922	0x4942f922	GET_CURRENT_NOTIMPLEMENTED_1
1229125923	0x4942f923	GET_CURRENT_NOTIMPLEMENTED_2
1229125924	0x4942f924	CREATE_OPERATION_LIST_NOTIMPLEMENTED_1

Decimal minor exception codes 1229125925 through 1330446365

Decimal	Hexadecimal	Minor code reason
1229125925	0x4942f925	CREATE_OPERATION_LIST_NOTIMPLEMENTED_2
1229125926	0x4942f926	GET_DEFAULT_CONTEXT_NOTIMPLEMENTED_1
1229125927	0x4942f927	GET_DEFAULT_CONTEXT_NOTIMPLEMENTED_2
1229125928	0x4942f928	SHUTDOWN_NOTIMPLEMENTED
1229125929	0x4942f929	WORK_PENDING_NOTIMPLEMENTED
1229125930	0x4942f92a	PERFORM_WORK_NOTIMPLEMENTED
1229125931	0x4942f92b	COPY_TK_ABSTRACT_NOTIMPLEMENTED
1229125932	0x4942f92c	PI_CLIENT_GET_POLICY_NOTIMPLEMENTED

Decimal	Hexadecimal	Minor code reason
1229125933	0x4942f92d	PI_SERVER_GET_POLICY_NOTIMPLEMENTED
1229125934	0x4942f92e	ADDRESSING_MODE_NOTIMPLEMENTED_1
1229125935	0x4942f92f	ADDRESSING_MODE_NOTIMPLEMENTED_2
1229125936	0x4942f930	SET_OBJECT_RESOLVER_NOTIMPLEMENTED
1229126017	0x4942f981	MARSHAL_NO_MEMORY_1
1229126018	0x4942f982	MARSHAL_NO_MEMORY_2
1229126019	0x4942f983	MARSHAL_NO_MEMORY_3
1229126020	0x4942f984	MARSHAL_NO_MEMORY_4
1229126021	0x4942f985	MARSHAL_NO_MEMORY_5
1229126022	0x4942f986	MARSHAL_NO_MEMORY_6
1229126023	0x4942f987	MARSHAL_NO_MEMORY_7
1229126024	0x4942f988	MARSHAL_NO_MEMORY_8
1229126025	0x4942f989	MARSHAL_NO_MEMORY_9
1229126026	0x4942f98a	MARSHAL_NO_MEMORY_10
1229126027	0x4942f98b	MARSHAL_NO_MEMORY_11
1229126028	0x4942f98c	MARSHAL_NO_MEMORY_12
1229126029	0x4942f98d	MARSHAL_NO_MEMORY_13
1229126030	0x4942f98e	MARSHAL_NO_MEMORY_14
1229126031	0x4942f98f	MARSHAL_NO_MEMORY_15
1229126032	0x4942f990	MARSHAL_NO_MEMORY_16
1229126033	0x4942f991	MARSHAL_NO_MEMORY_17
1229126034	0x4942f992	MARSHAL_NO_MEMORY_18
1229126035	0x4942f993	MARSHAL_NO_MEMORY_19
1229126036	0x4942f994	MARSHAL_NO_MEMORY_20
1229126037	0x4942f995	MARSHAL_NO_MEMORY_21
1229126038	0x4942f996	MARSHAL_NO_MEMORY_22
1229126039	0x4942f997	MARSHAL_NO_MEMORY_23
1229126040	0x4942f998	MARSHAL_NO_MEMORY_24
1229126041	0x4942f999	MARSHAL_NO_MEMORY_25
1229126042	0x4942f99a	MARSHAL_NO_MEMORY_26
1229126043	0x4942f99b	MARSHAL_NO_MEMORY_27
1229126044	0x4942f99c	MARSHAL_NO_MEMORY_28
1229126045	0x4942f99d	MARSHAL_NO_MEMORY_29
1229126046	0x4942f99e	MARSHAL_NO_MEMORY_30
1229126047	0x4942f99f	MARSHAL_NO_MEMORY_31
1229126401	0x4942fb01	RESPONSE_TIMED_OUT
1229126402	0x4942fb02	FRAGMENT_TIMED_OUT
1229126529	0x4942fb81	NO_SERVER_SC_IN_DISPATCH
1229126530	0x4942fb82	NO_SERVER_SC_IN_LOOKUP
1229126531	0x4942fb83	NO_SERVER_SC_IN_CREATE_ DEFAULT_SERVER
1229126532	0x4942fb84	NO_SERVER_SC_IN_SETUP
1229126533	0x4942fb85	NO_SERVER_SC_IN_LOCATE
1229126534	0x4942fb86	NO_SERVER_SC_IN_DISCONNECT
1229126539	0x4942fb8b	ORB_CONNECT_ERROR_1
1229126540	0x4942fb8c	ORB_CONNECT_ERROR_2
1229126541	0x4942fb8d	ORB_CONNECT_ERROR_3
1229126542	0x4942fb8e	ORB_CONNECT_ERROR_4
1229126543	0x4942fb8f	ORB_CONNECT_ERROR_5
1229126544	0x4942fb90	ORB_CONNECT_ERROR_6
1229126545	0x4942fb91	ORB_CONNECT_ERROR_7

Decimal	Hexadecimal	Minor code reason
1229126546	0x4942fb92	ORB_CONNECT_ERROR_8
1229126547	0x4942fb93	ORB_CONNECT_ERROR_9
1229126548	0x4942fb94	ORB_REGISTER_1
1229126549	0x4942fb95	ORB_REGISTER_2
1229126553	0x4942fb99	ORB_REGISTER_LOCAL_1
1229126554	0x4942fb9a	ORB_REGISTER_LOCAL_2
1229126657	0x4942fc01	LOCATE_UNKNOWN_OBJECT
1229126658	0x4942fc02	BAD_SERVER_ID_1
1229126659	0x4942fc03	BAD_SERVER_ID_2
1229126660	0x4942fc04	BAD_IMPLID
1229126665	0x4942fc09	BAD_SKELETON_1
1229126666	0x4942fc0a	BAD_SKELETON_2
1229126673	0x4942fc11	SERVANT_NOT_FOUND_1
1229126674	0x4942fc12	SERVANT_NOT_FOUND_2
1229126675	0x4942fc13	SERVANT_NOT_FOUND_3
1229126676	0x4942fc14	SERVANT_NOT_FOUND_4
1229126677	0x4942fc15	SERVANT_NOT_FOUND_5
1229126678	0x4942fc16	SERVANT_NOT_FOUND_6
1229126679	0x4942fc17	SERVANT_NOT_FOUND_7
1229126687	0x4942fc1f	SERVANT_DISCONNECTED_1
1229126688	0x4942fc20	SERVANT_DISCONNECTED_2
1229127297	0x4942fe81	UNKNOWN_CORBA_EXC
1229127298	0x4942fe82	RUNTIMEEXCEPTION
1229127299	0x4942fe83	UNKNOWN_SERVER_ERROR
1229127300	0x4942fe84	UNKNOWN_DSI_SYSEX
1229127301	0x4942fe85	UNNOWN_IN_READ_VALUE
1229127302	0x4942fe86	UNKNOWN_CREATE_EXCEPTION_RESPONSE
1229127312	0x4942fe90	UNKNOWN_PI_EXC
1229127313	0x4942fe91	UNKNOWN_PI_EXC_2
1229127314	0x4942fe92	PI_ARGS_FAILURE
1229127315	0x4942fe93	PI_EXCEPTS_FAILURE
1229127316	0x4942fe94	PI_CONTEXTS_FAILURE
1229127317	0x4942fe95	PI_OP_CONTEXT_FAILURE
1229127326	0x4942fe9e	USER_DEFINED_ERROR
1229127327	0x4942fe9f	UNKNOWN_RUNTIME_IN_BOOTSTRAP
1229127328	0x4942fea0	UNKNOWN_THROWABLE_IN_BOOTSTRAP
1229127329	0x4942fea1	UNKNOWN_RUNTIME_IN_INSAGENT
1229127330	0x4942fea2	UNKNOWN_THROWABLE_IN_INSAGENT
1330446337	0x4f4d0001	CHARACTER_NOT_MAPPED
1330446338	0x4f4d0002	INCOMPATIBLE_VALUE_IMPL
1330446339	0x4f4d0003	NAME_ALREADY_USED_IN_THE_CONTEXT_IN_IFR
1330446340	0x4f4d0004	CANNOT_MARSHAL_LOCAL_OBJECT
1330446341	0x4f4d0005	NAME_CLASH_IN_INHERITED_CONTEXT
1330446342	0x4f4d0006	INCORRECT_TYPE_FOR_ABSTRACT_ INTERFACE
1330446343	0x4f4d0007	INS_BAD_SCHEME_NAME
1330446344	0x4f4d0008	INS_BAD_ADDRESS
1330446345	0x4f4d0009	INS_BAD_SCHEME_SPECIFIC_PART
1330446346	0x4f4d000a	INS_OTHER
1330446350	0x4f4d000e	INVALID_PI_CALL

Decimal	Hexadecimal	Minor code reason
1330446351	0x4f4d000f	SERVICE_CONTEXT_ID_EXISTS
1330446359	0x4f4d0017	NO_TRANSMISSION_CODE
1330446362	0x4f4d001a	INVALID_SERVICE_CONTEXT
1330446363	0x4f4d001b	NULL_OBJECT_ON_REGISTER
1330446364	0x4f4d001c	INVALID_COMPONENT_ID
1330446365	0x4f4d001d	INVALID_IOR_PROFILE

If none of these steps fixes your problem, check to see if the problem has been identified and documented by looking at the available online support (hints and tips, technotes, and fixes). If you do not find your problem listed there, contact IBM support.

Sybase troubleshooting tips

This article describes how to diagnose the following problems related to Sybase:

- Executing the DatabaseMetaData.getBestRowIdentifier() method in an XA transaction causes errors
- Sybase requirements for using the escapes and DatabaseMetaData methods
- Database deadlocks and XA_PROTO errors occur when using Sybase
- Executing a stored procedure containing a SELECT INTO command causes exception
- Error is incorrectly reported about IMAGE to VARBINARY conversion
- Java Database Connectivity (JDBC) 1.0 standard methods are not implemented and generate a SQL exception when used
- Sybase transaction manager fails after trying to alleviate deadlock error
- Starting an XA transaction when the autoCommit value of the connection is false causes error
- Sybase does not throw an exception when an incorrect database name is specified

Executing the DatabaseMetaData.getBestRowIdentifier() method in an XA transaction causes errors

Executing the DatabaseMetaData.getBestRowIdentifier() method while in an XA transaction causes the following errors:

```
SQL Exception: The 'CREATE TABLE' command is not allowed within a
multi-statement transaction in the 'tempdb' database.  Calling
DatabaseMetaData.getBestRowIdentifier()
```

Currently, this method fails when using Sybase. This problem occurs with other methods as well, including:

- getBestRowIdentifier();
- getVersionColumns();
- getTablePrivileges();
- getProcedureColumns();
- getPrimaryKeys();
- getIndexInfo();
- getImportedKeys();
- getExportedKeys();

- getCrossReference();
- getColumns();
- getColumnPrivileges();

Case 10880427 has been opened with Sybase to resolve this problem.

Sybase requirements for using the escapes and DatabaseMetaData methods

To use the escapes and DatabaseMetaData methods, you must install stored procedures on the Adaptive Server Enterprise or Adaptive Server Anywhere database where you want to use these methods. These stored procedures are also required by some of the connection methods.

To check for the presence of LOCATE ():

1. Open a Sybase **isql** command prompt.
2. Type the command **use master**.
3. Type the command **go**.
4. Type the SQL command and select * from jdbc_function_escapes.
5. Type the command **go**.

The following appears:

```
escape_name          map_string
 ----------------------------
 abs          abs(%1)
 acos         acos(%1)
 asin         asin(%1)
 atan         atan(%1)
 atan2        atn2(%1, %2)
 ceiling      ceiling(%1)
::::::::::::::::::::::

locate        charindex ((convert (varchar, %1)), (convert (varchar, %2)))
```

If the function does not exist, upgrade jConnect to at least Version 5.2 EBF 10635 and run the following command:

```
java IsqlApp -U sa -P -S jdbc:sybase:Tds:<hostname>:4100
          -I %JDBC_HOME%\sp\sql_server12.sql -c go
```

Database deadlocks and XA_PROTO errors occur when using Sybase

When using Sybase with the IBM WebSphere Application Server, do one of the following to prevent database deadlocks and errors:

- Change the transaction isolation level on the connection to TRANSACTION_READ_COMMITTED. Set the isolation level on the connection for unshareable connections or, for shareable connections, define the isolation levels in the resource reference for your data source using the Application Assembly Tool (AAT).
- Modify Sybase by doing one of the following:
  - If you want to use the existing tables, modify the table locking scheme using the **alter table <*table name*> lock datarows** to get a row lock level granularity.
  - If you want to set the system-wide locking scheme to datarows, all subsequently created tables inherit that value and have a locking scheme of datarows.

Note: You must drop your original databases and tables.

Executing a stored procedure containing a SELECT INTO command causes exception

An attempt to execute a stored procedure containing a **SELECT INTO** command results in the following exception:

```
SVR-ERROR: SQL Exception SELECT INTO command not allowed within multi-statement transaction
```

Case 10868947 has been opened with Sybase to resolve this problem.

Error is incorrectly reported about IMAGE to VARBINARY conversion

The following error is incorrectly reported:

```
com.sybase.jdbc2.jdbc.SybSQLException: Implicit conversion from
data type 'IMAGE' to 'VARBINARY' is not allowed.
 Use the CONVERT function to run this query.
```

The error is about a VARBINARY column only and causes confusion if you also have an IMAGE column.

Do one of the following to work around this problem:

- Use a PreparedStatement.setBytes() method instead of a PreparedStatement.setBinaryStream() method
- Use a LONG VARBINARY for the column type if you want to continue using the setBinaryStream() method. You might want to make this change because the size limit for VARBINARY is 255 bytes.

For example:

```
// ***************CORRECTION*****************************
// setBinaryStream fails for column type of VARBINARY , use setBytes() instead
//stmt4.setBinaryStream(8,new java.io.ByteArrayInputStream(tempbyteArray),
// tempbyteArray.length);
         stmt4.setBytes(8,tempbyteArray);
```

Java Database Connectivity 1.0 standard methods are not implemented and generate a SQL exception when used

The following Java Database Connectivity (JDBC) 1.0 standard methods are not implemented and generate a SQL exception when used:

- ResultSetMetaData.getSchemaName()
- ResultSetMetaData.getTableName() (implemented only for text and image datatypes)
- ResultSetMetaData.getCatalogName()

Sybase transaction manager fails after trying to alleviate a deadlock error

If an application encounters a deadlock, Sybase detects the deadlock and throws an exception. Because of this detection, the transaction manager calls an `xa_end` with a `TMFAIL` in it.

The call succeeds, but causes another Sybase exception, `XAERR_PROTO`. This exception only appears in the error log and does not cause any functional problems. All applications should continue to run, therefore no workaround is necessary.

Case 10869169 has been opened with Sybase to resolve this problem.

Starting an XA transaction when the autoCommit value of the connection is false causes error

The exception thrown is `javax.transaction.xa.XAException` with stack trace similar to the following:

```
at com.sybase.jdbc2.jdbc.SybXAResource.sendRPC(SybXAResource.java:711)
at com.sybase.jdbc2.jdbc.SybXAResource.sendRPC(SybXAResource.java:602)
at com.sybase.jdbc2.jdbc.SybXAResource.start(SybXAResource.java:312)
```

This problem affects you when you do both local and global transactions. If, in a local transaction, the autoCommit default value is set to *false*, and a global or XA transaction starts (either a user transaction started by you, or a container transaction started by a container), the exception occurs.

This problem is a Sybase bug as the start() method can fail unexpectedly, regardless of the value of autoCommit. Currently, there is no workaround for this problem, therefore it is not recommended that you mix local and global transactions. Case 10880792 has been opened to resolve this problem.

Sybase does not throw an exception when an incorrect database name is specified

Verify that your database name is correctly entered on the data source properties.

Most databases (DB2, Oracle, Informix , MS SQL Server and Cloudscape) throw an exception when the database specified does not exist. But Sybase does not throw an exception when an incorrect database name is specified. Sybase generates an SQL warning and then connects to the default database. If you misspell the requested database name, Sybase connects you to the master or the default database where the table you requested is not found.

If none of these steps fixes your problem, check to see if the problem has been identified and documented by looking at the available online support (hints and tips, technotes, and fixes). If you do not find your problem listed there, contact IBM support.

DB2 troubleshooting tips

Illegal conversion occurs on any VARCHAR FOR BIT DATA column in a container-managed persistent bean

When enterprise beans with container-managed persistent (CMP) types that have any VARCHAR FOR BIT DATA columns defined on a DB2 table are deployed in the DB2 universal JDBC type 4 driver to persist the data, an SQLException of illegal conversion is thrown at run time. This exception only occurs when you use the DB2 universal JDBC type 4 driver and with the deferPrepares property being set to true. When the deferPrepares property is set to true, the DB2 universal JDBC type 4 driver uses the standard JDBC data mapping.

Currently, the generated deployed code does not follow the standard JDBC specification mapping. The failure at execution time is because of a problem in the tool that prepared the enterprise beans for execution.

To avoid receiving this exception, choose one of the following options:

- Set the deferPrepares property to false in the data source configuration.
- Do not use the DB2 universal JDBC type 4 driver if your table has any VARCHAR FOR BIT DATA or LONG VARCHAR FOR BIT DATA columns. Use the DB2 legacy CLI-based JDBC driver to persist the data. Refer to DB2 V8.1 readme for more details.

Chapter 5. Message reference

You can log WebSphere Application Server system messages from a variety of sources, including application server components and applications. Messages logged by application server components and associated IBM products start with a unique message identifier that indicates the component or application that issued the message. The message identifier can be either 8 or 9 characters in length and has the form:

```
CCCC1234X
```

where:

CCCC is a four character alphabetic component or application identifier.

1234 is a four character numeric identifier used to identify the specific message for that component.

X is an optional alphabetic severity indicator. (I=Informational, W=Warning, E=Error)

To view the messages generated by WebSphere Application Server components, select the **Quick Reference** view of the WebSphere Application Server Version 5 InfoCenter and expand the topic **Messages**.

Chapter 6. CORBA minor codes

Overview

Common Object Request Broker Architecture (CORBA) is an industry-wide standard for object-oriented communication between processes, which is supported in several programming languages. Several subcomponents of WebSphere Application Server use CORBA to communicate across processes.

When a CORBA process fails, that is a request from one process to another cannot be sent, completed, or returned, a high-level exception is thrown, such as TransactionRolledBackException: CORBA TRANSACTION_ROLLEDBACK. In order to show the underlying cause of the failure, applications which use CORBA services generate minor codes, which appear in the exception stack. Look for ″minor code″ in the exception stack to locate these exceptions.

Minor codes used by WebSphere Application Server components

Range	Related subcomponent	Where to find details
0x49424300-0x494243FF	Security	(″Security components troubleshooting tips″)
0x49421050-0x4942105F, 0x49421070-0x4942107F	ORB services	(″Object request broker component troubleshooting tips″)
0x4f4d and above	Standard CORBA exceptions	http://www.omg.org
0x49421080-0x4942108F	Naming services	Javadoc for class ws.code.naming.src.com.ibm.websphere.naming.WsnCorbaMinorCodes
0x49421080-0x4942108F	Workload Management	Javadoc for class com.ibm.websphere.wlm.WsCorbaMinorCodes

Chapter 7. Working with message logs

WebSphere Application Server can write system messages to several general purpose logs. These include the JVM logs, the process logs and the IBM service log.

The JVM logs are created by redirecting the `System.out` and `System.err` streams of the JVM to independent log files. WebSphere Application Server writes formatted messages to the `System.out` stream. In addition, applications and other code can write to these streams using the `print()` and `println()` methods defined by the streams. Some JDK built-ins such as the `printStackTrace()` method on the `Throwable` class can also write to these streams. Typically, the `System.out` log is used to monitor the health of the running application server. The `System.out` log can be used for problem determination, but it is recommended to use the IBM Service log and the advanced capabilities of the Log Analyzer instead. The `System.err` log contains exception stack trace information that is useful when performing problem analysis.

Since each application server represents a JVM, there is one set of JVM logs for each application server and all of its applications, located by default in the `installation_root/logs/server_name` directory. In the case of a WebSphere Application Server Network Deployment configuration, JVM logs are also created for the deployment manager and each node manager, since they also represent JVMs.

The process logs are created by redirecting the `stdout` and `stderr` streams of the process to independent log files. Native code, including the Java Virtual Machine (JVM) itself writes to these files. As a general rule, WebSphere Application Server does not write to these files. However, these logs can contain information relating to problems in native code or diagnostic information written by the JVM.

As with JVM logs, there is a set of process logs for each application server, since each JVM is an operating system process, and in the case of a WebSphere Application Server Network Deployment configuration, a set of process logs for the deployment manager and each node manager.

The IBM service log contains both the WebSphere Application Server messages that are written to the `System.out` stream and some special messages that contain extended service information that is normally not of interest, but can be important when analyzing problems. There is one service log for all WebSphere Application Server JVMs on a node, including all application servers. The IBM Service log is maintained in a binary format and requires a special tool to view. This viewer, the Log Analyzer, provides additional diagnostic capabilities. In addition, the binary format provides capabilities that are utilized by IBM support organizations.

In addition to these general purpose logs, WebSphere Application Server contains other specialized logs that are very specific in nature and are scoped to a particular component or activity. For example, the HTTP server plugin maintains a special log. Normally, these logs are not of interest, but you might be instructed to examine one or more of these logs while performing specific problem determination procedures.

Viewing the JVM logs

The JVM logs are written as plain text files. Therefore there are no special requirements to view these logs. They are located in the `installation_directory/logs/applicationServerName` directory, and by default are named `SystemOut.log` and `SystemErr.log`.

There are two techniques that you can use to view the JVM logs for an application server:

- Use the adminstrative console. This supports viewing the JVM logs from a remote machine.
- Use a text editor on the machine where the logs are stored.

Steps for this task

1. View the JVM logs from the administrative console.
   a. Start the administrative console.
   b. Click **Troubleshooting > Logging and Tracing** in the console navigation tree, then click *server* **> JVM Logs**.
   c. Select the runtime tab.
   d. Click **View** corresponding to the log you want to view.
2. View the JVM logs from the machine where they are stored.
   a. Go to the machine where the logs are stored.
   b. Open the file in a text editor or drag and drop the file into an editing and viewing program.

Interpreting the JVM logs

The JVM logs contain print data written by applications. The application can write this data written directly in the form of `System.out.print()`, `System.err.print()`, or other method calls. The application can also write data indirectly by calling a JVM function, such as an `Exception.printStackTrace()`. In addition, the `System.out` JVM log contains system messages written by the WebSphere Application Server.

You can format application data to look like WebSphere Application Server system messages by using the Installed Application Output field of the JVM Logs properties panel, or as plain text with no additional formatting . WebSphere Application Server system messages are always formatted.

Depending on how the JVM log is configured, formatted messages can be written to the JVM logs in either basic or advanced format.

Message formats

Formatted messages are written to the JVM logs in one of two formats:

Basic Format
The format used in earlier versions of WebSphere Application Server.

Advanced Format
Extends the basic format by adding information about an event, when possible.

Basic and advanced format fields

Basic and Advanced Formats use many of the same fields and formatting techniques. The various fields that may be found in these formats follow:

TimeStamp
The timestamp is formatted using the locale of the process where it is formatted. It includes a fully qualified date (for example YYMMDD) , 24 hour time with millisecond precision and a time zone.

ThreadId
An 8 character hexidecimal value generated from the hash code of the thread that issued the message.

ShortName
The abbreviated name of the logging component that issued the message or trace event. This is typically the class name for WebSphere Application Server internal components, but can be some other identifier for user applications.

LongName
The full name of the logging component that issued the message or trace event. This is typically the fully qualified class name for WebSphere Application Server internal components, but can be some other identifier for user applications.

EventType
A one character field that indicates the type of the message or trace event. Message types are in upper case. Possible values include:

A An Audit message.

I An Informational message.

W A Warning message.

E An Error message.

F A Fatal message.

O A message that was written directly to `System.out` by the user application or internal components.

R A message that was written directly to `System.err` by the user application or internal components.

u A special message type used by the message logging component of the WebSphere Application Server run time.

Z A placeholder to indicate the type was not recognized.

ClassName
The class that issued the message or trace event.

MethodName
The method that issued the message or trace event.

Organization
The organization that owns the application that issued the message or trace event.

Product
The product that issued the message or trace event.

Component
The component within the product that issued the message or trace event.

Basic format

Message events displayed in basic format use the following format. The notation <name> indicates mandatory fields that will always appear in the basic format message. The notation [name] indicates optional or conditional fields that will be included if they can be determined.

```
<timestamp><threadId><shortName><eventType>[className][methodName]<message>
```

Advanced format

Message events displayed in advanced format use the following format. The notation <name> is used to indicate mandatory fields that will always appear in the advanced format for message entries. The notation [name] is used to indicate optional or conditional fields that will be included if they can be determined.

```
<timestamp><threadId><eventType><UOW><source=longName>
[className][methodName]<Organization><Product><Component>
<message>
```

Configuring the JVM logs

Before you begin

Use the administrative console to configure the JVM logs for an application server. Configuration changes for the JVM logs that are made to a running application server are not applied until the next restart of the application server.

Steps for this task

1. Start the administrative console
2. Click **Troubleshooting > Logging and Tracing**, then click *server* **> JVM Logs**.
3. Select the Configuration tab.
4. Scroll through the panel to display the attributes for the stream to configure.
5. Change the appropriate configuration attributes and click **Apply**.
6. Save your configuration changes.

JVM log settings

Use this page to view and modify the settings for the Java Virtual Machine (JVM) `System.out` and `System.err` logs.

To view this administrative console page, click **Troubleshooting > Logs and Trace > *server* > JVM Logs**.

File Name

Specifies the name of one of the log file described on this page.

The first file name field specifies the name of the `System.out` log. The second file name field specifies the name of the `System.err` file.

Press the **View** button on the Runtime tab to view the contents of a selected log file.

The file name specified for the `System.out` log or the `System.err` log must have one of the following values:

filename
The name of a file in the file system. It is recommended that you use a

fully qualified file name. If the file name is not fully qualified, it is considered to be relative to the current working directory for the server. Each stream must be configured with a dedicated file. For example, you cannot redirect both `System.out` and `System.err` to the same physical file.

If the directory containing the file already exists, the user ID under which the server is running requires read/write access to the directory. If the directory does not exist, it will be created with the proper permissions. The user id under which the server is running must have authority to create the directory.

console
This is a special file name used to redirect the stream to the corresponding process stream. If this value is specified for `System.out`, the file is redirected to `stdout`. If this value is specified for `System.err`, the file is redirected to `stderr`.

none Discards all data written to the stream. Specifying **none** is equivalent to redirecting the stream to dev/null on a Unix system.

File formatting
Specifies the format to use in saving the `System.out` file.

Log file rotation
Use this set of configuration attributes to configure the `System.out` or `System.err` log file to be self-managing.

A self-managing log file writes messages to a file until reaching either the time or size criterion. At the specified time or when the file reaches the specified size, logging temporarily suspends while the log file rolls over, which involves closing and renaming the saved file. The new saved file name is based on the original name of the file plus a timestamp qualifier that indicates when the renaming occurs. Once the renaming completes, a new, empty log file with the original name reopens and logging resumes. All messages remain after the log file rollover, although a single message can split across the saved and the current file.

You can only configure a log to be self-managing if the corresponding stream is redirected to a file.

File Size
Click this attribute for the log file to manage itself based on its file size. Automatic roll over occurs when the file reaches the specified size you specify in the maximum size field.

Maximum Size
Specify the maximum size of the file in megabytes. When the file reaches this size, it rolls over.

This attribute is only valid if you click File size.

Time Click this attribute for the log file to manage itself based on the time of day. At the time specified in the start time field, the file rolls over.

Start Time
Specify the hour of the day, from 1 to 24, from which the periodic rollover algorithm commences. The periodic rollover algorithm uses this hour to load the algorithm at application server startup. Once started, the rollover algorithm runs without adjustment until the application server is stopped. This rollover pattern continues without adjustment until the Application Server stops.

Repeat Time
Specify the number of hours, from 1 to 24, when a periodic rollover occurs.

Repeat time
Specifies the number of hours after which the log file rolls over. Valid values range from 1 to 24.

Configure a log file to roll over by time, by size, or by time and size. Click **File Size** and **Time** to roll the file at the first matching criterion. For example, if the repeat time field is 5 hours and the maximum file size is 2 MB, the file rolls every 5 hours, unless it reaches 2 MB before the interval elapses. After the size rollover, the file continues to roll at each interval.

Maximum Number of Historical Log Files
Specifies the number of historical (rolled) files to keep. The stream writes to the current file until it rolls. At rollover, the current file closes and is saved as a new name consisting of the current name plus the rollover timestamp. The stream then reopens a new file with the original name to continue writing. The number of historical files grows from zero to the value of the maximum number of historical files field. The next rollover deletes the oldest historical file.

Installed Application Output
Specifies whether `System.out` or `System.err` print statements issued from application code are logged and formatted.

Show application print statements
Click this field to show messages that applications write to the stream using **print** and **println** stream methods. WebSphere Application Server system messages always appear.

Format print statements
Click this field to format application print statement like WebSphere Application Server system messages.

Process logs

WebSphere Application Server processes contain two output streams that are accessible to native code running in the process. These streams are the stdout and stderr streams. Native code, including the JVM, may write data to these process streams. In addition, the JVM provided System.out and System.err streams can be configured to write their data to these streams also.

By default, the `stdout` and `stderr` streams are redirected to log files at application server startup, which contain text written to the `stdout` and `stderr` streams by native modules (`.dlls`, `.exes`, UNIX libraries, and other modules). By default, these files are stored as `installation_root/logs/applicationServerName/native_stderr.log` and `native_stdout.log`.

This is a change from previous versions of WebSphere Application Server, which by default had one log file for both JVM standard output and native standard output, and one log file for both JVM standard error and native error output.

Viewing the service log

The service log is a special log written in a binary format. You cannot view the log directly using a text editor. You should never directly edit the service log, as doing so will corrupt the log. To move the service log from one machine to another, you must use a mechanism like FTP, which supports binary file transfer.

You can view the service log in two ways:

- It is recommended that you use the Log Analyzer tool to view the service log. This tool provides interactive viewing and analysis capability that is helpful in identifying problems.
- If you are unable to use the Log Analyzer tool, you can use the Showlog tool to convert the contents of the service log to a text format that you can then write to a file or dump to the command shell window. The steps for using the Showlog tool are described below.

Steps for this task

1. Open a shell window on the machine where the service log resides.
2. Change directory to `install_directory/bin` where `install_directory` is the fully qualified path where the WebSphere Application Server product is installed.
3. **(Optional)** Run the showlog script with no parameters to display usage instructions.

   On Windows systems, the script is named `<samp>showlog.bat` . On UNIX systems, the script is named `<samp>showlog.sh` .
4. Run the showlog script with the appropriate parameters.

   For example:

   To display the service log contents to the shell window, use the invocation `showlog service_log_filename`. If the service log is not in the default location, you must fully qualify the `service_log_filename`.

   To format and write the service log contents to a file use the invocation `showlog service_log_filename output_filename`. If the service log is not in the default location, you must fully qualify the `service_log_filename`.

Interpreting the service log

To take advantage of the Log Analyzer's browsing and analysis capabilities, start the Log Analyzer tool:

- On Unix systems: `installation_root/bin/waslogbr`
- On Windows systems: *installation_root*/bin/waslogbr.bat

Use the **File > open** menu item, and select the file `/logs/activity.log`. You can also browse to activity logs from other WebSphere Application Server installations, and even other versions of the product. Expand the tree of admininstrative and application server logging sessions. Uncolored records are ″normal″, yellow records are warnings, and pink records are errors. If you select a record, you will see its contents, including the basic error or warning message, date, time, which component logged the record, and which process (i.e., administrative server or an application server) it came from, in the upper right-hand pane.

The activity log does not analyze any other log files, such as `default_stderr.log` or `tracefile`. To analyze these records, right click on a record in the tree on the left (click on the **UnitOfWorkView** at the top to get them all), and select **Analyze**. Any records with a green check mark next to them match a record in the symptom database. When you select a check-marked record, you will see an explanation of the problem in the lower right-hand pane.

Updating the symptom database

The database of known problems and resolutions used when you click the Analyze menu item is periodically enhanced as new problems come to light and new versions of the product are introduced. To ensure that you have the latest version of the database, use the **File > Update Database > WebSphere Application Server Symptom Database** menu item from within the log analyzer tool at least once a month. Users who have just installed the product and have never run the update should do so immediately, since updates may have occurred since the version was first released.

The knowledge base used for analysis is built gradually from problems reported by users. For a recently released version of the product, you might not find any analysis hits. However, the Log Analyzer tool still provides a way to see all error messages and warnings from all components in a convenient, user-friendly way.

Configuring the service log

Before you begin

By default, the service log is shared among all server processes for a node. The configuration values for the service log are inherited by each server process from the node configuration. You can configure a separate service log for each server process by overriding the configuration values at the server level.

Steps for this task

1. Start the administrative console.
2. Click **Troubleshooting > Logs and Trace >** *server_name* **> IBM Service Logs**.
3. Select the **Enable** box to enable the service log, clear the check box to disable the log.
4. Set the name for the service log.

   The default name is `install_directory/logs/activity.log`. If the name is changed, the run time requires write access to the new file, and the file must use the `.log` extension.
5. Set the maximum file size.

   Specifies the number of megabytes to which the file can grow. When the file reaches this size, it wraps, replacing the oldest data with the newest data.
6. Set the message filter level to the desired state.
7. Save the configuration.
8. Restart the server to apply the configuration changes.

IBM service log settings

Use this page to configure the IBM Service log, also known as the *activity log*.

To view this administrative console page, click **Troubleshooting > Logs and Trace** > *server* > **IBM Service Logs**.

Enable service log

Specifies creation of a log file by the IBM Service log.

File Name

Specifies the name of the file used by the IBM Service log.

Maximum File Size

Specifies the maximum size in megabytes of the service log file. The default value is 2 megabytes.

When this size is reached, the service log wraps in place. Note that the service log does not roll over to a new log file like the JVM logs.

Message Filtering
Specifies the message filter level to the desired state. You can use this option to filter out or suppress some message categories. This filter value is applied to all logs, not just the service log.

Enable Correlation ID
Specifies the generation of a correlation ID that is logged with each message.

You can use the correlation ID to correlate activity to a particular client request, or correlate activities on multiple application servers.

Configuration problem settings

Use this page to identify and view problems that exist in the current configuration.

To view this administrative console page, click **Troubleshooting > Configuration Problems** in the console navigation tree.

Click a configuration problem in the Configuration Problems table to see more information about the problem.

Configuration document validation
Use these fields to specify the level of validation to perform on configuration documents.

Cross validation
Enables cross validation of configuration documents.

Enabling cross validation enables comparison of configuration documents for conflicting settings.

Configuration Problems
Displays current configuration problem error messages. Click a message for detailed information about the problem.

Scope
Sorts the configuration problem list by the configuration file where each error occurs. Click a message for detailed information about the problem.

Fields that explain each configuration problem:

The following informational fields appear when you click a configuration problem message.

Message
Displays the message returned from the validator.

Explanation
A brief explanation of the problem.

User action

Specifies the recommended action to correct the problem.

Target Object

Identifies the configuration object where the validation error occurred.

Severity

Indicates the severity of the configuration error, with 1 being a severe error. Severity decreases as the severity descriptor increases.

Local URI

Specifies the local URI of the configuration file where the error occurred.

Full URI

Specifies the full URI of the configuration file where the error occurred.

Validator classname

The classname of the validator reporting the problem.

Chapter 8. Debugging with the Application Server Toolkit

The Application Server Toolkit, included with the WebSphere Application Server on a separately-installable CD, includes debugging functionality that is built on the Eclipse workbench and that includes the following:

The WebSphere Application Server debug adapter
which allows you to debug Web objects that are running on WebSphere Application Server and that you have launched in a browser. These objects include EJBs, JSPs, and servlets.

The JavaScript debug adapter
which enables server-side JavaScript debugging.

The Compiled language debugger
which allows you to detect and diagnose errors in compiled-language applications.

The Java development tools (JDT) debugger
which allows you to debug Java.

All of the debug components in the Application Server Toolkit can be used for debugging locally and for remote debugging.

To learn more about the debug components, launch the Application Server Toolkit, select **Help > Help Contents** and choose the **Debugger Guide bookshelf** entry. To learn about known limitations and problems that are associated with the Application Server Toolkit, see the Application Server Toolkit release notes.

Debugging WebSphere Application Server applications

In order to debug your application, you must use your application development tool (such as WebSphere Studio Application Developer) to create a Java project or a project with a Java nature. You must then import the program that you want to debug into the project. By following the steps below, you can import the WebSphere Application Server examples into a Java project.

There are two debugging styles available:

- **Step-by-step** debugging mode prompts you whenever the server calls a method on a Web object. A dialog lets you step into the method or skip it. In the dialog, you can turn off step-by-step mode when you are finished using it.
- **Breakpoints** debugging mode lets you debug specific parts of programs. Add breakpoints to the part of the code that you must debug and run the program until one of the breakpoints is encountered.

Breakpoints actually work with both styles of debugging. Step-by-step mode just lets you see which Web objects are being called without having to set up breakpoints ahead of time.

You need not import an entire program into your project. However, if you do not import all of your program into the project, some of the source might not compile. You can still debug the project. Most features of the debugger work, including breakpoints, stepping, and viewing and modifying variables. You must import any source that you want to set breakpoints in.

The inspect and display features in the source view do not work if the source has build errors. These features let you select an expression in the source view and evaluate it.

Steps for this task

1. Create a Java Project by opening the New Project dialog.
2. Select **Java** from the left side of the dialog and **Java Project** in the right side of the dialog.
3. Click **Next** and then specify a name for the project (such as WASExamples).
4. Press **Finish** to create the project.
5. Select the new project, choose **File > Import > File System**, then **Next** to open the import file system dialog.
6. Select the directory Browse pushbutton and go to the following directory:

   ```
   installation_root\installedApps\node_name\
                   DefaultApplication.ear\DefaultWebApplication.war
   ```

   .
7. Select the checkbox next to DefaultWebApplication.war in the left side of the Import dialog and then click **Finish**.

   This will import the JSPs and Java source for the examples into your project.
8. Add any JAR files needed to build to the Java Build Path.

   To do this, select **Properties** from the right-click menu. Choose the Java Build Path node and then select the Libraries tab. Use the Add External JARs pushbutton to add the following JAR files:

   - *installation_root*\installedApps*node_name*\ DefaultApplication.ear\Increment.jar.

     Once you have added this JAR file, select it and use the **Attach Source** pushbutton to attach Increment.jar as the source - Increment.jar contains both the source and class files.
   - *installation_root*\lib\j2ee.jar
   - *installation_root*\lib\pagelist.jar
   - *installation_root*\\lib\webcontainer.jar

   Click **OK** when you have added all of the JARs.
9. **(Optional)** You can set some breakpoints in the source at this time if you like, however, it is not necessary as step-by-step mode will prompt you whenever the server calls a method on a Web object.

   Step-by-step mode is explained in more detail below.
10. To start debugging, you need to start the WebSphere Application Server in debug mode and make note of the JVM debug port.

    The default value of the JVM debug port is 7777.
11. Once the server is started, switch to the debug perspective by selecting **Window > Open Perspective > Debug**. You can also enable the debug launch in the Java Perspective by choosing **Window > Customize Perspective** and selecting the **Debug** and **Launch** checkboxes in the **Other** category.
12. Select the workbench toolbar **Debug** pushbutton and then select **WebSphere Application Server Debug** from the list of launch configurations. Click the **New** pushbutton to create a new configuration.
13. Give your configuration a name and select the project to debug (your new WASExamples project). Change the port number if you did not start the server on the default port (7777).

14. Click **Debug** to start debugging.
15. Load one of the examples in your browser (for example, http://localhost:9080/hitcount).

What to do next

To learn more about debugging, launch the Application Server Toolkit, select **Help > Help Contents** and choose the **Debugger Guide bookshelf** entry. To learn about known limitations and problems that are associated with the Application Server Toolkit, see the Application Server Toolkit release notes.

Debugging Service details

Use this page to view and modify the settings used by the Debugging Service.

To view this administrative console page, click **Servers > Application Servers** > *server* > **Debugging Service**.

Startup

Specifies whether the server will attempt to start the Debug service when the server starts.

JVM debug port

Specifies the port that the Java Virtual Machine will listen on for debug connections.

JVM debug arguments

Specifies the debugging argument string used to start the JVM in debug mode.

Debug class filters

Specifies an array of classes to ignore during debugging. When running in step-by-step mode, the debugger whill not stop in classes that match a filter entry.

BSF debug port

Specifies the port that the BSF Debug Manager listens on.

BSF logging level

Specifies the level of logging provided by the BSF Debug Manager. The valid range is 0-3, with 3 being the highest level of logging.

Chapter 9. Working with trace

Use trace to obtain detailed information about the execution of WebSphere Application Server components, including application servers, clients, and other processes in the environment. Trace files show the time and sequence of methods called by WebSphere Application Server base classes, and you can use these files to pinpoint the failure.

Collecting a trace is often requested by IBM technical support personnel. If you are not familiar with the internal structure of WebSphere Application Server, the trace output might not be meaningful to you.

Steps for this task

1. Configure an output destination to which trace data is sent.
2. Enable trace for the appropriate WebSphere Application Server or application components.
3. Run the application or operation to generate the trace data.
4. Analyze the trace data or forward it to the appropriate organization for analysis.

Enabling trace

Trace for an application server process is enabled while the server process runs by using the administrative console. You can configure the application server to start in a trace-enabled state by setting the appropriate configuration properties. You can only enable trace for an application client or stand-alone process at process startup.

Trace strings

By default, the trace service for all WebSphere Application Server components is disabled. To request a change to the current state of the trace service, a trace string is passed to the trace service. This trace string encodes the information detailing which level of trace to enable or disable and for which components.

You can type in Trace strings, or construct them with a user-friendly GUI in the administrative console. Trace strings must conform to a specific grammar for processing by the trace service. The specification of this grammar follows:

```
TRACESTRING=COMPONENT_TRACE_STRING[:COMPONENT_TRACE_STRING]*

 COMPONENT_TRACE_STRING=COMPONENT_NAME=LEVEL=STATE[,LEVEL=STATE]*

LEVEL = all | entryExit | debug | event

STATE = enabled | disabled

COMPONENT_NAME = COMPONENT | GROUP
```

The COMPONENT_NAME is the name of a component or group registered with the trace service. Typically, WebSphere Application Server components register using a fully qualified Java classname, for example com.ibm.servlet.engine.ServletEngine. In addition, you can use a wildcard character of asterisk (*) to terminate a component name and indicate multiple classes or

packages. For example, use a component name of com.ibm.servlet.* to specify all components whose names begin with com.ibm.servlet.

Examples of legal trace strings include:

```
com.ibm.ejs.ras.ManagerAdmin=debug=enabled
com.ibm.ejs.ras.ManagerAdmin=all=enabled,event=disabled
com.ibm.ejs.ras.*=all=enabled
com.ibm.ejs.ras.*=all=enabled:com.ibm.ws.ras=debug=enabled,entryexit=enabled
```

Trace strings cannot contain blanks.

Trace strings are processed from left to right. Specifying a trace string like

```
abc.*=all=enabled,event=disabled
```

first enables all trace for all components whose names start with `abc`, then disables event tracing for those same components. This means that the trace string

```
abc.*=all=enabled,event=disabled
```

is equivalent to

```
abc.*=debug=enabled,entryexit=enabled
```

Enabling trace at server startup

The diagnostic trace configuration settings for a server process determines the initial trace state for a server process. The configuration settings are read at server startup and used to configure the trace service. You can also change many of the trace service properties or settings while the server process is running.

Steps for this task

1. Start the administrative console.
2. Click **Troubleshooting > Logs and Trace** in the console navigation tree, then click **Server > Diagnostic Trace**.
3. Click **Configuration**.
4. Select the **Enable Trace** check box to enable trace, clear the check box to disable trace.
5. Set the trace specification to the desired state by entering the proper TraceString.
6. Select whether to direct trace ouput to either a file or an in-memory circular buffer.
7. **(Optional)** If the in-memory circular buffer is selected for the trace output set the size of the buffer, specified in thousands of entries.

   This is the maximum number of entries that will be retained in the buffer at any given time.
8. **(Optional)** If a file is selected for trace output, set the maximum size in megabytes to which the file should be allowed to grow.

   When the file reaches this size, the existing file will be closed, renamed, and a new file with the original name reopened. The new name of the file will be based upon the original name with a timestamp qualifier added to the name. In addition, specify the number of history files to keep.
9. Select the desired format for the generated trace.
10. Save the changed configuration.
11. Start the server.

Enabling trace on a running server

You can modify the trace service state that determines which components are being actively traced for a running server by using the following procedure.

Steps for this task

1. Start the administrative console.
2. Click **Troubleshooting > Logging and Tracing** in the console navigation tree, then click **server > Diagnostic Trace**.
3. Select the **Runtime** tab.
4. **(Optional)** Select the **Save Trace** check box if you want to write your changes back to the server configuration.
5. Change the existing trace state by changing the trace specification to the desired state.
6. **(Optional)** Configure the trace output if a change from the existing one is desired.
7. Click **Apply**.

Enabling trace on an application client or stand-alone process

Many stand-alone processes and the WebSphere J2EE application client, define a process-specific mechanism for enabling trace. To enable trace and message logging for application clients or processes that have not defined such a mechanism, add the `-DtraceSettingsFile=filename` system property to the startup script or command.

For example, to trace the stand-alone client application program named com.ibm.sample.MyClientProgram, enter the command:

```
java -DtraceSettingsFile=MyTraceSettings.properties com.ibm.samples.MyClientProgram
```

The file identified by *filename* must be a properties file placed in the classpath of the application client or stand-alone process. An example file is provided in `install_root/properties/TraceSettings.properties`.

Steps for this task

1. **(Optional)** Configure the `MyTraceSettings.properties` file to send trace output to a file.

   The `traceFileName` property in the trace settings file specifies one of two options:

   - The fully qualified name of an output file. For example, `traceFileName=c:\\MyTraceFile.log`. You must specify this property to generate visible output.
   - stdout. When specified, output is written to `System.out`.
2. **(Optional)** Specify a trace string for writing messages.

   The `Trace String` property specifies a startup trace specification, similar to that available on the server.

   For your convenience, enter multiple individual trace strings into the trace settings file, one trace string per line.

Results

Here are the results of using each optional property setting:

- Specify a valid setting for the traceFileName property without a trace string to write messages to the specified file or System.out only.
- Specify a trace string without a traceFileName property value to generate no output.
- Specify both a valid traceFileName property and a trace string to write both message and trace entries to the location specified in the traceFileName property.

Enabling trace on client and stand-alone applications

The WebSphere J2EE application client and many stand-alone processes define a process-specific mechanism for enabling trace.

Steps for this task

1. To enable trace and message logging for client applications or processes that have not defined such a mechanism, add the DtraceSettingsFile= *filename* system property to the startup script or command. To trace the stand-alone client application program named com.ibm.sample.MyClientProgram, you would enter the following command:

   ```
   java -DtraceSettingsFile=MyTraceSettings.properties com.ibm.samples.MyClientProgram
   ```

   The file identified by *filename* must be a properties file placed in the classpath of the application client or stand-alone process. An example file is provided in *install_root* /properties/TraceSettings.properties.
2. You can configure the MyTraceSettings.properties file to send trace output to a file using the traceFileName property. Specify one of two options:
   - The fully qualified name of an output file. For example, traceFileName=c:\\MyTraceFile.log. You must specify this property to generate visible output.
   - stdout. When specified, output is written to System.out.
3. You can also specify a trace string for writing messages with the Trace String property, Specify a startup trace specification similar to that available on the server.

   For your convenience, you can enter multiple individual trace strings into the trace settings file, one trace string per line.

Results

Here are the results of using each optional property setting:

- Specify a valid setting for the traceFileName property without a trace string to write messages to the specified file or System.out only.
- Specify a trace string without a traceFileName property value to generate no output.
- Specify both a valid traceFileName property and a trace string to write both message and trace entries to the location specified in the traceFileName property.

Managing the application server trace service

You can manage the trace service for a server process while the server is stopped and while it is running. You can specify which components to trace, where to send trace output, the characteristics of the trace output device, and which format to generate trace output in.

Steps for this task

1. Start the administrative console.
2. Click **Troubleshooting > Logging and Tracing** in the console navigation tree, then click *server* **> Diagnostic Trace**
3. If the server is running, select the **Runtime** tab.
4. **(Optional)** For a running server, check the Save trace check box to write your changes back to the server configuration.

   If Save trace is not selected, the changes you make will apply only for the life of the server process that is currently running.
5. Perform the desired operation:
   a. Enter the file name and click **Dump** to dump the in-memory circular buffer.
   b. To change the trace destination from a file to the in-memory circular buffer or to a different file, or to change from the in memory circular buffer to a file, select the appropriate radio buttons, then click Apply.
   c. To change the format in which trace output is generated, select the appropriate value from the drop-down list.

Interpreting trace output

On an application server, trace output can be directed either to a file or to an in-memory circular buffer. If trace output is directed to the in-memory circular buffer, it must be dumped to a file before it can be viewed.

On an application client or stand-alone process, trace output can be directed either to a file or to the process console window.

In all cases, trace output is generated as plain text in either basic, advanced or log analyzer format as specified by the user. The basic and advanced formats for trace output are similar to the basic and advanced formats that are available for the JVM message logs.

Basic and advanced format fields

Basic and Advanced Formats use many of the same fields and formatting techniques. The fields that can be used in these formats include:

TimeStamp
The timestamp is formatted using the locale of the process where it is formatted. It includes a fully qualified date (YYMMDD), 24 hour time with millisecond precision and the time zone.

ThreadId
An 8 character hexidecimal value generated from the hash code of the thread that issued the trace event.

ShortName
The abbreviated name of the logging component that issued the trace event. This is typically the class name for WebSphere Application Server internal components, but may be some other identifier for user applications.

LongName
The full name of the logging component that issued the trace event. This is typically the fully qualified class name for WebSphere Application Server internal components, but may be some other identifier for user applications.

EventType
A one character field that indicates the type of the trace event. Trace types are in lower case. Possible values include:

> a trace entry of type method entry.

< a trace entry of type method exit.

e a trace entry of type event.

d a trace entry of type debug.

m a trace entry of type dump.

u a trace entry of type unconditional.

Z a placeholder to indicate that the trace type was not recognized.

ClassName
The class that issued the message or trace event.

MethodName
The method that issued the message or trace event.

Organization
The organization that owns the application that issued the message or trace event.

Product
The product that issued the message or trace event.

Component
The component within the product that issued the message or trace event.

Basic format

Trace events displayed in basic format use the following format:

```
<timestamp><threadId><shortName><eventType>[className][methodName]<textmessage>
              [parameter 1]
              [parameter 2]
```

Advanced formats

Trace events displayed in advanced format use the following format:

```
<timestamp><threadId><eventType><UOW><source=longName>
[className][methodName]<Organization><Product><Component>
<textMessage>[parameter 1=parameterValue][parameter 2=parameterValue]
```

Log analyzer

Specifying the log analyzer format allows you to open trace output using the Log Analyzer. This is useful if you are trying to correlate traces from two different server processes, because it allows you to use the Log Analyzer's merge capability.

Diagnostic trace service settings

Use this page to review and modify the properties of the diagnostic trace service.

To view this page, click **Troubleshooting > Logs and Trace >** *server* **> Diagnostic trace**.

Enable Trace

Enables the trace service.

If this option is not selected, the following configuration properties are not passed to the application server trace service at server startup.

Save Trace

Save changes made on the runtime tab to the trace configuration as well.

Select this box to copy runtime trace changes to the trace configuration settings as well. Saving these changes to the trace configuration will cause the changes to persist even if the application is restarted.

Trace Specification

Specifies tracing details.

Enter a trace string that specifies the components, packages, or groups to trace. The trace string must conform to the specific grammar described below. You can enter the trace string directly, or generate it using the graphical trace interface. Click **Modify** to start the graphical trace interface.

If you start the graphical trace interface from the configuration tab, a list of well-known components, packages, and groups is displayed. This list might not be exhaustive.

If you start the graphical trace interface from the runtime tab, the list of components, packages, and groups displays all such components currently registered on the running server.

The format of the trace specification is:

`<component> = <trace_type>= enabled | disabled`

where: <*component*> is the component for which to enable or disable tracing, and <*trace_type*> is the type of tracing to enable or disable. Separate multiple tracing specifications with colons (:).

Components correspond to Java packages and classes, or to collections of Java packages. Use * as a wildcard to indicate components that include all classes in all packages contained by the specified component. For example:

***** Specifies all traceable code running in the application server, including WebSphere Application Server system code and customer code.

com.ibm.ws.* specifies all classes whose package name begins with `com.ibm.ws`.

com.ibm.ws.classloader.JarClassLoader Specifies only the JarClassLoader class.

For more information on trace string grammar, see the article *Enabling trace* in the WebSphere Application Server InfoCenter.

Note: An error can occur when setting a trace specification from the adminstrative console if selections are made from both the Groups and Components lists. In some cases, the selection made from one list is not lost when adding a selection

from the other list. To work around this problem, enter the desired trace specification directly into the Trace Specification entry field.

Trace Output

Specifies where trace output should be written.

The trace output can be written directly to an output file, or stored in memory and written to a file on demand using the Dump button found on the runtime page.

Memory Buffer
Specifies that the trace output should be written to an in-memory circular buffer. If you select this option you must specify the following parameters:

Maximum Buffer Size
Specifies the number of entries, in thousands, that can be cached in the buffer. When this number is exceeded, older entries are overwritten by new entries.

Dump File Name
The name of the file to which the memory buffer will be written when it is dumped. This option is only available from the runtime tab.

File Specifies to write the trace output to a self-managing log file.

The self-managing log file writes messages to the file until a size criteria is reached. When the file reaches the specified size, logging is temporarily suspended and the log file is closed and renamed. The new name is based on the original name of the file, plus a timestamp qualifier that indicates when the renaming occurred. Once the renaming is complete, a new, empty log file with the original name is reopened, and logging resumes. No messages are lost as a result of the rollover, although a single message may be split across the two files.

If you select this option you must specify the following parameters:

Maximum File Size
Specifies the maximum size, in megabytes, to which the output file is allowed to grow.

This attribute is only valid if the File Size attribute is selected. When the file reaches this size, it is rolled over as described above.

Maximum Number of Historical Files
Specifies the maximum number of rolled over files to keep.

File Name
Specifies the name of the file to which the trace output is written.

Trace Output Format

Specifies the format of the trace output.

You can specify one of three levels for trace output:

Basic (Compatible)
Preserves only basic trace information. Select this option to minimize the amount of space taken up by the trace output.

Advanced
Preserves more specific trace information. Select this option to see detailed trace information for use in troubleshooting and problem determination.

Log Analyzer
Preserves trace information in a format that is compatible with the Log Analyzer tool. Select this option if you want to use the trace output as input to the Log Analyzer tool.

Logging and tracing settings

Use this page to view and configure logging and trace settings for the server.

To view this administrative console page, click **Troubleshooting > Logs and Trace** *server_name* > *service_type*.

Chapter 10. Adding logging and tracing to your application

Designers and developers of applications that run with or under WebSphere Application Server, such as servlets, JSP files, enterprise beans, client applications, and their supporting classes, may find it useful to use the same facility for generating messages that WebSphere Application Server itself uses, JRas.

This approach has advantages over simply adding `System.out.println()` statements to your code:

- Your messages appear in the WebSphere Application Server standard message format with additional data, such as a date and time stamp, added automatically.
- You can more easily correlate problems and events in your own application to problems and events associated with WebSphere Application Server components.
- You can take advantage of the WebSphere Application Server log file management features.
- You can view your messages with the WebSphere Application Server user-friendly Log Analyzer tool.

Unless you choose to extend the JRas framework, using the JRas API set is only slightly more complicated than writing basic `System println()` statements.

Programming with the JRas framework

Use the JRas extensions to incorporate message logging and diagnostic trace into WebSphere Application Server applications. The JRas extensions are based on the stand-alone JRas logging toolkit.

Steps for this task

1. Retrieve a reference to the JRas manager.
2. Retrieve message and trace loggers by using methods on the returned manager.
3. Call the appropriate methods on the returned message and trace loggers to create message and trace entries, as appropriate.

Understanding the JRas facility

Developing, deploying and maintaining applications are complex tasks. For example, when a running application encounters an unexpected condition it might not be able to complete a requested operation. In such a case you might want the application to inform the administrator that the operation has failed and give information as to why. This enables the administrator to take the proper corrective action. Application developers or maintainers might need to gather detailed information relating to the execution path of a running application in order to determine the root cause of a failure that is due to a code bug. The facilities that are used for these purposes are typically referred to as message logging and diagnostic trace.

Message logging and diagnostic trace are conceptually quite similar, but do have important differences. It is important for application developers to understand these differences in order to use these tools properly. To start with, the following operational definitions of messages and trace are provided.

Message
A message entry is an informational record intended to be viewed by end users, systems administrators and support personnel. The text of the message must be clear, concise and interpretable by an end user. Messages are typically localized, meaning they are displayed in the national language of the end user. Although the destination and lifetime of messages might be configurable, some level of message logging is always enabled in normal system operation. Message logging must be used judiciously due to both performance considerations and the size of the message repository.

Trace A trace entry is an information record that is intended to be used by service engineers or developers. As such a trace record may be considerably more complex, verbose and detailed than a message entry. Localization support is typically not used for trace entries. Trace entries may be fairly inscrutable, understandable only by the appropriate developer or service personnel. It is assumed that trace entries are not written during normal runtime operation, but may be enabled as needed to gather diagnostic information.

WebSphere Application Server provides a message logging and diagnostic trace API that can be used by applications. This API is based on the stand-alone JRas logging toolkit which was developed by IBM. The stand-alone JRas logging toolkit is a collection of interfaces and classes that provide message logging and diagnostic trace primitives. These primitives are not tied to any particular product or platform. The stand-alone JRas logging toolkit provides a limited amount of support (typically referred to as systems management support, including log file configuration support based on property files.

As designed, the stand-alone JRas logging toolkit does not contain the support required for integration into the WebSphere Application Server runtime or for usage in a J2EE environment. To overcome these limitations, WebSphere Application Server provides a set of extension classes to address these shortcomings. This collection of extension classes is referred to as the JRas extensions. The JRas extensions do not modify the interfaces introduced by the stand-alone JRas logging toolkit, but simply provide the appropriate implementation classes. The conceptual structure introduced by the stand-alone JRas logging toolkit is described below. It is equally applicable to the JRas extensions.

JRas Concepts

The following is a basic overview of important concepts and constructs introduced by the stand-alone JRas logging toolkit. It is not meant to be an exhaustive overview of the capabilities of this logging toolkit, nor is it intended to be a detailed discussion of usage or programming paradigms. More detailed information, including code examples, is available in JRas extensions and its subtopics, including in the javadoc for the various interfaces and classes that make up the logging toolkit.

Event Types
The stand-alone JRas logging toolkit defines a set of event types for messages and a set of event types for trace. Examples of message types include informational, warning and error. Examples of trace types include entry, exit and trace.

Event Classes

The stand-alone JRas logging toolkit defines both message and trace event classes.

Loggers

A logger is the primary object with which the user code interacts. Two types of loggers are defined. These are message loggers and trace loggers. The set of methods on message loggers and trace loggers are different, since they provide different functionality. Message loggers create only message records and trace loggers create only trace records. Both types of loggers contain masks that indicates which categories of events the logger should process and which it should ignore. Although every JRas logger is defined to contain both a message and trace mask, the message logger only uses the message mask and the trace logger only uses the trace mask. For example, by setting a message logger's message mask to the appropriate state, it can be configured to process only Error messages and ignore Informational and Warning messages. Changing the state of a message logger's trace mask has no effect.

A logger contains one or more handlers to which it forwards events for further processing. When the user calls a method on the logger, the logger will compare the event type specified by the caller to its current mask value. If the specified type passes the mask check, the logger will create an event object to capture the information relating to the event that was passed to the logger method. This information may include information such as the names of the class and method which is logging the event, a message and parameters to log, among others. Once the logger has created the event object, it forwards the event to all handlers currently registered with the logger.

Methods that are used within the logging infrastructure itself should not make calls to the logger method. When an application uses an object that extends a thread class, implements the hashCode(), and makes a call to the logging infrastructure from that method, the result is a recursive loop.

Handlers

A handler provides an abstraction over an output device or event consumer. An example is a file handler, which knows how to write an event to a file. The handler also contains a mask that is used to further restrict the categories of events the handler will process. For example, a message logger may be configured to pass both warning and error events, but a handler attached to the message logger may be configured to only pass error events. Handlers also include formatters, which the handler invokes to format the data in the passed event before it is written to the output device.

Formatters

Handlers are configured with Formatters, which know how to format events of certain types. A handler may contain multiple formatters, each of which know how to format a specific class of event. The event object is passed to the appropriate formatter by the handler. The formatter returns formatted output to the handler, which then writes it to the output device.

JRas Extensions

JRas extensions

The stand-alone JRas logging toolkit defines interfaces and provides a variety of concrete classes that implement these interfaces. Since the stand-alone JRas logging

toolkit was developed as a general purpose toolkit, the implementation classes do not contain the configuration interfaces and methods necessary for use in the WebSphere Application Server product. In addition, many of the implementation classes are not written appropriately for use in a J2EE environment. To overcome these shortcomings, WebSphere Application Server provides the appropriate implementation classes that allow integration into the WebSphere Application Server environment. The collection of these implementation classes is referred to as the JRas extensions.

Usage Model

You can use the JRas extensions in three distinct operational modes:

Integrated
In this mode, message and trace records are written only to logs defined and maintained by the WebSphere Application Server runtime. This is the default mode of operation and is equivalent to the WebSphere Application Server 4.0 mode of operation.

stand-alone
In this mode, message and trace records are written solely to stand-alone logs defined and maintained by the user. You control which categories of events are written to which logs, and the format in which entries are written. You are responsible for configuration and maintenance of the logs. Message and trace entries are not written to WebSphere Application Server runtime logs.

Combined
In this mode message and trace records are written to both WebSphere Application Server runtime logs and to stand-alone logs that you must define, control, and maintain. You can use filtering controls to determine which categories of messages and trace are written to which logs.

The JRas extensions are specifically targeted to an integrated mode of operation. The integrated mode of operation can be appropriate for some usage scenarios, but there many scenarios are not adequately addressed by these extensions. Many usage scenarios require a stand-alone or combined mode of operation instead. A set of user extension points has been defined that allow the JRas extensions to be used in either a stand-alone or combined mode of operations.

JRas extension classes

WebSphere Application Server provides a base set of implementation classes that collectively are referred to as the JRas extensions. Many of these classes provide the appropriate implementations of loggers, handlers and formatters for use in a WebSphere Application Server environment. As previously noted, this collection of classes is targeted at an Integrated mode of operation. If you choose to use the JRas extensions in either stand-alone or combined mode, you can reuse the logger and manager class provided by the extensions, but you must provide your own implementations of handlers and formatters.

WebSphere Application Server Message and Trace loggers

The message and trace loggers provided by the stand-alone JRas logging toolkit cannot be directly used in the WebSphere Application Server environment. The JRas extensions provide the appropriate logger implementation classes. Instances of these message and trace logger classes are obtained directly and exclusively from the WebSphere Application Server Manager class, described below. You cannot

directly instantiate message and trace loggers. Obtaining loggers in any manner other than directly from the Manager is not allowed. Doing so is a direct violation of the programming model.

The message and trace loggers instances obtained from the WebSphere Application Server Manager class are subclasses of the `RASMessageLogger()` and `RASTraceLogger()` classes provided by the stand-alone JRas logging toolkit. The `RASMessageLogger()` and `RASTraceLogger()` classes define the set of methods that are directly available. Public methods introduced by the JRas extensions logger subclasses cannot be called directly by user code. Doing so is a violation of the programming model.

Loggers are named objects and are identified by name. When the Manager class is called to obtain a logger, the caller is required to specify a name for the logger. The Manager class maintains a name-to-logger instance mapping. Only one instance of a named logger will ever be created within the lifetime of a process. The first call to the Manager with a particular name will result in the logger being created and configured by the Manager. The Manager will cache a reference to the instance, then return it to the caller. Subsequent calls to the Manager that specify the same name will result in a reference to the cached logger being returned. Separate namespaces are maintained for message and trace loggers. This means a single name can be used to obtain both a message logger and a trace logger from the Manager, without ambiguity, and without causing a namespace collision.

In general, loggers have no predefined granularity or scope. A single logger could be used to instrument an entire application. Or users may determine that having a logger per class is more desirable. Or the appropriate granularity may lie somewhere in between. Partitioning an application into logging domains is rightfully determined by the application writer.

The WebSphere Application Server logger classes obtained from the Manager are thread-safe. Although the loggers provided as part of the stand-alone JRas logging toolkit implement the serializable interface, in fact loggers are not serializable. Loggers are stateful objects, tied to a Java virtual machine instance and are not serializable. Attempting to serialize a logger is a violation of the programming model.

Please note that there is no provision for allowing users to provide their own logger subclasses for use in a WebSphere Application Server environment.

WebSphere Application Server handlers

WebSphere Application Server provides the appropriate handler class that is used to write message and trace events to the WebSphere Application Server run-time logs. You cannot configure the WebSphere Application Server handler to write to any other destination. The creation of a WebSphere Application Server handler is a restricted operation and not available to user code. Every logger obtained from the Manager comes preconfigured with an instance of this handler already installed. You can remove the WebSphere Application Server handler from a logger when you want to run in stand-alone mode. Once you have removed it, you cannot add the WebSphere Application Server handler again to the logger from which it was removed (or any other logger). Also, you cannot directly call any method on the WebSphere Application Server handler. Attempting to create an instance of the WebSphere Application Server handler, to call methods on the WebSphere Application Server handler or to add a WebSphere Application Server handler to a logger by user code is a violation of the programming model.

WebSphere Application Server formatters

The WebSphere Application Server handler comes preconfigured with the appropriate formatter for data that is written to WebSphere Application Server logs. The creation of a WebSphere Application Server formatter is a restricted operation and not available to user code. No mechanism exists that allows the user to obtain a reference to a formatter installed in a WebSphere Application Server handler, or to change the formatter a WebSphere Application Server handler is configured to use.

WebSphere Application Server manager

WebSphere Application Server provides a Manager class located in the `com.ibm.websphere.ras` package. All message and trace loggers must be obtained from this Manager. A reference to the Manager is obtained by calling the static `Manager.getManager()` method. Message loggers are obtained by calling the `createRASMessageLogger()` method on the Manager. Trace loggers are obtained by calling the `createRASTraceLogger()` method on the Manager class.

The manager also supports a *group* abstraction that is useful when dealing with trace loggers. The group abstraction allows multiple, unrelated trace loggers to be registered as part of a named entity called a group. WebSphere Application Server provides the appropriate systems management facilities to manipulate the trace setting of a group, similar to the way the trace settings of an individual trace logger.

For example, suppose component A consist of 10 classes. Suppose each class is configured to use a separate trace logger. Suppose all 10 trace loggers in the component are registered as members of the same group (for example Component_A_Group). You can then turn on trace for a single class. Or you can turn on trace for all 10 classes in a single operation using the group name if you want a component trace. Group names are maintained within the namespace for trace loggers.

Extending the JRas framework

Since the Jras extensions classes do not provide the flexibility and behavior required for many scenarios, a variety of extension points have been defined. You are allowed to write your own implementation classes to obtain the required behavior.

In general, the JRas extensions require you to call the Manager class to obtain a message logger or trace logger. No provision is made to allow you to provide your own message or trace logger subclasses. In general, user-provided extensions cannot be used to affect the integrated mode of operation.The behavior of the integrated mode of operation is solely determined by the WebSphere Application Server run-time and the JRas extensions classes.

Handlers

The stand-alone JRas logging toolkit defines the `RASIHandler` interface. All handlers must implement this interface. You can write your own handler classes that implement the `RASIHandler` interface. You should directly create instances of user-defined handlers and add them to the loggers obtained from the Manager.

The stand-alone JRas logging toolkit provides several handler implementation classes. These handler classes are inappropriate for usage in the J2EE environment. You cannot directly use or subclass any of the Handler classes provided by the stand-alone JRas logging toolkit. Doing so is a violation of the programming model.

Formatters

The stand-alone JRas logging toolkit defines the `RASIFormatter` interface. All formatters must implement this interface. You can write your own formatter classes that implement the `RASIFormatter` interface. You can only add these classes to a user-defined handler. WebSphere Application Server handlers cannot be configured to use user-defined formatters. Instead, directly create instances of your formatters and add them to the your handlers appropriately.

As with handlers, the stand-alone JRas logging toolkit provides several formatter implementation classes. Direct usage of these formatter classes is not supported.

Message event types

The stand-alone JRas toolkit defines message event types in the `RASIMessageEvent` interface. In addition, the WebSphere Application Server reserves a range of message event types for future use. The `RASIMessageEvent` interface defines three types, with values of 0x01, 0x02, and 0x04. The values 0x08 through 0x8000 are reserved for future use. You can provide your own message event types by extending this interface appropriately. User-defined message types must have a value of 0x1000 or greater.

Message loggers retrieved from the Manager have their message masks set to *pass* or process all message event types defined in the `RASIMessageEvent` interface. In order to process user-defined message types, you must manually set the message logger mask to the appropriate state by user code after the message logger has been obtained from the Manager. WebSphere Application Server does not provide any built-in systems management support for managing any message types.

Message event objects

The stand-alone JRas toolkit provides a `RASMessageEvent` implementation class. When a message logging method is called on the message logger, and the message type is currently enabled, the logger creates and distributes an event of this class to all handlers currently registered with that logger.

You can provide your own message event classes, but they must implement the `RASIEvent` interface. You must directly create instances of such user-defined message event classes. Once it is created, pass your message event to the message logger by calling the message logger's `fireRASEvent()` method directly. WebSphere Application Server message loggers cannot directly create instances of user-defined types in response to calling a logging method (`msg()`, `message()`...) on the logger. In addition, instances of user-defined message types are never processed by the WebSphere Application Server handler. You cannot create instances of the `RASMessageEvent` class directly.

Trace event types

The stand-alone JRas toolkit defines trace event types in the `RASITraceEvent` interface. You can provide your own trace event types by extending this interface

appropriately. In such a case you must ensure that the values for the user-defined trace event types do not collide with the values of the types defined in the RASITraceEvent interface.

Trace loggers retrieved from the Manager typically have their trace masks set to reject all types. A different starting state can be specified by using WebSphere Application Server systems management facilities. In addition, the state of the trace mask for a logger can be changed at run-time using WebSphere Application Server systems management facilities.

In order to process user-defined trace types, the trace logger mask must be manually set to the appropriate state by user code. WebSphere Application Server systems management facilities cannot be used to manage user-defined trace types, either at start time or run-time.

Trace event objects

The stand-alone JRas toolkit provides a RASTraceEvent implementation class. When a trace logging method is called on the WebSphere Application Server trace logger and the type is currently enabled, the logger creates and distributes an event of this class to all handlers currently registered with that logger.

You can provide your own trace event classes. Such trace event classes must implement the RASIEvent interface. You must create instances of such user-defined event classes directly. Once it is created, pass the trace event to the trace logger by calling the trace logger's fireRASEvent() method directly. WebSphere Application Server trace loggers cannot directly create instances of user-defined types in response to calling a trace method (entry(), exit(), trace()) on the trace logger. In addition, instances of user-defined trace types are never processed by the WebSphere Application Server handler. You cannot create instances of the RASTraceEvent class directly.

User defined types, user defined events and WebSphere Application Server

By definition, the WebSphere Application Server handler will process user-defined message or trace types, or user-defined message or trace event classes. Message and trace entries of either a user-defined type or user-defined event class cannot be written to the WebSphere Application Server run-time logs.

Writing User Extensions

General Considerations

You can configure the WebSphere Application Server to use Java 2 security to restrict access to protected resources such as the file system and sockets. Since user written extensions typically access such protected resources, user written extensions must contain the appropriate security checking calls, using AccessController doPrivileged() calls. In addition, the user written extensions must contain the appropriate policy file. In general, it is recommended that you locate user written extensions in a separate package. It is your responsibility to restrict access to the user written extensions appropriately.

Writing a handler

User written handlers must implement the RASIHandler interface. The RASIHandler interface extends the RASIMaskChangeGenerator interface, which extends the

RASIObject interface. A short discussion of the methods introduced by each of these interfaces follows, along with implementation pointers. For more in depth information on any of the particular interfaces or methods, see the corresponding product javadoc.

RASIObject interface

The RASIObject interface is the base interface for stand-alone JRas logging toolkit classes that are stateful or configurable, such as loggers, handlers and formatters.

- The stand-alone JRas logging tookit supports rudimentary properties-file based configuration. To implement this configuration support, the configuration state is stored as a set of key-value pairs in a properties file. The methods `public Hashtable getConfig()` and `public void setConfig(Hashtable ht)` are used to get and set the configuration state. The JRas extensions do not support properties based configuration and it is recommended that these methods be implemented as no-operations. You can implement your own properties based configuration using these methods.
- Loggers, handlers and formatters can be named objects. For example, the JRas extensions require the user to provide a name for the loggers that are retrieved from the manager. You can name your handlers. The methods `public String getName()` and `public void setName(String name)` are provided to get or set the name field. The JRas extensions currently do not call these methods on user handlers. You can implement these methods as you want, including as no operations.
- Loggers, handlers and formatters can also contain a description field. The methods `public String getDescription()` and `public void setDescription(String desc)` can be used to get or set the description field. The JRas extensions currently do not use the description field. You can implement these methods as you want, including as no operations.
- The method `public String getGroup()` is provided for usage by the RASManager. Since the JRas extensions provide their own Manager class, this method is never called. It is recommended you implement this as a no-operation.

RASIMaskChangeGenerator interface

The RASIMaskChangeGenerator interface is the interface that defines the implementation methods for filtering of events based on a mask state. This means that it is currently implemented by both loggers and handlers. By definition, an object that implements this interface contains both a message mask and a trace mask, although both need not be used. For example, message loggers contain a trace mask, but the trace mask is never used since the message logger never generates trace events. Handlers however can actively use both mask values. For example a single handler could handle both message and trace events.

- The methods `public long getMessageMask()` and `public void setMessageMask(long mask)` are used to get or set the value of the message mask. The methods `public long getTraceMask()` and `public void setTraceMask(long mask)` are used to get or set the value of the trace mask.

In addition, this interface introduces the concept of *calling back* to interested parties when a mask changes state. The callback object must implement the RASIMaskChangeListener interface.

- The methods `public void addMaskChangeListener(RASIMaskChangeListener listener)` and `public void removeMaskChangeListener(RASIMaskChangeListener listener)` are used to add or remove listeners to the handler. The method

`public Enumeration getMaskChangeListeners()` returns an Enumeration over the list of currently registered listeners. The method `public void fireMaskChangedEvent(RASMaskChangeEvent mc)` is used to call back all the registered listeners to inform them of a mask change event.

For efficiency reasons, the Jras extensions message and trace loggers implement the `RASIMaskChangeListener` interface. The logger implementations maintain a "composite mask" in addition to the logger's own mask. The logger's composite mask is formed by logically *or'ing* the appropriate masks of all handlers that are registered to that logger, then *and'ing* the result with the logger's own mask. For example, the message logger's composite mask is formed by or'ing the message masks of all handlers registered with that logger, then and'ing the result with the logger's own message mask.

This means that all handlers are required to properly implement these methods. In addition, when a user handler is instantiated, the logger it is to be added to should be registered with the handler using the `addMaskChangeListener()` method. When either the message mask or trace mask of the handler is changed, the logger must be called back to inform it of the mask change. This allows the logger to dynamically maintain the composite mask.

The `RASMaskChangedEvent` class is defined by the stand-alone JRas logging toolkit. Direct usage of that class by user code is allowed in this context.

In addition the `RASIMaskChangeGenerator` introduces the concept of caching the names of all message and trace event classes that the implementing object will process. The intent of these methods is to allow a management program such as a GUI to retrieve the list of names, introspect the classes to determine the event types that they might possibly process and display the results. The JRas extensions do not ever call these methods, so they can be implemented as no operations, if desired.

- The methods `public void addMessageEventClass(String name)` and `public void removeMessageEventClass(String name)` can be called to add or remove a message event class name from the list. The method `public Enumeration getMessageEventClasses()` will return an enumeration over the list of message event class names. Similarly, the `public void addTraceEventClass(String name)` and `public void removeTraceEventClass(String name)` can be called to add or remove a trace event class name from the list. The `method public Enumeration getTraceEventClasses()` will return an enumeration over the list of trace event class names.

RASIHandler interface

The `RASIHandler` interface introduces the methods that are specific to the behavior of a handler.

The `RASIHandler` interface as provided by the stand-alone JRas logging toolkit supports handlers that run in either a synchronous or asynchronous mode. In asynchronous mode, events are typically queued by the calling thread and then written by a worker thread. Since spawning of threads is not allowed in the WebSphere Application Server environment, it is expected that handlers will not queue or batch events, although this is not expressly prohibited.

- The methods `public int getMaximumQueueSize()` and `public void setMaximumQueueSize(int size)` throw `IllegalStateException` are provided to manage the maximum queue size. The method `public int getQueueSize()` is provided to query the actual queue size.
- The methods `public int getRetryInterval()` and `public void setRetryInterval(int interval)` support the notion of error retry, which again implies some type of queueing.
- The methods `public void addFormatter(RASIFormatter formatter)`, `public void removeFormatter(RASIFormatter formatter)` and `public Enumeration getFormatters()` are provided to manage the list of formatters that the handler can be configured with. Different formatters can be provided for different event classes, if appropriate.
- The methods `public void openDevice()`, `public void closeDevice()` and `public void stop()` are provided to manage the underlying device that the handler abstracts.
- The methods `public void logEvent(RASIEvent event)` and `public void writeEvent(RASIEvent event)` are provided to actually pass events to the handler for processing.

Writing a formatter

User written formatters must implement the `RASIFormatter` interface. The `RASIFormatter` interface extends the `RASIObject` interface. The implementation of the `RASIObject` interface is the same for both handlers and formatters. A short discussion of the methods introduced by the `RASIFormatter` interface follows. For more in depth information on the methods introduced by this interface, see the corresponding product javadoc.

RASIFormatter interface

- The methods `public void setDefault(boolean flag)` and `public boolean isDefault()` are used by the concrete RASHandler classes provided by the stand-alone JRas logging toolkit to determine if a particular formatter is the default formatter. Since these `RASHandler` classes must never be used in a WebSphere Application Server environment, the semantic significance of these methods can be determined by the user.
- The methods `public void addEventClass(String name)`, `public void removeEventClass(String name)` and `public Enumeration getEventClasses()` are provided to determine which event classes a formatter can be used to format. You can provide the appropriate implementations as you see fit.
- The method `public String format(RASIEvent event)` is called by handler objects and returns a formatted String representation of the event.

Example: user written handler

The following is a very simple sample of a Handler class that writes formatted events to a file. This class is functional, but is intended solely to demonstrate concepts. For simplicity and clarity, much code (including appropriate boundary condition checking logic) has been ignored. This sample is not intended to be an example of good programming practice.

```
package com.ibm.ws.ras.test.user;

import com.ibm.ras.*;
import java.io.*;
import java.util.*;
import java.security.AccessController;
import java.security.PrivilegedAction;
```

```
import java.security.PrivilegedExceptionAction;
import java.security.PrivilegedActionException;

   /**
   * The <code>SimpleFileHandler</code> is a class that
   * implements the {{@link RASIHandler} interface. It is a simple
   * Handler that writes to a file. The name of the file must be specified
   * in the constructor.
   * <p>
   *If the file includes a path, the path separator may be a front-slash ('/')
   * or the platform-specific path separator character.  For example:
   *
   * /Dir1/Dir2/Dir3/MyStuff.log
   *
   */

public class SimpleFileHandler implements RASIHandler
{
  /**
   * A public boolean that can be inspected by the caller to determine
   * if an error has occurred during an operation.
   * This boolean can only be changed when the device synchronizer is held.
   */
  public boolean errorHasOccurred = false;
  /**
   * The name of the Handler
   */
  private String ivName = "";

  /**
   * The message mask which determines the types of messages that
   * will be processed.
   */
  private long ivMessageMask;



  /**
   * The trace mask which determines the types of trace points
   * that will be processed.
   */
  private long ivTraceMask;

  /**
   * The names of the message event classes which this object
   * processes.
   */
  private Vector ivMessageEventClasses;

  /**
   * The names of the trace event classes which this object
   * processes.
   */
  private Vector ivTraceEventClasses;

  /**
   * The set of {@link RASIMaskChangeListener} which want to be
   * informed of changes to the
   * <code>RASIMaskChangeGenerator</code> message or
   * trace mask configuration.
   */
  private Vector ivMaskChangeListeners;

  /**
   * The fully-qualified, normalized name of the file to which the
   * log entries are written.
   */
```

```
private String ivFqFileName;

/**
 * A boolean flag which indicates whether the device to which this
 * handler sends log
 * entries is open.  It is set to true when the device is open and
 * false otherwise.
 */
private boolean ivDeviceOpen = false;

/**
 * A Hashtable of RASIFormatters keyed by the name of the event class
 *  they format. Each event type can have exactly
 * one formatter. Different event classes can have different formatters.
 */
private Hashtable ivFormatters;

/**
 * An object on which the {@link #closeDevice closeDevice} and
 * {@link #writeEvent writeEvent} methods can synchronize.
 */
private Object ivDeviceLock = new Object();

/**
 * The stream to which formatted log events are written. This stream
 * will wrap a file.
 */
private PrintWriter ivWriter = null;

/**
 * Create a SimpleFileHandler.
 * <p>
 * The constructor will attempt to open a stream in append mode over
 * the specified file. If the operation does not complete
 * successfully, the errorHasOccurred boolean is set  to true. If no
 * exceptions are thrown by this constructor and the
 * errorHasOccurred booleans state is false, the stream is open and
 * the handler is usable.
 * <p>
 * @param name the name assigned to this handler object. Null is tolerated.
 * @param fileName a non-null file name. Caller must guarantee this
 * name is not null. A  fully qualified file name is preferred.
 */
public SimpleFileHandler(String name, String fileName) throws Exception  {
  setName(name);
  ivMessageMask = RASIMessageEvent.DEFAULT_MESSAGE_MASK;
  ivTraceMask = RASITraceEvent.DEFAULT_TRACE_MASK;
  // Allocate the Hashtables and Vectors required.
  ivMaskChangeListeners = new Vector();
  ivMessageEventClasses = new Vector();
  ivTraceEventClasses = new Vector();
  ivFormatters = new Hashtable();
  // Add the default event classes that this handler will process
  addMessageEventClass("com.ibm.ras.RASMessageEvent");
  addTraceEventClass("com.ibm.ras.RASTraceEvent");

  // Get the fully qualified, normalized file name. Open the stream
  File x = new File(fileName);
  ivFqFileName = x.getAbsolutePath();
  openDevice();
}

////////////////////////////////////////////////////////////////////
//
// The following methods are required by the RASIObject interface
//
////////////////////////////////////////////////////////////////////
```

```
/**
 * Return this objects configuration as a set of Properties in a
 * Hashtable.
 * <p>
 * This handler does not support properties-based configuration.
 * Therefore a call to this method always returns null
 * @return null is always returned.
 */
public Hashtable getConfig()  {
  return null;
}

/**
 * Set this objects configuration from the properties in the specified
 * Hashtable.
 * <p>
 * This handler does not support properties-based configuration.
 * This method is a no-operation.
 * @param hashTable a Hashtable containing the properties. Input is
 * ignored.
 */
public void setConfig(Hashtable ht)  {
  return;
}

/**
 * Return the name by which this object is known.
 * <p>
 * @return a String containing the name of this object, or an empty
 * string ("") if the name has not been set.
 */
public String getName()  {
  return ivName;
}

/**
 * Set the name by which this object is known.  If the specified name
 * is  <code>null</code>, the current name is not changed.
 * <p>
 * @param name The new name for this object. Null is tolerated.
 */
public void setName(String name)   {
  if (name != null)
    ivName = name;
}

/**
 * Return the description field of this object.
 * <p>
 * This handler does not use a description field. An empty String is
 * always returned.
 * <p>
 * @return an empty String.
 */
public String getDescription()  {
  return "";
}

/**
 * Set the description field for this object.
 * <p>
 * This handler does not use a description field. Input is ignored
 * and this method does nothing.
 * <p>
 * @param desc The description of this object. Input is ignored.
 */
```

```
public void setDescription(String desc)  {
  return;
}

/**
 * Return the name of the {@link com.ibm.ras.mgr.RASManager RASManager}
 * group
 * with which this object is associated.  This method is only used by
 * the  RAS Manager.
 * <p>
 * This object does not support RASManager configuration. Null is
 * always returned.
 * @return null is always returned.
 */
public String getGroup()  {
  return null;
}

////////////////////////////////////////////////////////////
//
// Methods required by the RASIMaskChangeGenerator interface
//
////////////////////////////////////////////////////////////

/**
 * Return the message mask which defines the set of message types that
 * will be  processed by this Handler. The set of possible
 * types is identified in the {@link RASIMessageEvent}
 * <code>TYPE_XXXX</code> constants.
 * <p>
 * @return The current message mask.
 */
public long getMessageMask()   {
  return ivMessageMask;
}

/**
 * Set the message mask which defines the set of message types
 * that will be  processed by this Handler. The set of possible
 * types is identified in the {@link RASIMessageEvent}
 * <code>TYPE_XXXX</code> constants.
 * The mask value is not validated against these types.
 * <p>
 * @param mask The message mask.
 */
public void setMessageMask(long mask)   {
  RASMaskChangeEvent mc =
            new RASMaskChangeEvent(this, ivMessageMask, mask, true);
  ivMessageMask = mask;
  fireMaskChangedEvent(mc);
}

/**
 * Return the trace mask which defines the set of trace
 * types that will be  processed by this Handler. The set of possible
 * types is identified in the {@link RASITraceEvent}
 * <code>TYPE_XXXX</code> constants.
 * <p>
 * @return The current trace mask.
 */
public long getTraceMask()   {
  return ivTraceMask;
}

/**
 * Set the trace mask which defines the set of trace types that will
 * be processed by this Handler. The set of possible types
```

```
 * is identified in the {@link RASITraceEvent}
 * <code>TYPE_XXXX</code> constants. The specified
 * trace mask value is not validated.
 * <p>
 * @param mask The trace mask.
 */
public void setTraceMask(long mask)   {
  RASMaskChangeEvent mc =
          new RASMaskChangeEvent(this, ivTraceMask, mask, false);
  ivTraceMask = mask;
  fireMaskChangedEvent(mc);
}

/**
 * Add a {@link RASIMaskChangeListener} object to the set of listeners
 * which wish to  be identified of a change in the message
 * or trace mask configuration.  If the specified listener is
 * <code>null</code> or is already registered,
 * this method does nothing.
 * <p>
 * @param listener The mask change listener.
 */
public void addMaskChangeListener(RASIMaskChangeListener listener)  {
  if (listener != null && (!ivMaskChangeListeners.contains(listener)))
    ivMaskChangeListeners.addElement(listener);
}

/**
 * Remove a {@link RASIMaskChangeListener} object from the list
 * of registered listeners  that wish to be informed of changes
 * in the message or trace mask configuration. If the  listener
 * is <code>null</code> or is not registered,
 * this method does nothing.
 * <p>
 * @param listener The mask change listener.
 */
public void removeMaskChangeListener(RASIMaskChangeListener listener)  {
  if (listener != null && ivMaskChangeListeners.contains(listener))
    ivMaskChangeListeners.removeElement(listener);
}

/**
 * Return an enumeration over the set of listeners currently registered
 * to be informed of changes in the message or trace mask configuration.
 * <p>
 * @return An Enumeration of mask change listeners. If no listeners
 * are registered, the Enumeration is empty.
 */
public Enumeration getMaskChangeListeners()  {
  return ivMaskChangeListeners.elements();
}

/**
 * Inform all registered <code>RASIMaskChangeListener</code>s
 8 that the message or trace mask has been changed.
 *<p>
 * @param mc A mask change event, indicating what has changed.
 */
public void fireMaskChangedEvent(RASMaskChangeEvent mc)  {
  RASIMaskChangeListener c;
  Enumeration e = getMaskChangeListeners();
  while (e.hasMoreElements()) {
    c = (RASIMaskChangeListener) e.nextElement();
    c.maskValueChanged(mc);
  }
}
```

```
/**
 * Add the name of a message event class to the list of message
 * event classes which this  handler processes. If the specified
 * event class is null or is already  registered, this method
 * does nothing.
 * <p>
 * @param name The event class name.
 */
public void addMessageEventClass(String name)  {
  if (name != null && (! ivMessageEventClasses.contains(name)))
    ivMessageEventClasses.addElement(name);
}

/**
 * Remove the name of a message event class from the list of names of
 * classes which this handler processes. If the specified event
 * class is null or is not registered, this method does nothing.
 * <p>
 * @param name The event class name.
 */
public void removeMessageEventClass(String name)  {
  if ((name != null) && (ivMessageEventClasses.contains(name)))
    ivMessageEventClasses.removeElement(name);
}

/**
 * Return an enumeration over the set of names of the message event
 * classes which this handler processes.
 * <p>
 * @return An Enumeration of RAS event class names.  If no event classes
 * are registered, the Enumeration is empty.
 */
public Enumeration getMessageEventClasses()  {
  return ivMessageEventClasses.elements();
}

/**
 * Add the name of a trace event class to the list of trace event classes
 8 which this handler processes.  If the specified event
 * class is null or is already  registered, this method does nothing.
 * <p>
 * @param name The event class name.
 */
public void addTraceEventClass(String name)  {
  if ((name != null) && (!ivTraceEventClasses.contains(name)))
    ivTraceEventClasses.addElement(name);
}

/**
 * Remove the name of a trace event class from the list of names of
 * classes which  this handler processes. If the
 * specified event class is null or is not registered, this method
 * does nothing.
 * <p>
 * @param name The event class name.
 */
public void removeTraceEventClass(String name)  {
  if ((name != null) && (ivTraceEventClasses.contains(name)))
    ivTraceEventClasses.removeElement(name);
}

/**
 * Return an enumeration over the set of names of the trace event
 * classes which this handler processes
 * <p>
 * @return An Enumeration of RAS event class names.  If no event classes
 * are registered, the Enumeration is empty.
```

```
 */
public Enumeration getTraceEventClasses()  {
  return ivTraceEventClasses.elements();
}

////////////////////////////////////////////////////////
//
// Methods required by the RASIHandler interface
//
////////////////////////////////////////////////////////

/**
 * Return the maximum number of {@link RASIEvent RASIEvents} which
 * this handler will queue before writing.
 * <p>
 * In the WebSphere Application Server environment, handlers may
 * not start threads. All writes will be done
 * synchonously and never queued. This handler does not queue events
 * for later retry if a write operation fails.
 * <p>
 * @return zero is always returned.
 */
public int getMaximumQueueSize()  {
  return 0;
}

/**
 * Set the maximum number of {@link RASIEvent RASIEvents} which the
 * handler will queue before writing.
 * <p>
 * This handler does not queue events. This method is a no-operation
 * <p>
 * @param size The maximum queue size. Input is ignored.
 */
public void setMaximumQueueSize(int size) throws IllegalStateException  {
  return;
}

/**
 * Return the amount of time (in milliseconds) that this handler
 * will wait before retrying a failed write
 * <p>
 * This handler does not retry or queue failed writes. If a write
 * operation fails, the event is simply discarded.
 * <p>
 * @return The retry interval. Zero is always returned.
 */
public int getRetryInterval()  {
  return 0;
}

/**
 * Set the amount of time (in milliseconds) that this handler will
 * wait before retrying a failed write.
 * <p>
 * This handler does not queue or retry failed writes. This method
 * is a no-operation.
 * <p>
 * @param interval The retry interval. Input is ignored.
 */
public void setRetryInterval(int interval)  {
  return;
}

/**
 * Return the current number of {@link RASIEvent RASIEvents} in
 * the handler's queue.
```

```
 * <p>
 * This handler does not queue events. Zero is always returned.
 * <p>
 * @return The current queue size. Zero is always returned.
 */
public int getQueueSize()  {
  return 0;
}

/**
 * Add a RASIFormatter to the set of formatters which are
 * currently registered to  this handler. The specified formatter
 * must be fully configured. Specifically, the formatter must be
 * configured with the set of {@link RASIEvent} classes which it
 * knows how to format.
 * <p>
 * @param formatter The event formatter. Null is tolerated. If the
 * specified formatter  supports formatting an event class which
 * already has an associated formatter, the existing formatter
 * is replaced with this one.
 **/
public void addFormatter(RASIFormatter formatter)  {
  if (formatter != null) {
    Enumeration e = formatter.getEventClasses();
    while (e.hasMoreElements()) {
      String name = (String) e.nextElement();
      ivFormatters.put(name, formatter);
    }
  }
}

/**
 * Remove a RASIFormatter from the set of formatters currently
 * registered with this handler.
 * <p>
 * @param formatter The event formatter. If the specified formatter
 * is null or is not registered, this method does nothing.
 */
public void removeFormatter(RASIFormatter formatter)  {
  if (formatter != null) {
    Enumeration e = formatter.getEventClasses();
    while (e.hasMoreElements())  {
      String name = (String) e.nextElement();
      ivFormatters.remove(name);
    }
  }
}

/**
 * Return an enumeration over the set of RASIFormatters currently
 * registered with this handler.
 * <p>
 * @return An Enumeration over the set of registered formatters.
 * If no formatters are
 * currently registered, the Enumeration is empty.
 **/
public Enumeration getFormatters()  {
  return ivFormatters.elements();
}

/**
 * Close the stream to which this handler is currently writing
 * its entries, if the stream is currently open.
 */
public void closeDevice()  {
  synchronized(ivDeviceLock) {
    if (ivWriter == null)
```

```
      return;
    ivWriter.flush();
    ivWriter.close();
    ivWriter = null;
  }
}

/**
 * Stop the handler, closing the stream to which this handler
 * is currently writing its entries
 * <p>
 * This method must be called when a handler is no longer needed.
 * Be careful not to call
 * this method if other loggers may still be using this handler.
 */
public void stop()  {
  // This handler does not have any queues to flush or preprocessing
  // to do. Simply call closeDevice().
  closeDevice();
}

/**
 * Asynchronously process a RAS event passed from a logger to this handler.
 * <p>
 * WebSphere Application Server loggers always operate synchronously.
 * It is expected that no events will be delivered via this method. This
 * handler also only supports synchronous operations. If events are
 * delivered via this method, simply process them synchronously
 * <p>
 * @param event A RAS event. Null is tolerated
 */
public void logEvent(RASIEvent event)  {
  writeEvent(event);
}

/**
 * Synchronously process a RAS event passed from a logger to this handler.
 * <p>
 * WebSphere Application Server loggers always operate synchronously.
 * It is expected that all
 * events will be delivered via this method.  This handler also only
 * supports synchronous operations.
 * <p>
 * @param event A RAS event. Null is tolerated
 */
public void writeEvent(RASIEvent event)  {
  if (event == null)
    return;
  synchronized(ivDeviceLock) {
    if (ivWriter == null)
      return;
    RASIFormatter formatter = findFormatter(event);
    if (formatter != null) {
      String msg = formatter.format(event);
      ivWriter.println(msg);
      // If an error occurs, simply set the boolean that caller can check
      if (ivWriter.checkError())
        errorHasOccurred = true;
    }
  }
}

////////////////////////////////////////////////////////
//
// Methods introduced by this implementation
//
////////////////////////////////////////////////////////
```

```
/**
 * Return the fully-qualified, normalized name of the file which
 * this handler is currently configured to write events to.
 * <p>
 * @return The fully-qualified, normalized name of the output file.
 */
public String getFileName()  {
  return ivFqFileName;
}

/**
 * Set this handler to write to a file other than the file it is
 * currently writing to. <p>
 * The current stream that the handler is writing to is closed. A
 * new stream is opened over the specified file.
 * <p>
 * @param name name of the file. May not be null. A fully-qualified
 * file name is recommended.
 * @exception An exception is thrown if the specified name is null,
 * the file cannot be created or some other error
 * occurs. If an exception is thrown, the handlers state is indeterminate.
 */
  public void setFileName(String name) throws Exception  {
  if (name == null)
    throw new Exception("Null passed for name");
  synchronized(ivDeviceLock) {
    closeDevice();
    File x = new File(name);
    ivFqFileName = x.getAbsolutePath();
    openDevice();
  }
}

/**
 * Open a stream over the file to which this handler will write
 * formatted log entries.
 * The stream will always be opened in append mode.
 * <p>
 * If a stream is already open over the file, the current stream
 * is closed.
 * If an error occurs during this operation, the errorHasOccurred
 * boolean is set to true
 * and a plain text error message is written to System.err along
 * with the exception
 * stack trace, if any. If the operation is successful, the
 * errorHasOccurred boolean is
 * set to false.
 * <p>
 */
public void openDevice()  {
  synchronized(ivDeviceLock) {
    try  {
      closeDevice();
      errorHasOccurred = false;
      // The file name may have been changed.Create the
      // directory for the file if it doesn't already exist.
      File x = new File(ivFqFileName);
      String dir = x.getParent();
      File dirs = new File(dir);
      if (fileExists(dirs) == false) {
        boolean result = makeDirectories(dirs);
        if (result == false) {
          errorHasOccurred = true;
          return;
        }
      }
```

```
      // Open a file output stream over the file in append mode.
      // Wrap the FileOutputStream in an OutputStreamWriter. Finally wrap
      // the OutputStreamWriter in a BufferedPrintWriter with line flushing enabled.
      FileOutputStream fos = createFileOutputStream(ivFqFileName, true);
      OutputStreamWriter osw = new OutputStreamWriter(fos);
      ivWriter = new PrintWriter(new BufferedWriter(osw), true);
    }
    catch (Throwable t) {
      // not much we can do here except set the error boolean.
      errorHasOccurred = true;
      System.err.println("Error occurred in openDevice() for handler "+ivName);
      t.printStackTrace();
    }
  }
}

/**
 * Return a reference to the formatter associated with the
 * specified event class. If the specified event class is not
 * registered, the superclasses of the event class will be
 * checked for a registered formatter.
 * <p>
 * @param event A RAS event. Must not be null.
 * @return formatter The formatter associated with the specified
 * event class. Null is returned if the event class is not registered.
 */
private RASIFormatter findFormatter(RASIEvent event)  {
  Class eventClass = event.getClass();
  RASIFormatter formatter = null;

  while (eventClass != null) {
    String className = eventClass.getName();
    if (ivFormatters.containsKey(className)) {
      return (RASIFormatter) ivFormatters.get(className);
    }
    else
      eventClass = eventClass.getSuperclass();
  }
  return null;
}

/**
 * A worker method that wraps the creation of a FileOutputStream
 * in a doPrivileged block.
 * <p>
 * @param fileName the name of the file to create the stream over.
 * @param append a boolean, when true indicates the file should
 * be opened in append
 *  mode
 * @ return the FileOutputStream.
 * @exception SecurityException A security violation has occurred.
 * This class is not authorized
 *  to access the specified file.
 * @exception PrivilegedActionException a checked exception was
 * thrown in the course of
 *  running the privileged action. The checked exception is contained
 * within the
 *  PrivilegedActionException. Most likely the wrapped exception
 * is a FileNotFound.
 */
private FileOutputStream createFileOutputStream(String fileName, boolean append)
      throws PrivilegedActionException
{
  final String tempFileName = fileName;
  final boolean tempAppend = append;
  FileOutputStream fs = (FileOutputStream) AccessController.doPrivileged(
    new PrivilegedExceptionAction() {
```

```
      public Object run() throws IOException {
        return new FileOutputStream(tempFileName, tempAppend);
      }
    }
  );
  return fs;
}

/**
 * A worker method that wraps the check for the existence of a file in a
 * doPrivileged block.
 * <p>
 * @param fileToCheck a <code>File</code>
 * object whose abstract pathname corresponds
 *  to the physical file whose existence is to be checked.
 * @return true if and only if the file exists. Otherwise false.
 * @exception SecurityException A security violation has occurred.
 * This class is not authorized
 *  to access the specified file.
 */
private boolean fileExists(File fileToCheck) throws SecurityException
{
  final File tempFileToCheck = fileToCheck;
  Boolean exists = (Boolean) AccessController.doPrivileged(
    new PrivilegedAction() {
      public Object run() {
        return new Boolean(tempFileToCheck.exists());
      }
    }
  );
  return exists.booleanValue();
}

/**
 * A worker method that wraps the creation of directories in a
 * doPrivileged block.
 * <p>
 * @param dirToMake a non-null <code>File</code>
 * object whose abstract pathname represents
 *  the fully qualified directory to create.
 * @return true is returned if and only if all necessary
 * directories were created. Otherwise
 *  false is returned.
 * @exception SecurityException A security violation has occurred.
 *  This class is not authorized
 *  to access at leas one of the specified directories.
 */
private boolean makeDirectories(File dirToMake) throws SecurityException
{
  final File tempDirToMake = dirToMake;
  Boolean result = (Boolean) AccessController.doPrivileged(
    new PrivilegedAction() {
      public Object run() {
        return new Boolean(tempDirToMake.mkdirs());
      }
    }
  );
  return result.booleanValue();
}

}
```

Example: user written formatter

The following is a very simple sample of a Formatter class. This class is functional, but is intended solely to demonstrate concepts. For simplicity and clarity, much

code (including appropriate boundary condition checking logic) has been ignored. This sample is not intended to be an example of good programming practice.

```
package com.ibm.ws.ras.test.user;
import com.ibm.ras.*;
import java.text.*;
import java.util.*;

/**
 * The <code>SimpleFormatter</code> implements
 * the RASIFormatter interface. It is a
 * simple implementation used for demonstration purposes only.
 * It does not do any
 * advanced formatting, it simply formats the message and
 * parameters in an event.
 * It does not include the timestamp in the formatted result, for example.
 */
public class SimpleFormatter implements RASIFormatter
{

  /**
   * The name of the formatter
   */
  private String ivName = "";

  /**
   * A vector containing the event classes this Formatter
   * knows how to process.
   */
  private Vector ivEventClasses = new Vector();

  /**
   * Create a <code>SimpleFormatter</code>.
   */
  public SimpleFormatter(String name)  {
    setName(name);
  }

  ////////////////////////////////////////////////////////
  //
  // Methods required by the RASIObject Interface
  //
  ////////////////////////////////////////////////////////

  /**
   * Return this objects configuration as a set of Properties in a Hashtable.
   * <p>
   * This formatter does not support properties-based configuration.
   * Therefore a call to this method always returns null
   * @return null is always returned.
   */
  public Hashtable getConfig()  {
    return null;
  }

  /**
   * Set this objects configuration from the properties in the
   * specified Hashtable.
   * <p>
   * This formatter does not support properties-based configuration.
   * This method is a no-operation.
   * @param hashTable a Hashtable containing the properties.
   * Input is ignored.
   */
  public void setConfig(Hashtable ht)  {
    return;
  }
```

```
/**
 * Return the name by which this formatter is known.
 * <p>
 * @return a String containing the name of this object, or an empty
 * string ("") if the name has not been set.
 */
public String getName()   {
  return ivName;
}

/**
 * Set the name by which this formatter is known.  If the specified name
 * is  <code>null</code>, the current name is not changed.
 * <p>
 * @param name The new name for this object. Null is tolerated.
 */
public void setName(String name)   {
  if (name != null)
    ivName = name;
}

/**
 * Return the description field of this formatter.
 *<p>
 * This formatter does not use a description field. An empty String is
 * always returned.<p>
 * @return an empty String.
 */
public String getDescription()  {
  return "";
}

/**
 * Set the description field for this formatter.
 *<p>
 * This formatter does not use a description field. Input is ignored
 * and this method does nothing.
 *<p>
 * @param desc The description of this object. Input is ignored.
 */
public void setDescription(String desc)  {
  return;
}

/**
 * Return the name of the {@link com.ibm.ras.mgr.RASManager RASManager}
 * group with which this formatter is associated.  This method is only
 * used by the  RAS Manager.
 *<p>
 * This formatter does not support RASManager configuration.
 * Null is always returned.
 * @return null is always returned.
 */
public String getGroup()  {
  return null;
}

////////////////////////////////////////////////////
//
// Methods required by the RASIFormatter Interface
//
////////////////////////////////////////////////////

/**
 * Set a flag that indicates whether this formatter is the
 * default formatter used by
```

```
 * {@link com.ibm.ras.RASHandler} objects to format events.
 *<p>
 * Instances of com.ibm.ras.RASHandler are not allowed to be
 * instantiated in the WebSphere Application Server environment.
 * This formatter cannot be the default formatter for handlers of
 * this type. This method does nothing.
 *<p>
 * @param flag input is ignored, since this formatter cannot be the
 * default formatter.
 */
public void setDefault(boolean flag)  {
  return;
}

/**
 * Return a boolean that indicates whether or not this is the
 * default formatter
 * used by a {@link com.ibm.ras.RASHandler} to format the RAS events.
 *<p>
 * com.ibm.ras.RASHandlers will never be instantiated in a
 * WebSphere Application Server environment so this method always returns false.
 *<p>
 * @return false is always returned.
 */
public boolean isDefault()  {
  return false;
}

/**
 * Add the name of a {@link com.ibm.ras.RASIEvent} class to the
 * list of classes which this formatter can process.
 * If the specified class name is null or it is already registered,
 * this method does nothing.
 *<p>
 * @param name The event class name. Null is tolerated.
 */
public void addEventClass(String name)  {
  if ((name != null) && (! ivEventClasses.contains(name)))
    ivEventClasses.addElement(name);
}

/**
 * Remove the name of a {@link com.ibm.ras.RASIEvent} class
 * from the list of classes which this formatter
 * can process. If the specified class name is null or is not
 * registered, this method does nothing.
 *<p>
 * @param name The event class name.
 */
public void removeEventClass(String name)  {
  if ((name != null) && (ivEventClasses.contains(name)))
    ivEventClasses.removeElement(name);
}

/**
 * Return an enumeration over the set of names of
 * {@link com.ibm.ras.RASIEvent} classes which this formatter can process.
 *<p>
 * @return An enumeration of RAS event class names. If no event classes
 *  are registered, the enumeration is empty.
 */
public Enumeration getEventClasses() {
  return ivEventClasses.elements();
}

/**
 * Format the specified {link com.ibm.ras.RASIEvent} object and return
```

```
 * a String containing the formatted output.
 *<p>
 * @param event The event to format. Null is tolerated.
 * @return The formatted event contents. Null may be returned.
 */
public String format(RASIEvent event)  {
  if (event == null)
    return null;
  if (event.isMessageEvent())
    return formatMessage(event);
  else
    return formatTrace(event);
}

/**
 * Format a message event.
 *<p>
 * If a message key is used and that key is not found in any message
 * file, the message text becomes an error message
 * indicating that the key was not found.
 *<p>
 * @param event The event to format.
 * @return The formatted event.
 */
public String formatMessage(RASIEvent event)  {
  String messageKey = "null";
  try {
    messageKey = event.getText();
    String[] messageInserts = event.getParameters();
    if (event instanceof com.ibm.ras.RASMessageEvent) {
      // RASMessageEvents usually contain localizable messages.
      RASMessageEvent rme = (RASMessageEvent)event;
      String bundleName = rme.getMessageFile();
      if (bundleName != null) {
        // Not a text message, get localized message and return
        ResourceBundle bundle
                = ResourceBundle.getBundle(bundleName, Locale.getDefault());
        String localizedKey =  bundle.getString(messageKey);
        return MessageFormat.format(localizedKey, messageInserts);
      }
      else {
        // Text message
        for (int i=0; i<messageInserts.length; ++i)
               {messageKey = messageKey + " " + messageInserts[i];
        }
        return messageKey;
      }
    }
    else {
      // A User defined type. Append paramaters to key and return
      for (int i=0; i<messageInserts.length; ++i){
        messageKey = messageKey + " " + messageInserts[i];
      }
      return messageKey;
    }
  }
  catch (Throwable t) {
    t.printStackTrace();
    return "SimpleFormattter: Error while formatting message "+messageKey;
  }
}

 /**
 * Format a trace event.
 *<p>
 * Append the parameters (in order of specification) to the
 * text message in the trace event object.
```

```
     *<p>
     * @param event The event to format.
     * @return The formatted event.
     */
    private String formatTrace(RASIEvent event)  {
      String text = "null";
      try {
        text = event.getText();
        String[] parms = event.getParameters();
        if (parms != null) {
          for (int i=0; i<parms.length; ++i){
           text = text + " " + parms[i];
          }
        }
        return text;
      }
      catch (Throwable t) {
        t.printStackTrace();
        return "SimpleFormatter: Error while formatting trace "+text;
      }
    }

}
```

Programming model summary

The programming model described in this section builds upon and summarizes some of the concepts already introduced. This section also formalizes usage requirements and restrictions. Use of the WebSphere Application Server JRas extensions in a manner that does not conform to the following programming guidelines is prohibited.

As described previously, you can use the WebSphere Application Server JRas extensions in three distinct operational modes. The programming models concepts and restrictions apply equally across all modes of operation.

- You must not use implementation classes provided by the stand-alone JRas logging toolkit directly, unless specifically noted otherwise. Direct usage of those classes is not supported. IBM Support will provide no diagnostic aid or bug fixes relating to direct usage of classes provided by the stand-alone JRas logging toolkit.
- You must obtain message and trace loggers directly from the Manager class. You cannot directly instantiate loggers.
- There is no provision that allows you to replace the WebSphere Application Server message and trace logger classes.
- You must guarantee that the logger names passed to the Manager are unique, and follow the naming constraints documented below. Once a logger is obtained from the Manager, you must not attempt to change the name of the logger by calling the `setName()` method.
- Named loggers are idempotent. For any given name, the first call to the Manager results in the Manager creating a logger that is associated with that name. Subsequent calls to the Manager that specify the same name result in a reference to the existing logger being returned.
- The Manager maintains a hierarchical namespace for loggers. It is recommended but not required that a dot-separated, fully qualified class name be used to identify any given logger. Other than dots or periods, logger names cannot contain any punctuation characters, such as asterisk (*), comma(.), equals sign(=), colon(:), or quotes.
- Group names must comply with the same naming restrictions as logger names.

- The loggers returned from the Manager are subclasses of the RASMessageLogger and RASTraceLogger provided by the stand-alone JRas logging toolkit. You are allowed to call any public method defined by the RASMessageLogger and RASTraceLogger classes. You are not allowed to call any public method introduced by the provided subclasses.
- If you want to operate in either stand-alone or combined mode, you must provide your own Handler and Formatter subclasses. You are not allowed to use the Handler and Formatter classes provided by the stand-alone JRas logging toolkit. User written Handlers and Formatters must conform to the documented guidelines.
- Loggers obtained from the Manager come with a WebSphere Application Server handler installed. This handler will write message and trace records to logs defined by the WebSphere Application Server runtime. Manage these logs using the provided systems management interfaces.
- You can programmatically add and remove user-defined Handlers from a logger at any time. Multiple additions and removals of user defined handlers are allowed. You are responsible for creating an instance of the handler to add, configuring the handler by setting the handler's mask value and formatter appropriately, then adding the handler to the logger using the `addHandler()` method. You are responsible for programmatically updating the masks of user-defined handlers as appropriate.
- You may get a reference to the handler installed within a logger by calling the `getHandlers()` method on the logger and processing the results. You must not call any methods on the handler obtained in this fashion. You are allowed to remove the WebSphere Application Server handler from the logger by calling the logger's `removeHandler()` method, passing in the reference to the WebSphere Application Server handler. Once removed, the WebSphere Application Server handler cannot be re-added to the logger.
- You are allowed to define your own message type. The behavior of user-defined message types and restrictions on their definitions is discussed in Extending the JRas framework.
- You are allowed to define your own message event classes.á The usage of user-defined message event classes is discussed in Extending the JRas framework.
- You are allowed to define your own trace types. The behavior of user-defined trace types and restrictions on your definitions is discussed in Extending the JRas framework.
- You are allowed to define your own trace event classes. The usage of user-defined trace event classes is discussed in Extending the JRas framework.
- You must programmatically maintain the bits in the message and trace logger masks that correspond to any user-defined types. If WebSphere Application Server facilities are being used to manage the predefined types, these updates must not modify the state of any of the bits corresponding to those types. If you are assuming ownership responsibility for the predefined types then you can change all bits of the masks.

JRas Messages and Trace event types

This section describes JRas message and trace event types.

Event types

The base message and trace event types defined by the stand-alone JRas logging toolkit are not the same as the ″native″ types recognized by the WebSphere

Application Server runtime. Instead the basic JRas types are mapped onto the native types. This mapping may vary by platform or edition. The mapping is discussed below.

Platform Message Event Types

The message event types that are recognized and processed by the WebSphere Application Server runtime are defined in the RASIMessageEvent interface provided by the stand-alone JRas logging toolkit. These message types are mapped onto the native message types as follows.

WebSphere Application Server native type	JRas RASIMessageEvent type
`Audit`	TYPE_INFO, TYPE_INFORMATION
`Warning`	TYPE_WARN, TYPE_WARNING
`Error`	TYPE_ERR, TYPE_ERROR

Platform Trace Event Types

The trace event types recognized and processed by the WebSphere Application Server runtime are defined in the `RASITraceEvent` interface provided by the stand-alone JRas logging toolkit. The `RASITraceEvent` interface provides a rich and overly complex set of types. This interface defines both a simple set of levels, as well as a set of enumerated types.

- For a user who prefers a simple set of levels, `RASITraceEvent` provides `TYPE_LEVEL1`, `TYPE_LEVEL2`, and `TYPE_LEVEL3`. The implementations provide support for this set of levels. The levels are hierarchical (that is, enabling level 2 will also enable level 1, enabling level 3 also enables levels 1 and 2).
- For users who prefer a more complex set of values that can be *OR'd* together, `RASITraceEvent` provides `TYPE_API`, `TYPE_CALLBACK`, `TYPE_ENTRY_EXIT`, `TYPE_ERROR_EXC`, `TYPE_MISC_DATA`, `TYPE_OBJ_CREATE`, `TYPE_OBJ_DELETE`, `TYPE_PRIVATE`, `TYPE_PUBLIC`, `TYPE_STATIC`, and `TYPE_SVC`.

The trace event types are mapped onto the native trace types as follows:

Mapping WebSphere Application Server trace types to JRas `RASITraceEvent` "Level" types.

WebSphere Application Server native type	JRas RASITraceEvent level type
`Event`	TYPE_LEVEL1
`EntryExit`	TYPE_LEVEL2
`Debug`	TYPE_LEVEL3

Mapping WebSphere Application Server trace types to JRas `RASITraceEvent` enumerated types.

WebSphere Application Server native type	JRas RASITraceEvent enumerated types
`Event`	TYPE_ERROR_EXC, TYPE_SVC, TYPE_OBJ_CREATE, TYPE_OBJ_DELETE
`EntryExit`	TYPE_ENTRY_EXIT, TYPE_API, TYPE_CALLBACK, TYPE_PRIVATE, TYPE_PUBLIC, TYPE_STATIC
`Debug`	TYPE_MISC_DATA

For simplicity, it is recommended that one or the other of the tracing type methodologies is used consistently throughout the application. For users who decide to use the non-level types, it is further recommended that you choose one type from each category and use those consistently throughout the application to avoid confusion.

Message and Trace parameters

The various message logging and trace method signatures accept parameter types of `Object`, `Object[]` and `Throwable`. WebSphere Application Server will process and format the various parameter types as follows.

Primitives
: Primitives, such as `int` and `long` are not recognized as subclasses of `Object` and cannot be directly passed to one of these methods. A primitive value must be transformed to a proper `Object` type (`Integer`, `Long`) before being passed as a parameter.

`Object`
: `toString()` is called on the object and the resulting `String` is displayed. The `toString()` method should be implemented appropriately for any object passed to a message logging or trace method. It is the responsibility of the caller to guarantee that the `toString()` method does not display confidential data such as passwords in clear text, and does not cause infinite recursion.

`Object[]`
: The `Object[]` is provided for the case when more than one parameter is passed to a message logging or trace method. `toString()` is called on each `Object` in the array. Nested arrays are not handled. (i.e. none of the elements in the `Object` array should be an array).

`Throwable`
: The stack trace of the `Throwable` is retrieved and displayed.

Array of Primitives
: An array of primitive (e.g. `byte[]`, `int[]` is recognized as an `Object`, but is treated somewhat as a second cousin of `Object` by Java. In general, arrays of primitives should be avoided, if possible. If arrays of primitives are passed, the results are indeterminate and may change depending on the type of array passed, the API used to pass the array and the release of the product. For consistent results, user code should preprocess and format the primitive array into some type of String form before passing it to the method. If such preprocessing is not performed, the following may result.

  - [B@924586a0b - This is deciphered as ″a byte array at location X″. This is typically returned when an array is passed as a member of an `Object[]`. It is the result of calling `toString()` on the `byte[]`.
  - Illegal trace argument : array of long. This is typically returned when an array of primitives is passed to a method taking an `Object`.
  - 01040703... : the hex representation of an array of bytes. Typically this may be seen when a byte array is passed to a method taking a single `Object`. This behavior is subject to change and should not be relied on.
  - ″1″ ″2″ ... : The String representation of the members of an `int[]` formed by converting each element to an Integer and calling `toString` on the Integers. This behavior is subject to change and should not be relied on.
  - [Ljava.lang.Object;@9136fa0b : An array of objects. Typically this is seen when an array containing nested arrays is passed.

Controlling message logging

Writing a message to a WebSphere Application Server log requires that the message type passes three levels of filtering or screening.

1. The message event type must be one of the message event types defined in the `RASIMessageEvent` interface.
2. Logging of that message event type must be enabled by the state of the message logger's mask.
3. The message event type must pass any filtering criteria established by the WebSphere Application Server runtime itself.

When a WebSphere Application Server logger is obtained from the Manager, the initial setting of the mask is to forward all native message event types to the WebSphere Application Server handler. It is possible to control what messages get logged by programmatically setting the state of the message logger's mask.

Some editions of the product allow the user to specify a message filter level for a server process. When such a filter level is set, only messages at the specified severity levels are written to WebSphere Application Server logs. This means that messages types that pass the message logger's mask check may be filtered out by the WebSphere Application Server itself.

Controlling Tracing

Each edition of the product provides a mechanism for enabling or disabling trace. The various editions may support static trace enablement (trace settings are specified before the server is started), dynamic trace enablement (trace settings for a running server process can be dynamically modified) or both.

Writing a trace record to a WebSphere Application Server requires that the trace type passes three levels of filtering or screening.

1. The trace event type must be one of the trace event types defined in the `RASITraceEvent` interface.
2. Logging of that trace event type must be enabled by the state of the trace logger's mask.
3. The trace event type must pass any filtering criteria established by the WebSphere Application Server runtime itself.

When a logger is obtained from the Manager, the initial setting of the mask is to suppress all trace types. The exception to this rule is the case where the WebSphere Application Server runtime supports static trace enablement and a non-default startup trace state for that trace logger has been specified. Unlike message loggers, the WebSphere Application Server may dynamically modify the state of a trace loggers trace mask. WebSphere Application Server will only modify the portion of the trace logger's mask corresponding to the values defined in the `RASITraceEvent` interface. WebSphere Application Server will not modify undefined bits of the mask that may be in use for user defined types.

When the dynamic trace enablement feature available on some platforms is used, the trace state change is reflected both in the Application Server runtime and the trace loggers trace mask. If user code programmatically changes the bits in the trace mask corresponding to the values defined by in the `RASITraceEvent` interface, the trace logger's mask state and the runtime state will become unsynchronized

and unexpected results will occur. Therefore, programmatically changing the bits of the mask corresponding to the values defined in the `RASITraceEvent` interface is not allowed.

Instrumenting an application with JRas extensions

To instrument an application using the WebSphere Application Server JRas extensions, perform the following steps:

Steps for this task

1. Determine the mode the extensions will be used in: integrated, stand-alone or combined.
2. If the extensions will be used in either stand-alone or combined mode, create the necessary handler and formatter classes.
3. If localized messages will be used by the application, create a resource bundle as described in Creating JRas resource bundles and message files.
4. In the application code, get a reference to the Manager class and create the manager and logger instances as described in Creating JRas manager and logger instances.
5. Insert the appropriate message and trace logging statements in the application as described in Creating JRas manager and logger instances.

Creating JRas resource bundles and message files

The WebSphere Application Server message logger provides the `message()` and `msg()` methods to allow the user to log localized messages. In addition, it provides the `textMessage()` method for logging of messages that are not localized. Applications can use either or both, as appropriate.

The mechanism for providing localized messages is the Resource Bundle support provided by the Java Development Kit (JDK). If you are not familiar with resource bundles as implemented by the JDK, you can get more information from various texts, or by reading the javadoc for the `java.util.ResourceBundle`, `java.util.ListResourceBundle` and `java.util.PropertyResourceBundle` classes, as well as the `java.text.MessageFormat` class.

The `PropertyResourceBundle` is the preferred mechanism to use. In addition, note that the JRas extensions do not support the extended formatting options such as {1, date} or {0,number, integer} that are provided by the `MessageFormat` class.

You can forward messages that are written to the internal WebSphere Application Server logs to other processes for display. For example, messages displayed on the administrator console, which can be running in a different location than the server process, can be localized using the *late binding* process. Late binding means that WebSphere Application Server does not localize messages when they are logged, but defers localization to the process that displays the message.

To properly localize the message, the displaying process must have access to the resource bundle where the message text is stored. This means that you must package the resource bundle separately from the application, and install it in a location where the viewing process can access it. If you do not want to take these steps, you can use the early binding technique to localize messages as they are logged.

The two techniques are described as follows:

Early binding

The application must localize the message before logging it. The application looks up the localized text in the resource bundle and formats the message. When formatting is complete, the application logs the message using the `textMessage()` method. Use this technique to package the application's resource bundles with the application.

Late binding

The application can choose to have the WebSphere Application Server runtime localize the message in the process where it is displayed. Using this technique,the resource bundles are packaged in a stand-alone `.jar` file, separately from the application. You must then install the resource bundle `.jar` file on every machine in the installation from which an administrator's console or log viewing program might be run. You must install the `.jar` file in a directory that is part of the extensions classpath. In addition, if you forward logs to IBM service, you must also forward the `.jar` file containing the resource bundles.

To create a resource bundle, perform the following steps.

Steps for this task

1. Create a text properties file that lists message keys and the corresponding messages.

   The properties file must have the following characteristics:

   - Each property in the file is terminated with a line-termination character.
   - If a line contains only white space, or if the first non-white space character of the line is the symbol # (pound sign) or ! (exclamation mark), the line is ignored. The # and ! characters can therefore be used to put comments into the file.
   - Each line in the file, unless it is a comment or consists only of white space, denotes a single property. A backslash (\) is treated as the line-continuation character.
   - The syntax for a property file consists of a key, a separator, and an element. Valid separators include the equal sign (=), colon (:), and white space ( ).
   - The key consists of all characters on the line from the first non-white space character to the first separator. Separator characters can be included in the key by escaping them with a backslash (\), but doing this is not recommended, because escaping characters is error prone and confusing. It is instead recommended that you use a valid separator character that does not appear in any keys in the properties file.
   - White space after the key and separator is ignored until the first non-white space character is encountered. All characters remaining before the line-termination character define the element.

   See the Java documentation for the `java.util.Properties` class for a full description of the syntax and construction of properties files.
2. The file can then be translated into localized versions of the file with language-specific file names (for example, a file named `DefaultMessages.properties` can be translated into DefaultMessages_de.properties for German and DefaultMessages_ja.properties for Japanese).
3. When the translated resource bundles are available, write them to a system-managed persistent storage medium.

Resource bundles are then used to convert the messages into the requested national language and locale.

4. When a message logger is obtained from the JRas manager, it can be configured to use a particular resource bundle. Messages logged via the `message()` API will use this resource bundle when message localization is performed.

   At run time, the user's locale setting is used to determine the properties file from which to extract the message specified by a message key, thus ensuring that the message is delivered in the correct language.

5. **(Optional)** If the message loggers `msg()` method is called, a resource bundle name must be explicitly provided.

What to do next

The application locates the resource bundle based on the file's location relative to any directory in the classpath. For instance, if the property resource bundle named DefaultMessages.properties is located in the `baseDir/subDir1/subDir2/resources` directory and `baseDir` is in the class path, the name `subdir1.subdir2.resources.DefaultMessage` is passed to the message logger to identify the resource bundle.

Developing JRas resource bundles

Resource bundle sample

You can create resource bundles in several ways. The best and easiest way is to create a properties file that supports a `PropertiesResourceBundle`. This sample shows how to create such a properties file.

For this sample, four localizable messages are provided. The properties file is created and the key-value pairs inserted into it. All the normal properties files conventions and rules apply to this file. In addition, the creator must be aware of other restrictions imposed on the values by the Java `MessageFormat` class. For example, apostrophes must be ″escaped″ or they will cause a problem. Also avoid use of non-portable characters. WebSphere Application Server does not support usage of extended formatting conventions that the `MessageFormat` class supports, such as {1, date} or {0,number, integer}.

Assume that the base directory for the application that uses this resource bundle is ″`baseDir`″ and that this directory will be in the classpath. Assume that the properties file is stored in a subdirectory of `baseDir` that is not in the classpath (e.g. `baseDir/subDir1/subDir2/resources`). In order to allow the messages file to be resolved, the name `subDir1.subDir2.resources.DefaultMessage` is used to identify the `PropertyResourceBundle` and is passed to the message logger.

For this sample, the properties file is named `DefaultMessages.properties`.

```
# Contents of DefaultMessages.properties file
MSG_KEY_00=A message with no substitution parameters.
MSG_KEY_01=A message with one substitution parameter: parm1={0}
MSG_KEY_02=A message with two substitution parameters: parm1={0}, parm2 = {1}
MSG_KEY_03=A message with three parameter: parm1={0}, parm2 = {1}, parm3={2}
```

Once the file `DefaultMessages.properties` is created, the file can be sent to a translation center where the localized versions will be generated.

Creating JRas manager and logger instances

You can use the JRas extensions in integrated, stand-alone, or combined mode. Configuration of the application will vary depending on the mode of operation, but usage of the loggers to log message or trace entries is identical in all modes of operation.

Integrated mode is the default mode of operation. In this mode, message and trace events are sent to the WebSphere Application Server logs. See Setting up for integrated JRas operation for information on configuring for this mode of operation.

In the combined mode, message and trace events are logged to both WebSphere Application Server and user-defined logs. See Setting up for combined JRas operation for more information on configuring for this mode of operation.

In the stand-alone mode, message and trace events are logged only to user-defined logs. See Setting up for stand-alone JRas operation for more information on configuring for this mode of operation.

Using the message and trace loggers

Regardless of the mode of operation, the use of message and trace loggers is the same. See Creating JRas resource bundles and message files for more information on using message and trace loggers.

Using a message logger

The message logger is configured to use the `DefaultMessages` resource bundle. Message keys must be passed to the message loggers if the loggers are using the `message()` API.

```
msgLogger.message(RASIMessageEvent.TYPE_WARNING, this, methodName, "MSG_KEY_00");
... msgLogger.message(RASIMessageEvent.TYPE_WARN, this, methodName, "MSG_KEY_01",
   "some string");
```

If message loggers use the `msg()` API, you can specify a new resource bundle name.

```
 msgLogger.msg(RASIMessageEvent.TYPE_ERR, this, methodName, "ALT_MSG_KEY_00",
"alternateMessageFile");
```

You can also log a text message. If you are using the `textMessage` API, no message formatting is done.

```
msgLogger.textMessage(RASIMessageEvent.TYPE_INFO, this, methodName, "String and Integer",
 "A String", new Integer(5));
```

Using a trace logger

Since trace is normally disabled, trace methods should be guarded for performance reasons.

```
private void methodX(int x, String y, Foo z)
{
   // trace an entry point. Use the guard to make sure tracing is enabled. Do this checking
before we waste cycles gathering parameters to be traced.
   if (trcLogger.isLoggable(RASITraceEvent.TYPE_ENTRY_EXIT) {
        // since I want to trace 3 parameters, package them up in an Object[]
        Object[] parms = {new Integer(x), y, z};
        trcLogger.entry(RASITraceEvent.TYPE_ENTRY_EXIT, this, "methodX", parms);
  }
```

```
... logic
  // a debug or verbose trace point
  if (trcLogger.isLoggable(RASITraceEvent.TYPE_MISC_DATA) {
        trcLogger.trace(RASITraceEvent.TYPE_MISC_DATA, this, "methodX" "reached here");
  }
  ...
  // Another classification of trace event. Here an important state change has been detected,
so a different trace type is used.
  if (trcLogger.isLoggable(RASITraceEvent.TYPE_SVC) {
     trcLogger.trace(RASITraceEvent.TYPE_SVC, this, "methodX", "an important event");
  }
 ...
  // ready to exit method, trace. No return value to trace
    if (trcLogger.isLoggable(RASITraceEvent.TYPE_ENTRY_EXIT)) {
        trcLogger.exit(RASITraceEvent.TYPE_ENTRY_EXIT, this, "methodX");
   }
}
```

Setting up for integrated JRas operation

In the integrated mode of operation, message and trace events are sent to WebSphere Application Server logs. This is the default mode of operation.

Steps for this task

1. Import the requisite JRas extensions classes

   ```
   import com.ibm.ras.*;
   import com.ibm.websphere.ras.*;
   ```

2. Declare logger references.

   ```
   private RASMessageLogger msgLogger = null;
   private RASTraceLogger trcLogger = null;
   ```

3. Obtain a reference to the Manager and create the loggers.

   Since loggers are named singletons, you can do this in a variety of places. One logical candidate for enterprise beans is the `ejbCreate()` method. For example, for the enterprise bean named ″myTestBean″, place the following code in the `ejbCreate()` method.

   ```
   com.ibm.websphere.ras.Manager mgr = com.ibm.websphere.ras.Manager.getManager();
   msgLogger = mgr.createRASMessageLogger("Acme", "WidgetCounter", "RasTest",
   myTestBean.class.getName());
   // Configure the message logger to use the message file created for this application.
   msgLogger.setMessageFile("acme.widgets.DefaultMessages");
   trcLogger = mgr.createRASTraceLogger("Acme", "Widgets", "RasTest",
   myTestBean.class.getName());
   mgr.addLoggerToGroup(trcLogger, groupName);
   ```

Setting up for combined JRas operation

In combined mode, messages and trace are logged to both WebSphere Application Server logs and user-defined logs. The following sample assumes that you have written a user defined handler named `SimpleFileHandler` and a user defined formatter named `SimpleFormatter`. It also assumes that you are not using user defined types or events.

Steps for this task

1. Import the requisite JRas extensions classes

   ```
   import com.ibm.ras.*;
   import com.ibm.websphere.ras.*;
   ```

2. Import the user handler and formatter.

   ```
   import com.ibm.ws.ras.test.user.*;
   ```

3. Declare the logger references.

```
private RASMessageLogger msgLogger = null;
  private RASTraceLogger trcLogger = null;
```

4. Obtain a reference to the Manager, create the loggers and add the user handlers.

   Since loggers are named singletons, you can obtain a reference to the loggers in a number of places. One logical candidate for enterprise beans is the `ejbCreate()` method. Make sure that multiple instances of the same user handler are not accidentally inserted into the same logger. Your initialization code must handle this. The following sample is a message logger sample. The procedure for a trace logger is similar.

```
com.ibm.websphere.ras.Manager mgr = com.ibm.websphere.ras.Manager.getManager();
    msgLogger = mgr.createRASMessageLogger("Acme", "WidgetCounter", "RasTest",
myTestBean.class.getName());
    // Configure the message logger to use the message file defined in the
    //ResourceBundle sample.
    msgLogger.setMessageFile("acme.widgets.DefaultMessages");

   // Create the user handler and formatter. Configure the formatter, then add
   //it to the handler.
   RASIHandler handler = new SimpleFileHandler("myHandler", "FileName");
   RASIFormatter formatter = new SimpleFormatter("simple formatter");
    formatter.addEventClass("com.ibm.ras.RASMessageEvent");
    handler.addFormatter(formatter);

   // Add the Handler to the logger. Add the logger to the list of the handlers
   //listeners, then set the handlers
   // mask, which will update the loggers composite mask appropriately.
   // WARNING - there is an order dependency here that must be followed.
   msgLogger.addHandler(handler);
   handler.addMaskChangeListener(msgLogger);
   handler.setMessageMask(RASIMessageEvent.DEFAULT_MESSAGE_MASK);
```

Setting up for stand-alone JRas operation

In stand-alone mode, messages and traces are logged only to user-defined logs. The following sample assumes that you have a user-defined handler named `SimpleFileHandler` and a user-defined formatter named `SimpleFormatter`. It is also assumes that no user-defined types or events are being used.

Steps for this task

1. Import the requisite JRas extensions classes

```
import com.ibm.ras.*;
import com.ibm.websphere.ras.*;
```

2. Import the user handler and formatter.

```
import com.ibm.ws.ras.test.user.*;
```

3. Declare the logger references.

```
private RASMessageLogger msgLogger = null;
  private RASTraceLogger trcLogger = null;
```

4. Obtain a reference to the Manager, create the loggers and add the user handlers.

   Since loggers are named singletons, you can obtain a reference to the loggers in a number of places. One logical candidate for enterprise beans is the `ejbCreate()` method. Make sure that multiple instances of the same user handler are not accidentally inserted into the same logger. Your initialization code must handle this. The following sample is a message logger sample. The procedure for a trace logger is similar.

```
com.ibm.websphere.ras.Manager mgr = com.ibm.websphere.ras.Manager.getManager();
    msgLogger = mgr.createRASMessageLogger("Acme", "WidgetCounter",
"RasTest", myTestBean.class.getName());
    // Configure the message logger to use the message file defined
    //in the ResourceBundle sample.
    msgLogger.setMessageFile("acme.widgets.DefaultMessages");

    // Get a reference to the Handler and remove it from the logger.
    RASIHandler aHandler = null;
    Enumeration enum = msgLogger.getHandlers();
    while (enum.hasMoreElements()) {
         aHandler = (RASIHandler)enum.nextElement();
         if (aHandler instanceof WsHandler)
              msgLogger.removeHandler(wsHandler);
    }

    // Create the user handler and formatter. Configure the formatter,
    // then add it to the handler.
    RASIHandler handler = new SimpleFileHandler("myHandler", "FileName");
    RASIFormatter formatter = new SimpleFormatter("simple formatter");
    formatter.addEventClass("com.ibm.ras.RASMessageEvent");
    handler.addFormatter(formatter);

   // Add the Handler to the logger. Add the logger to the list of the
   //handlers listeners, then set the handlers
   // mask, which will update the loggers composite mask appropriately.
   // WARNING - there is an order dependency here that must be followed.
   msgLogger.addHandler(handler);
   handler.addMaskChangeListener(msgLogger);
   handler.setMessageMask(RASIMessageEvent.DEFAULT_MESSAGE_MASK);
```

Chapter 11. Working with troubleshooting tools

WebSphere Application Server includes a number of troubleshooting tools that are designed to help you isolate the source of problems. Many of these tools are designed to generate information to be used by IBM Support, and their output might not be understandable by the customer.

This section only discusses tools that are bundled with the WebSphere Application Server product. A wide range of tools which address a variety of problems is available from the (WebSphere Application Server Technical Support Web site).

Steps for this task

1. Select the appropriate tool for the task.

   For more information on the capacities of the supplied troubleshooting tools, see the relevant articles in this section.
2. Run the tool as described in the relevant article.
3. Contact IBM Support for assistance in deciphering the output of the tool.

Collector tool

The collector tool gathers information about your WebSphere Application Server installation and packages it in a Java archive (JAR) file that you can send to IBM Customer Support to assist in determining and analyzing your problem. Information in the JAR file includes logs, property files, configuration files, operating system and Java data, and the presence and level of each software prerequisite.

There are two phases of using the collector tool. The first phase runs the collector tool on your WebSphere Application Server product and produces a Java archive (JAR) file . The IBM Support team performs the second phase, which is analyzing the Java archive (JAR) file that the collector program produces.

The collector program runs to completion as it creates the JAR file, despite any errors that it might find. Errors might include missing files or commands. The collector tool collects as much data in the JAR file as possible.

Running the collector tool

The collector tool gathers extensive information about your WebSphere Application Server installation and packages it in a Java archive (JAR) file that you can send to IBM Customer Support to assist in determining and analyzing your problem. Information in the JAR file includes logs, property files, configuration files, operating system and Java data, and the absence or level of each software prerequisite.

The collector program runs to completion despite any errors that it might find. Errors might include missing files or commands. The collector tool collects as much data in the JAR file as possible.

You can also run the collector summary option to create a lightweight version of the information in a text file and on the console. The lightweight information is useful for getting started in communicating your problem to IBM Support.

Steps for this task

1. Log on to the system as **root** (or **Administrator** in a Windows platform).
2. Verify that Java 1.2.2 or higher is available in the path.

   The collector program requires Java to run. It also collects data about the Java Development Kit (JDK) in which it runs. If there are multiple JDKs on the system, verify that the JDK that the WebSphere Application Server product uses, is the one in the path for the collector program. If the JDK being used by the WebSphere Application Server is not available, putting another JDK in the path for the collector program lets you collect all data but information about the JDK.
3. Verify that all necessary information is in the path being used by the collector program and that you are not running the program from within the WebSphere Application Server product installation root directory.
   a. If this system is a Linux or UNIX-based platform, verify that the path contains:
      - `/bin`
      - `/sbin`
      - `/usr/bin`
      - `/usr/sbin`
   b. If this system is a Windows platform, include `regedit` in the path.
4. Create a work directory where you can start the collector program.
5. Make the work directory the current directory.

   The collector program writes its output JAR file to the current directory. The program also creates and deletes a number of temporary files in the current directory. Creating a work directory to run the collector program avoids naming collisions and makes cleanup easier. You cannot run the collector tool in a directory under the WebSphere Application Server installation directory.
6. Run the collector program by entering the command: `collector` from the command line.

   Using the **collector** command with no additional parameters gathers one copy of the node data and data from each server in the node, and stores them in a single JAR output file. To gather data from a specific server in the node, use the command `collector servername` , where `servername` is the name of the problem server.

   Note:

   Set the path correctly to use the non-qualified version of the command. For Linux and UNIX-based platforms, `install_root/bin` must be in the path to locate the **collector.sh** command. For Windows platforms, `install_root\bin` must be in the path to locate the **collector.bat** command.

   The WebSphere Application Server installation root directory is determined at installation. It is identified in the `setupCmdLine.sh` file (or the `setupCmdLine.bat` file on a Windows platform).

   You can enter a fully qualified path to the collector command. For example, enter this command in a default installation on a Windows platform:

   ```
   c:\WebSphere\AppServer\bin\collector.bat
   ```

Results

The collector program creates a log file, `Collector.log`, and an output JAR file in the current directory.

The name of the JAR file is based on the hostname and package of the Application Server product, in the format: `hostname- ND-WASenv.jar` or `hostname-Base-WASenv.jar`. For example, if you run the collector tool on the server `ws-laceweb` within a Network Deployment cell, the filename is `ws-laceweb-ND-WASenv.jar`.

The `Collector.log` log file is one of the files collected in the `hostname-ND-WASenv.jar` or `hostname-Base-WASenv.jar` file.

What to do next

Send the `hostname-ND-WASenv.jar` or `hostname-Base-WASenv.jar` file to IBM Support for analysis.

Analyzing collector tool output

The first step in using the collector tool on your WebSphere Application Server product is to run the tool to ″Running the collector tool″. The second step in using the collector tool is to analyze its output. The preferred method of performing this analysis is to send the JAR file to IBM Support for analysis. However, you can use this topic to understand the content of the JAR file if you perform your own analysis.

You can view the files contained in the JAR file without extracting the files from the JAR file. However, it is easier to extract all files and view the contents of each file individually. To extract the files, use one of the following commands:

- `jar -xvf WASenv.jar`
- `unzip WASenv.jar`

Wasenv.jar stands for the name of the JAR file that the collector tool creates.

The JAR file contains:

- A collector tool log file, `collector.log`
- Copies of stored WebSphere Application Server files and their full paths
- Operating system information in a directory named `OS`
- Java information in a directory named `Java`
- WebSphere Application Server information in a directory named `WAS`
- Collector shell script (or batch file) execution information in a directory named `debug`
- MQ information in a directory named `MQ`, if you installed WebSphere MQ or the embedded messaging feature
- A JAR file manifest

Tips and suggestions

- Unzip the JAR file to an empty directory for easy access to the gathered files and for simplified cleanup.
- Check the `collector.log` file for errors:
  - Some errors might be normal or expected. For example, when the collector attempts to gather files or directories that do not exist for your specific installation, it logs an error about the missing files.
  - A non-zero return code means that a command that the collector tool attempted to run does not exist. This might be expected in some cases. If this type of error occurs repeatedly, there might actually be a problem.

- On Linux and UNIX-based systems, the file `OS/commands` has the location of all commands used. If you are missing command output, check this file to see if the command was found.
- On Linux and UNIX-based systems, the collector runs some shell scripts. The shell script output is saved in files in the `OS` directory, while the corresponding debug information is saved in the `debug` directory. If the output of a shell script is missing, check the corresponding file in the `debug` directory.
- When you issue the **collector** command when there are multiple installation instances, the tool that runs depends on what is in the `PATH` statement. For example, if you install both the base WebSphere Application Server and the Deployment Manager product on the same machine, the `bin` directory that first appears in the `PATH` variable is the one that furnishes the collector tool. To work around this problem, use a fully qualified filepath when calling the collector tool as shown in this example for a Windows platform:

  ```
  c:\WebSphere\AppServer\bin\collector.bat
  ```
- On Windows systems, the `OS` directory contains a file named `installed.out`. This file contains a list of programs found in the Add/Remove Programs list. This same information is contained in the file `Desktop\My Computer\Control Panel\Add/Remove Programs\Install/Uninstall`.

Collector summary option

WebSphere Application Server products include an enhancement to the collector tool beginning with Version 5.0.2, known as the *collector summary option*.

The collector summary option helps you communicate with WebSphere Application Server technical staff at IBM Support. Run the collector tool with the `-Summary` option to produce a lightweight text file and console version of some of the information in the Java archive (JAR) file that the tool produces without the `-Summary` parameter. You can use the collector summary option to retrieve basic configuration and prerequisite software level information when starting a conversation with IBM Support.

To run the collector summary option, start from a temporary directory outside of the WebSphere Application Server product installation root directory and enter the following command:

- **Linux and UNIX-based platforms:**

  ```
  <install_root>/bin/collector.sh -Summary
  ```
- **Windows platforms:**

  ```
  <install_root>\bin\collector.bat -Summary
  ```

The collector summary option produces version information for the WebSphere Application Server product and the operating system as well as other information. It stores the information in the `Collector_Summary.txt` file and writes it to the console. You can use the information to answer initial questions from IBM Support or you can send the `Collector_Summary.txt` file directly to IBM Support.

Run the collector command to create the JAR file if IBM Support needs more information to solve your problem.

First Failure Data Capture tool

The First Failure Data Capture tool preserves the information generated from a processing failure and returns control to the affected engines. The captured data is saved in a log file for use in analyzing the problem.

The First Failure Data Capture tool is intended primarily for use by IBM Service. It runs as part of the IBM WebSphere Application Server, and you cannot start or stop it. It is recommended that you not attempt to configure the First Failure Data Capture tool. If you experience conditions requiring you to contact IBM Service, your IBM Service representative will assist you in reading and analyzing the First Failure Data Capture log.

The First Failure Data Capture tool does not affect the performance of the IBM WebSphere Application Server.

Log Analyzer

The Log Analyzer takes one or more service or activity logs, merges all of the data, and displays the entries. Based on its symptom database, the tool analyzes and interprets the event or error conditions in the log entries to help you diagnose problems. Log Analyzer has a special feature enabling it to download the latest symptom database from the IBM Web site.

To download the latest updates to the symptom database, use the **File -> Update Database -> WebSphere Application Server Symptom Database** option for WebSphere Application Server, or **WebSphere Application Server Network Deployment Symptom Database** option for WebSphere Application Server Network Deployment in the Log Analyzer interface.

About the service or activity log

The application server creates the service or activity log file from the activity of the various WebSphere Application Server components. Log Analyzer is used to view the service or activity log file. Log Analyzer can merge service or activity log files into one log file. The service or activity log file, `activity.log`, is a binary file in the `logs` directory of the install_root.

You cannot view the service or activity log with a text editor. The Log Analyzer tool lets you view the file.

Viewing a service or activity log file in the absence of a graphical interface

The Log Analyzer tool cannot view remote files. If the operating system on which you are running WebSphere Application Server does not support the use of a graphical interface, transfer the file in binary mode to the system on which you are running the Java administrative console. Use the Log Analyzer tool there.

In cases where transferring the file is impractical or inconvenient, use the alternate viewing tool, **showlog**, to view the service or activity log file:

1. Change directory to `bin` directory of the install_root.
2. Run the **showlog** tool with no parameters to display usage instructions:
   - On Windows systems, run **showlog.bat**.
   - On UNIX systems, run **showlog.sh**.

To direct the service or activity log (`activity.log`) contents to stdout, use the **showlog activity.log** command.

To dump the service or activity log to a text file for viewing with a text editor, use the **showlog activity.log** *textFileName* command.

Accessing Log Analyzer help files

You can access Log Analyzer help files on Windows platforms, using the operating system default Internet browser only. You cannot access the help files using an Internet browser other than the default. However, Windows does let you select either Netscape or Internet Explorer as the default browser.There is an option to let you select either Netscape or Internet Explorer as the browser to display HTML help files.

Access help files using any Internet browser on UNIX platforms. You can use such browsers as Netscape Navigator, by explicitly setting the location of its executable in the tool Preferences dialog. The option that appears to allow you to select either Netscape or Internet Explorer as the browser to display HTML help files is not used on UNIX systems.

To specify the browser on UNIX platforms:

1. Click **File > Preferences** in the Log Analyzer tool.
2. Click **Help** from the **General** folder in the Log Analyzer Preferences dialog.
3. Set the path to the Internet browser executable in the Browser Location field.

Installing Log Analyzer silently

Installing Log Analyzer ″silently″ prevents installation messages from being displayed, but the file `responsefile.txt` for silent installation needs more information to install Log Analyzer. To silently install Log Analyzer, add the following option to this file:

```
-P logAnalyzerBean.active="true"
```

The Performance and Analysis Tools property in the tile `responsefile.txt` needs to be set to `true` to install the Log Analyzer tool. The property in the `responsefile.txt` is: -P performanceAndAnalysisToolsBean.active=″true″.

Using the Log Analyzer

To view the service or `activity.log` using the Log Analyzer:

Steps for this task

1. Change directory to: *install_dir*/bin.
2. Run the waslogbr script file.

   This file is named:

   - `waslogbr.bat` on Windows systems.
   - `waslogbr` on UNIX systems.

   This script must be run from the *install_dir*/bin directory.

   This starts the Log Analyzer interface.
3. Select **File -> Open**.
4. Navigate to the directory containing the service or activity log file.
5. Select the service or activity log file and click **Open**.
6. To analyze the records, right click on a record in the tree on the left, select **UnitOfWorkView** from the right-click menu, and select **Analyze**.

   Now any records with a green check mark next to them match a record in the symptom database. When you select a check-marked record, you will see an explanation of the problem in the lower-right-hand pane.

Note: When starting the Log Analyzer for the first time, or after the Log Analyzer preferences files of the users have been deleted, the following message is displayed in the Log Analyzer's shell window:

```
Cannot open input stream for waslogbrsys
```

You can disregard this message, as it is informational and not indicative of abnormal operation.
WebSphere Application Server includes the following Log Analyzer files for use with the WebSphere Commerce Suite:

- install_root\bin:
  - wcslogbr.bat
  - wcslogbrsys.cfg
  - wcslogbrsys.ini
- install_root\properties\logbr:
  - wcsanalyzers.xml
  - wcslogtypes.xml
  - wcsrecdef.xml

You can ignore these files.

Log Analyzer main window

To view this page, launch the Log Analyzer, install_root/bin/waslogbr on UNIX systems or install_root\bin\waslogbr.bat on Windows NT or Windows 2000 systems. Click **Help > Tasks**.

The Log Analyzer takes one or more service or activity logs, merges all the data, and, by default, displays the entries in unit of work (UOW) groupings. It analyzes event and error conditions in the log entries to provide message explanations. The Log Analyzer main window interface has the following elements:

- Three window panes
- Status line
- Menu bar
- Pop-up actions

Window panes

The Log Analyzer window has three panes:

Logs pane (left)

By default, Log Analyzer Logs pane displays log entries by UOW. It lists all the UOW instances and its associated entries from the logs that you have opened. You may find the UOW grouping useful when you are trying to find related entries in the service or activity log or when you are diagnosing problems across multiple machines. The file name of the first log you open appears in the pane title bar. There is a root folder and under it, each UOW has a folder icon which you can expand to show all the entries for that UOW. All log entries without any UOW identification are grouped into a single folder in this tree view. The UOW folders are sorted to show the UOW with the latest timestamp at the top of the list. The entries within each UOW are listed in the reverse sequence, that is the first (earliest) entry for that UOW is displayed at the top of the list. If you have

merged several logs in the Log Analyzer, all the log entries are merged in timestamp sequence within each UOW folder, as if they all came from the same log.

Every log entry is assigned an entry number, Rec_*nnnn*, when a log is opened in the Log Analyzer. If more than one file is opened in the Log Analyzer (merged files), the Rec_*nnnn* identification will not be unique because the number is relative to the entry sequence in the original log file and not to the merged data that the Log Analyzer is displaying. This Rec_*nnnn* appears in the first line (**RecordId**) in the Records pane.

By default, each entry in this pane is color-coded to help you quickly identify the ones that have high severity errors. The values listed here are the default values, you can configure your own colors.

- – Non-selected log entry with background color of:
  - Pink indicates that it has a severity 1 error.
  - Yellow indicates that it has a severity 2 error.
  - White indicates that it has a severity 3 error.
  - Selected log entry with background color of:
    - Red indicates that it has a severity 1 error.
    - Green indicates that it has a severity 2 error.
    - Blue indicates that it has a severity 3 error.

  These colors are configurable and can be changed in the Log Analyzer Preferences Log page. See the help for the Severity page in the Log Analyzer Preferences notebook for different error severity levels and for more information on how to do this.

  The Log Analyzer can display the log entries in different groupings. Use the Log Analyzer Preferences notebook: Logs page to set the grouping filters.

  After the Analyze action has been invoked, each analyzed log entry has the following icons:
  - A check icon indicates that the entry has some analysis information in one or more pages in the Analysis pane.
  - A cascading plus sign (+) icon indicates that the entry has some analysis information and that it has a reraised or remapped exception. You may want to look at the log entry prior to this one when diagnosing problems.
  - A question mark icon indicates that the entry has either a severity 1 or 2 error but no additional analysis information is available for it.
  - An ″x″ icon indicates that the entry has a severity 3 error and it has no analysis information.

Record pane (upper right)

When you select an entry in the Logs pane, you see the entry in the Record pane. The entry identification appears in the pane title bar. Right-click in the Record pane to see actions that you can perform on the selected entry. A drop down arrow next to Record lets you look at the last ten records you have viewed. The cache for the historical data (10, by default) is set in the Preferences General page.

Note:

- The page does not display associated analysis data for these cached records. To see analysis information for cached data, reselect the entry from the Logs pane.
- You can enable or disable line wrap mode for the Record Pane using the Log Analyzer Preferences notebook: Record. To print contents of this pane, select **Record > Print** when the Record pane is in focus.

Analysis pane (lower right)

When the analyze action has been invoked and additional information is available, the information will appear in the Symptom page. If the page tab is grayed out, there is no information in that page. The pages of the Analysis pane are:

Symptom
The Log Analyzer provides a database of information on common events and errors to help you recover from some common errors. As a part of the analyze action, if such information is found in the database for the selected log entry, the information is displayed in this page.

Status line

There is a status line at the bottom of the window showing the status of actions.

Menu bar

The menu bar in the Log Analyzer main window, has the following selections:

File

Open...

Opens a new log file. You can select either a service or activity log or a previously saved XML file. If you want the Log Analyzer to format a raw log file (by running the showlog command) prior to opening it, name the log file with suffix.log. If the Log Analyzer finds that the .log file contains formatted data, it skips the showlog formatting step.

If you want to merge data from another log, select **Merge with**.

Merge with...

When another log file is already opened in the Log Analyzer, use the **Merge with** action to open subsequent logs. The Log Analyzer merges the data from all the logs that it opens and displays all the entries within timestamp sequence in the UOW folders. The data appears as if they came from one log.

If you want the Log Analyzer to format a raw log file (by running the showlog command) prior to opening it, name the log file with suffix.log. If the Log Analyzer finds that the .log file contains formatted data, it skips the showlog formatting step.

Redisplay logs
To redisplay the logs using the recently set filters.

Save as...

Saves the log as an XML file (or text file). If **analyze action** has been performed, all the Symptom analysis information is also saved. If logs are merged in the Log Analyzer, the saved file contains entries of all the merged logs in the sequence that is shown in the Logs pane.

Note: If the merged logs have different timestamp formats, you should not save the merged information because the Log Analyzer only recognizes a single timestamp format for each file that it opens.

Save

Is only enabled if the first file that you opened is an XML file. It resaves the XML file with all the data that is currently displayed in the Log Analyzer. If **analyze action** has been performed, all the Symptom analysis information is also saved. If logs are merged in the Log Analyzer, the saved file contains entries of all the merged logs in the sequence that is shown in the Logs pane.

Note: If the merged logs have different timestamp formats, you should not save the merged information because the Log Analyzer only recognizes a single timestamp format for each file that it opens.

Print Log...
Prints all the entries that the Log Analyzer is displaying. If logs are merged in the Log Analyzer, the output contains entries of all the merged logs in the sequence that is shown in the Logs pane. If analyze action has been performed, all Symptom analysis information is also printed. To print parts of the log, use **Record > Print**.

Close Closes the opened log.

Update Database
Updates the symptom database which is used for Symptom analysis. It downloads the latest version of the symptom database from the URL specified in the ivblogbr.properties file.

Preferences...
Lets you configure and change the appearance of the Log Analyzer window and its contents.

Exit Exits the Log Analyzer and closes its window.

Edit

Copy Copies the selected text in the Record or Analysis pane to the clipboard. If you have not selected any text, **Copy** does not appear in the menu.

Find Allows you to find text strings in the focused pane.

View

Logs Toggles the visibility of the Logs pane.

Record
Toggles the visibility of the Record pane.

Symptom
Toggles the visibility of the Symptom page in the Analysis pane.

Record

All the actions under this menu applies to the focused pane.

To select several entries, hold down the **Ctrl** key when making the selection. When a folder is selected, the action applies to all the entries in that folder.

Analyze

Retrieves and displays additional documentation on known events and event messages in the Analysis pane (Symptom page). Select the folders and entries in the Logs pane, right-click to select the **Analyze** action, or from the menu bar, select **Record > Analyze**.

Note: If you invoke Analyze for the root folder, then all the entries in the log that you are viewing will be analyzed. If some analysis information is available for an entry, it will either have a check icon or a cascading plus sign (+) icon next to it in the Logs pane. If the analyze action has already been performed, the selection will be grayed out.

Save to file

Saves the selected entries in the Logs pane. If folders are selected, all the entries in the folder are saved. Any retrieved analysis information is also saved. If the focused pane is either the Records or Analysis pane, then only information in that pane is saved.

Print

- If the focused pane is Logs, the action prints the selected folders and entries. Any retrieved analysis information for those entries is also printed.
- If the focused pane is Record, the action prints the entry that is currently in the Record pane. Any retrieved analysis information is not printed.
- If the focused pane is Analysis, the action prints Symptom page contents.

Windows

If you detach the Symptom page in the Analysis pane into separate windows, all windows appear under this menu. You can select windows to bring them to the foreground.

Help Provides a list of online documentation for additional information.

Pop-up actions

In the focused pane, right-click to bring up a list of actions in a pop-up menu. Actions that you cannot perform are grayed out. When a folder is selected in the Logs pane, the action applies to all entries in that folder. To select several folders or entries in the Logs pane, hold down the **Ctrl** key when making the selection.

Log Analyzer find window

To view this page, launch the Log Analyzer, install_root/bin/waslogbr on UNIX systems or install_root\bin\waslogbr.bat on Windows NT or Windows 2000 systems. Click **Edit > Find**.

The Log Analyzer Find window lets you look for text strings in the focused pane. For example, if you remember the Unit of Work identification, you can enter that text string in the Find window to quickly locate the Unit of Work folder in the Logs pane.

Log Analyzer Preferences notebook - General

To view this page, launch the Log Analyzer, install_root/bin/waslogbr on UNIX systems or install_root\bin\waslogbr.bat on Windows NT or Windows 2000 systems. Click **File > Preferences > General**.

The General page of the Log Analyzer Preferences notebook lets you specify the behavior of panes in the Log Analyzer window:

Show title bars
Shows the title bars of window and its panes.

Highlight selected pane
Highlights the pane that is in focus.

Pane history cache size
Specifies a number of records to save in the cache. The Log Analyzer keeps a history of the (specified number of) records that you have viewed. You can use the drop down list next to Record in the pane title bar to see these cached entries.

Note: The associated analysis data for these records are not saved. To see analysis information, reselect the entry from the Logs pane.

Show logo at startup
Shows the logo when you start-up the Log Analyzer.

When you are finished, click **OK** to apply your changes and close the Preferences notebook.

Log Analyzer Preferences notebook - Appearance

To view this page, launch the Log Analyzer, install_root/bin/waslogbr on UNIX systems or install_root\bin\waslogbr.bat on Windows NT or Windows 2000 systems. Click **File > Preferences > General > Appearance**.

The Appearance page of the Log Analyzer Preferences notebook lets you define the overall appearance of the Log Analyzer. You can select the family of products and its texture schemes that you want the Log Analyzer window to emulate.

When you are finished, click **OK** to apply your changes and close the Preferences notebook.

Log Analyzer Preferences notebook - Toolbars

To view this page, launch the Log Analyzer, install_root/bin/waslogbr on UNIX systems or install_root\bin\waslogbr.bat on Windows NT or Windows 2000 systems. Click **File > Preferences > General > Toolbars**.

The Toolbars page of the Log Analyzer Preferences notebook lets you customize the appearance and contents of the toolbar in the Log Analyzer window. You can select whether there is text and/or icon in the toolbar, as well as, the functions that you want in the toolbar.

When you are finished, click **OK** to apply your changes and close the Preferences notebook.

Log Analyzer Preferences notebook - Help

To view this page, launch the Log Analyzer, install_root/bin/waslogbr on UNIX systems or install_root\bin\waslogbr.bat on Windows NT or Windows 2000 systems. Click **File > Preferences > General > Help**.

The Help page of the Log Analyzer Preferences notebook lets you select the browser that is to display online help files.

For Windows, the default Web browser is used. You need not update any settings unless there are problems when bringing up the default browser.

For AIX, HP-UX, and Solaris, you must update the following settings, especially the full path of the browser in the **Browser location** entry.

Help browser
Select the Web browser you want to use.

Browser location
Select the location of the browser executable file. This should be correct by default, but if you cannot access help then you may need to explicitly enter the browser location.

When you are finished, click **OK** to apply your changes and close the Preferences notebook.

Log Analyzer Preferences notebook - Proxy

To view this page, launch the Log Analyzer, install_root/bin/waslogbr on UNIX systems or install_root\bin\waslogbr.bat on Windows NT or Windows 2000 systems. Click **File > Preferences > Proxy**.

The symptom database included in the Log Analyzer package contains entries for common events and errors. New versions of the symptom database provide additional entries. Download new versions of the database from the IBM FTP site. The URL for the FTP site is located in file: *install_dir*/bin/ivblogbr.properties.

The default setting for the FTP site is:

```
ftp://ftp.software.ibm.com/software/websphere/info/tools/
loganalyzer/symptoms/std/symptomdb.xml
```

You can update your symptom database in one of two ways:

1. Download a new version from the FTP site, and replace your existing database with the new version. Your database is: *install_dir*/symptoms/std/symptomdb.xml.
2. Use the Log Analyzer graphical user interface (GUI) to update your database by selecting: **File -> Update database -> WebSphere Application Server Symptom Database** (for WebSphere Application Server) or **WebSphere Application Server Network Deployment Symptom Database** for WebSphere Application Server Network Deployment.

Setting the proxy definition

If your organization uses a FTP or SOCKS proxy server, contact your system administrator for the host name and port number of the proxy server.

If you use the Log Analyzer GUI to update the database, you can add a proxy definition to the Proxy Preferences page as described below:

1. Select **File -> Preferences -> Proxy**.
2. Select the appropriate proxy type.
3. Enter the host name and port number of the proxy server on the **Proxy** panel.

If you do not use the Log Analyzer GUI, add the proxy definition to the command that launches Log Analyzer.

- Do the following to add the proxy definition for the FTP proxy server:
  - For Windows:
    1. Modify file: *install_dir*\bin\waslogbr.bat.
    2. Add the following text to the file:

```
%JAVA_HOME%\bin\java -DIVB_HOME=%USERPROFILE%/logbr ^
....
-Dftp.proxyHost=proxy_host -Dftp.proxyPort=port_number ^
```

  - For UNIX:
    1. Modify file: *install_dir*/bin/waslogbr.
    2. Add the following text to the file:

```
$JAVA_HOME/bin/java -ms10m -mx255m -DIVB_HOME=$HOME/logbr \
....
-Dftp.proxyHost=proxy_host -Dftp.proxyPort=port_number \
```

- Do the following to add the proxy definition for the SOCKS proxy server:
  - For Windows:
    1. Modify file: *install_dir*\bin\waslogbr.bat.
    2. Add the following text to the file:

```
%JAVA_HOME%\bin\java -DIVB_HOME=%USERPROFILE%/logbr ^
....
-DsocksProxyHost=proxy_host -DsocksProxyPort=port_number ^
```

  - For UNIX:
    1. Modify file: *install_dir*/bin/waslogbr.
    2. Add the following text to the file:

```
$JAVA_HOME/bin/java -ms10m -mx255m -DIVB_HOME=$HOME/logbr \
....
-DsocksProxyHost=proxy_host -DsocksProxyPort=port_number \
```

Log Analyzer Preferences notebook — Logs

To view this page, launch the Log Analyzer, install_root/bin/waslogbr on UNIX systems or install_root\bin\waslogbr.bat on Windows NT or Windows 2000 systems. Click **File > Preferences > Logs**.

The Logs page of the Log Analyzer Preferences notebook lets you group the entries in the logs by different entry fields for viewing. For example, you can select to group the log entries by TimeStamp or clientHostName when they are displayed in the Logs pane.

Primary sort field

Use this filter to set the first level of grouping when log entries are displayed in the Logs pane. By default, the log entries are grouped by UnitOfWork.

Secondary sort field

Use this filter to set the second level of grouping (that is, within the grouping of the primary sort field) when log entries are displayed in the Logs pane.

All the entries within the grouped folders are always sorted in timestamp sequence with the earliest entry at the top of the list.

Redisplay log file immediately

Select this box to immediately regroup the logs entries (after you have clicked **OK**) based on the new filter settings. The entries in the Logs Pane are redisplayed according to the new grouping. If you want to delay the grouping, then do not select this box and, at a later time, you can use the **File > Redisplay logs...** menu selection to regroup and display the log entries based on the changed filter settings.

When you are finished, click **OK** to apply your changes and close the Preferences notebook.

Log Analyzer Preferences notebook — Severity

To view this page, launch the Log Analyzer, install_root/bin/waslogbr on UNIX systems or install_root\bin\waslogbr.bat on Windows NT or Windows 2000 systems. Click **File > Preferences > Logs > Severity**.

The Severity page of the Log Analyzer Preferences notebook lets you change background colors of log entries that appear in the Logs pane. Use colors to quickly indicate entries with high severity errors and the currently selected entry.

Use colors to indicate severities

Select this checkbox to color-code the background of log entries and folders. When selected, the radio button selections in this page are enabled.

Background color

For each folder and entry in the Logs pane, there is some text describing the entry. To choose a background color for selected log entry that has a severity 1 error, do the following:

1.
   - Select **Selected node**.
   - Select **Severity 1**.
   - Select the color by clicking on the color Swatches. To use the default setting, click **Restore Default**. To see the results of your change, look at the Preview box.
   - Click **Apply** to save that setting.

Repeat similar steps to change the background color for selected log entries that have severity 2 and 3 errors.

To choose a background color for an unselected log entry that has a severity 1 error, do the following:

1. Select **Unselected node**.

2. Select **Severity 1**.
3. Select the color by clicking on the color Swatches. To use the default setting, click **Restore Default**. To see the results of your change, look at the Preview box.
4. Click **Apply** to save that setting.

Repeat similar steps to change the background color for unselected log entries that have severity 2 and 3 errors.

Sample
You can see the result of you color change prior to applying the change. Look at the nodes shown in the Sample box. For color changes of selected nodes, click on the node in the sample box to see the color change.

When you are finished, click **OK** to apply your changes and close the Preferences notebook.

Log Analyzer Preferences notebook — Analyzer output

To view this page, launch the Log Analyzer, install_root/bin/waslogbr on UNIX systems or install_root\bin\waslogbr.bat on Windows NT or Windows 2000 systems. Click **File > Preferences > Analyzer output**.

The Log Analyzer Preferences notebook lets you enable line wrap for information that appears in the Analysis pane.

Set line wrap
Select the appropriate checkbox to enable line wrap for the Symptom page that appears in the Analysis pane.

When you are finished, click **OK** to apply your changes and close the Preferences notebook.

Log Analyzer Preferences notebook — Record

To view this page, launch the Log Analyzer, install_root/bin/waslogbr on UNIX systems or install_root\bin\waslogbr.bat on Windows NT or Windows 2000 systems. Click **File > Preferences > Record**.

The Record page of the Log Analyzer Preferences notebook lets you set the line wrap mode for the data that is displayed in the Record pane. It also lets you set the time and date format of the timestamp displayed in the Record pane.

Enable line-wrap mode for record pane
Select this checkbox to enable line wrapping for the information that appears in the Record pane.

Date format
When viewing logs, this date format only changes the timestamp format that is displayed in the Record pane. The timestamp format in the log file and the timestamp shown in the Logs pane are not affected by this setting.

Time format
When viewing logs, this time format only changes the timestamp format that is displayed in the Record pane. The timestamp format in the log file and the timestamp shown in the Logs pane are not affected by this setting.

When you are finished, click **OK** to apply your changes and close the Preferences notebook.

Installing the Log Analyzer silently

To silently install the Log Analyzer, you must set the relevant properties in the `responsefile.txt` file.

Steps for this task

1. Add the following option to the `responsefile.txt` file:
   - `-P logAnalyzerBean.active="true"`
   - Set the Performance and Analysis Tools property in the `responsefile.txt` file to true for the Log Analyzer to install.

     The property in the responsefile.txt file is:

     –

     `-P performanceAndAnalysisToolsBean.active="true"`

Accessing the Log Analyzer help files

For Windows platforms, you can only access the Log Analyzer help files using the operating system default Internet browser. You cannot access the help files using any Internet browser, even though there are options allowing you to select either Netscape or Internet Explorer and set the location of the browser to display HTML help files.

For UNIX platforms, you can access the help files using any Internet browser, such as Netscape Navigator, by explicitly setting the location of the browser executable in the tool Preferences dialog. The option that seemingly allows you to select either Netscape or Internet Explorer as the browser to display HTML help files is not used.

The following steps describe how to specify the browser on UNIX platforms:

Steps for this task

1.
   - In the Log Analyzer tool, select **File -> Preferences**
   - In the Log Analyzer **Preferences** dialog, click **Help** from the**General** folder.
   - Set the path to the Internet browser executable in the **Browser Location** field.

Chapter 12. Diagnosing and fixing problems: Resources for learning

In addition to the WebSphere Application Server Version 5 InfoCenter, there are several Web-based resources for researching and resolving problems related to the WebSphere Application Server.

The WebSphere Application Server support page

The official site for providing tools and sharing knowledge about WebSphere Application Server problems is the WebSphere Application Server support page: **http://www.ibm.com/software/webservers/appserv/support.html** . Among the features it provides are:

- A search field for searching the entire support site for documentation and fixes related to a specific exception, error message, or other problem. Use this search function before contacting IBM Support directly.
- *Hints and Tips*, *Technotes*, and *Solutions* links take you to specific problems and resolutions documented by WebSphere Application Server technical support personnel.
- A link *All e-fixes, fixpaks, and tools* provides free WebSphere Application Server maintenance upgrades and problem determination tools.
  - e-fixes are software patches which address specific WebSphere Application Server defects. Selecting a specific defect from the list in the *All e-fixes, fixpaks, and tools* page takes you to a description of what problem the e-fix addresses.
  - Fixpaks are rollups of multiple efixes, tested together and released as a maintenance upgrade to WebSphere Application Server. If you select a fixpak from this page, you are taken to a page describing the target platform, WebSphere Application Server prerequisite level, and other related information. Selecting the *list defects* link on that page displays a list of the e-fixes which the fixpak includes. If you intend to install an e-fix which is part of a fixpak, it is usually better to upgrade to the complete fixpak rather than to just install the individual e-fix.
  - Tools are free programs that help you analyze the configuration, behavior and performance of your WebSphere Application Server installation.

Accessing WebSphere Application Server support page resources

Some resources on the WebSphere Application Server support page are marked with a key icon. To access these resources, you must supply a user ID and password, or to register if do not already have an ID. When registering, you are asked for your contract number, which is supplied as part of a WebSphere Application Server purchase.

WebSphere Developer Domain

The Developer Domains are IBM-supported sites for enabling developers to learn about IBM software products and how to use them. They contain resources such as articles, tutorials, and links to newsgroups and user groups. You can reach the WebSphere Developer Domain at **http://www7b.software.ibm.com/wsdd/** .

Chapter 13. Obtaining help from IBM

If you are not able to resolve a WebSphere Application server problem by following the steps described in the Troubleshooting guide, by looking up error messages in the message reference, or looking for related documentation on the online help, contact IBM Technical Support.

Purchase of WebSphere Application Server entitles you to one year of telephone support under the Passport Advantage program. For details on the Passport Advantage program, visit http://www.lotus.com/services/passport.nsf/WebDocs/Passport_Advantage_Home

The number for Passport Advantage members to call for WebSphere Application Server support is 1-800-237-5511. Please have the following information available when you call:

- Your Contract or Passport Advantage number.
- Your WebSphere Application Server version and revision level, plus any installed e-fixes.
- Your operating system name and version.
- Your database type and version.
- Basic topology data: how many machines are running how many application servers, and so on.
- Any error or warning messages related to your problem.

The Collector Tool

WebSphere Application Server comes with a built-in utility that collects logs and configuration information into one file, the Collector Tool. IBM Technical Support may ask you to run this tool and submit the output.

Tracing

WebSphere Application Server support engineers might ask you to enable tracing on a particular component of the product to diagnose a difficult problem. For details on how to do this, see ″Enabling trace″.

Consulting

For complex issues such as high availability and integration with legacy systems, education, and help in getting started quickly with the WebSphere product family, consider using IBM consulting services. To learn about these services, browse the Web site http://www-1.ibm.com/services/fullservice.html.