
CHAPTER 25

Problem Prevention and Determination Methodology

Objectives

This chapter covers the following concepts:

- Problem prevention best practices
- Change control best practices
- WebSphere best practices
- Working with IBM WebSphere support

25.1 Problem Prevention Best Practices

25.1.1 Testing Best Practices

One of the best ways to avoid and flush out problems prior to releasing a new application into the production environment is to properly test it. To most, the concept of testing is an obvious prerequisite to putting an application into the production environment. However, it is important to note that the key is not just “testing,” but “proper testing”. To paraphrase a famous statement, it is not whether you test, but *how* you test. A large percentage of the problems that occur in production environments can be prevented if the application is properly tested. It is the goal of this section to describe a test methodology that, when followed, will result in a significantly reduced risk of production outages.

25.1.1.1 Properly Scaled Tests

Prior to testing an application, it is important to ensure the environment in which it is tested is appropriate for that application. An appropriate test environment for an application is one that is a scale model of its production environment, including all components from Web server to data-

base. Having an exact replica of the production environment for testing may not always be economically feasible; however, it is highly recommended that the test environment be an accurate model of the production environment.

What Does It Mean to Have a Scale Model of Production in Test? Prior to discussing the benefits of having a scale model of the production environment in test, it is important to clearly articulate what we mean when using this phrase. It is broader than simply having a fraction of the production computing power in the test environment. As an example, imagine a production environment consisting of four machines, each with four processors. Two possible test configurations may be

1. One 4-processor machine
2. Four 1-processor machines

While it is true that both of these proposed test environments have one-quarter the processing power of the production environment, both leave out complexity. The first configuration, one 4-processor machine, disregards the complexity of having the application distributed across multiple systems. The second option, four 1-processor machines, ignores the impact of having multiple processors per machine. Although the number of processors per machine does not directly affect the administrative configuration of the WebSphere environment, it does impact the ability to troubleshoot the application. More specifically, this configuration increases the difficulty of finding synchronization bugs.

Most synchronization problems are only found on multiple processor machines where true multithreading and parallelization occur. Thus, not testing on multiprocessor servers can lead to deploying an application, fraught with synchronization problems, into the production environment. A more correct scale would be to have four 2-processor machines or, at a minimum, two 2-processor machines.

Another key component to having a scale model of the production environment in test is to ensure that the entire path length of the application is tested with no parts skipped, ensuring that all tests are functionally and systematically complete. This includes having all database connections, Web servers, proxy servers, messaging servers, firewalls, and so on, configured and running in the test environment as they will in production. Failing to test the entire environment may result in deployment or run time failures.

Benefits of a Scale Model The most obvious reason for testing an application is for correctness. It is important to verify that the application actually works as designed, but this is only one of the reasons for testing. Another commonly overlooked reason for testing is to allow you to become familiar with the dynamics of the application, its attributes in steady state, startup, and under load in a safe environment. This is important because it will allow you to quickly identify and isolate anomalies when the application enters the production environment. If the test environment is not to scale of the production environment, then the dynamics observed will

change as the application moves to production, thus crippling your problem determination ability and putting the company in a dangerous situation.

Maximizing performance tuning efforts is yet another reason for having a scale model of the production environment in test. Performance is one of the key characteristics of any application, often the second most important next to correctness. For an application to achieve maximum performance, it must be tuned. Performance tuning is a highly iterative process dependent on many parameters set both within an application server and the wider environment. If the test environment is not to scale of the production environment, much of the hard work that went into tuning the application and its environment is wasted because the parameters that achieved the best results cannot be directly applied in production. A concrete example of a parameter that has a large effect on an application server's performance, thus an application's performance, is the Java Virtual Machine's heap size. For example, if, by tuning, it is discovered that a maximum heap size of 256 megabytes achieves the best performance in the test environment and the test environment is one-half scale of production, then this number can be doubled in production with a higher degree of confidence than if the test environment were not to scale of production.

25.1.1.2 Isolated Test Environment

To ensure the accuracy of any test, it is important to mandate that the test environment be isolated from other environments and activities. This is necessary to ensure that the observed behaviors are the result of the application or services with which the application is interacting and not some external event. An isolated test environment consists of machines dedicated to running WebSphere, an isolated subnet, dedicated database, and any other external systems and resources. One of the most frustrating situations when attempting to debug a problem is being unable to isolate its source, only to later find out that the problem occurred because of some external situation.

25.1.1.3 Test Scenarios

To ensure that an application is ready to be deployed in the production environment, care must be taken to ensure that the scenarios under which it is tested are realistic. Realistic test scenarios consist of two parts.

The first step in creating realistic test scenarios is to ensure that the scripts used for testing the application are realistic examples of what a user might do in production. General usage scenarios are usually known prior to developing an application in the architecture phase; however, like most things, enterprise applications are not always used the way their architects envisioned. Instead, people use the application in the way that feels most intuitive to them. Therefore, it is best to get test cases from the users themselves. There are several ways to gather this information, which vary depending on what is being deployed into production.

If the application is an update to an existing application already in production, then real-world usage can be tracked on the current production version of the application, either by logging or with an external monitoring tool capable of capturing transactions.

If the application is completely new, with functions never before available to its users, then the task of gathering real-world scenarios becomes more difficult. One option is to use IBM's

Rational Unified process or the Extreme Programming paradigm at <http://www.extreme-programming.org/>, in which the end users are constantly evaluating the application as it is being developed. Another option is to verify the application's general functionality via some architected use cases then release the application to a small alpha or beta test group to gather more user oriented test cases.

**GOTCHA**

A commonly forgotten step when creating test scripts is to validate the returned results. By default, many of the stress-testing applications only check for a HTTP 200 response, which is insufficient when testing J2EE applications. If the results are not verified, the application could be displaying incomplete Web pages or, worse yet, the data for a different user.

Once the test cases have been created, the next step is to run them against the application in the test environment. As it is important to gather realistic test scenarios, it is also important to run a diverse mixture of these scenarios while testing the application. The mixture should be a representative set of your user population. For instance, if there are six different user types expected in the production environment and the majority (50 percent) are expected to be of one type (type A) while the remaining 50 percent will be equally distributed among the five remaining user types, then you will want to run the tests with a similar distribution. In this example 50 percent of the virtual users would be executing test A while the remaining 50 percent would be distributed evenly among the other test cases.

**GOTCHA**

It is important to capture and test all possible user types even if there is one user type that is only used once in a great while. An example would be an administrator performing some weekly or monthly functions. Though this administrator may be less than 1 percent of the user population, it is important to understand the implications of that user's actions.

**TIP**

While monitoring the performance of the application and its dependent parts, also monitor the statistics of the testing machine. Never let the stress testing software or hardware be the bottleneck in a test environment.

Test Types After the test scenarios have been created, the next step is to execute them in various test types. The first tests that should be run are those that verify the correctness of the application. There are several others tests that go beyond just testing functionality of the application which should also be run. Three of the most important tests are performance, stress, and endurance. Describing the detailed steps required to execute each of these tests is beyond the scope of this book, so we will give a brief overview of each and pointers to more in-depth resources as appropriate.

The performance test is probably the most well-known test type. In a performance test, as the name suggests, the goal is to optimize the performance tuning parameters to maximize the

overall performance of the applications being tested. The process for performance tuning WebSphere Application Server is described in great detail in chapter 22 of this book and in the whitepaper, *WebSphere 4.0 Performance Tuning Methodology* found at http://www.ibm.com/software/webservers/appserv/doc/v40/ws_40_tuning.pdf. The paper was written for the 4.0 version of WebSphere Application Server; however, the logic and methodology detailed is applicable to every WebSphere release, including the latest, V5.1.

One of the main themes of this chapter, thus far, has been the importance of having a scale model of your production environment in test. A key component of this is having an accurate model of user workload to ensure that the planned capacity is enough to handle the estimated user load. Stress testing is designed to apply user workload, above and beyond what is expected, in an attempt to find the breaking point of the application and its environment. This type of test has several benefits. First, it will help you better understand the characteristics of the application under high load and what piece of the application is likely to fail. If a potential weak link can be identified, procedures can be put into place to deal with the problem, allowing you to be proactive instead of reactive. The second benefit is that application bugs, which might not surface under normal load, will get flushed out. One of the most prominent types of these bugs are those caused by synchronization (or lack of it). Flushing out synchronization bugs is a function of time, load, and parallelism. The larger these components are, the greater the possibility of finding a synchronization problem. Stress testing increases the load part of this equation.

The last type of test we discuss is the endurance test. The length of the endurance test is what differentiates it from all other tests. Most tests last several minutes or, at most, an hour, but in the endurance test the application is run under heavy load, as expected in production, for many hours or even days. The purpose of this is to uncover problems that may appear after extended usage. There are many problems that fall into this category, including session problems, intermittent failures, and the aforementioned synchronization bugs.

25.1.2 Change Control Best Practices

The production environment should be a strictly controlled environment. Failure to adhere to a stringent change control policy can result in inexplicable problems in the production environment. This section describes some best practices associated with change control with the production environment.

25.1.2.1 Restricting Administrative Privileges

The first step in ensuring proper change control is to limit the number of people who have administrative privileges on the production machines. Oftentimes we have encountered production systems on which many people in the organization have administrative privileges, leading to a myriad of problems. This is especially problematic if the production machines are shared across many departments within the organization. In most cases, each department will have their own set of priorities, and, if the administrative privileges are not restricted to one or a select few administrators, people can unwittingly make changes that impact other applications within the environment.

25.1.2.2 Access History

Even with proper restriction on administrative privileges, problems can occur because of administrative errors. When they happen it is useful to have a log detailing who made the last change, at what time, and from which remote machine. This is not to place blame but rather to identify what actions caused the problem and to be able to inquire as to why that particular action was performed. Once the problematic action is identified, an alternative procedure can be created to avoid future outages.

Another reason to keep an access history is to identify hackers. By following the potential hacker's actions, you can see what techniques he/she is using to access your site, and you may be able to identify their intentions if they should break in.

25.1.2.3 Backing Up Your Configuration

Prior to making any configuration change, it is a good idea to back up the current, working configuration. WebSphere provides a utility, `backupConfig`, to aid this process. `BackupConfig` is command line utility (found in the `<WASND_ROOT>/bin` and `<WAS_ROOT>/bin` directories) that creates a backup of the installation's configuration files, including application and server data. `BackupConfig` can be run by executing the provided executable. Be aware that its default behavior is to stop all WebSphere processes prior to performing the backup. This behavior can be overridden by supplying the switch `-nostop`. The complete command would be `backupConfig -nostop`.

**TIP**

When using WebSphere in a distributed environment, `backupConfig` only needs to be run on the Deployment Manager.

To restore from a previously saved configuration, use the `restoreConfig` command. Like `backupConfig`, `restoreConfig`'s default behavior is to stop the WebSphere processes prior to restoring the configuration. Again, the `-nostop` command line option can be used to override the default behavior.

**TIP**

Unfortunately, we are all human and it is possible to forget to back up the configuration prior to making a change. Creating cron jobs, a scheduled execution of a task, is one way to overcome this human limitation. By creating a daily or weekly cron job that runs the `backupConfig` command, you can ensure that you have a recovery point even if you forget to manually run the `backupConfig` command.

Beyond backing up the configuration, it is also a good idea to backup the entire WebSphere installation directory immediately after installation, before and after any major changes, such as adding a node to a cell. This is useful in the event an administrator accidentally deletes a critical file, such as `startupServer.bat` or `java.exe`. Mistakes such as these are not uncommon, especially when working from the command line and dealing with many different directories at once.

Regardless of whether this has happened in your organization before, it is best to protect against these types of mistakes and create regular backups.

25.1.2.4 Maintain a Log History

Usually a problem is discovered only after several occurrences and perhaps not until an external complaint is logged. By keeping a log history, you can uncover how long this problem has been occurring and potentially find a pattern in the events that caused the problem to occur.

25.1.2.5 Documented Procedures

It is easy to forget a step when performing a complex operation such as installing WebSphere or upgrading an application. The best way to avoid such errors is to document the steps for these complex operations. The document should consist of clear and detailed steps with screen shots that show expected results to avoid possible confusions and misinterpretations. Every administrator should have a hard copy of these documents and follow them to the letter.

Execution of the procedures should involve, at the very least, two people. The first administrator executes the steps and the other verifies them.



TIP

Automating the documented procedures is an advanced technique that, when implemented properly, eliminates all potential user error. The risk is that there is no human control on the process. For more information on automated scripting, refer to chapter 20, Automated WebSphere Administration.

25.1.3 WebSphere Best Practices

IBM provides numerous white papers and articles about best practices and design patterns for WebSphere. The WebSphere Developer Domain Web site is a great place to find many useful WebSphere articles, including best practices-specific papers. A Web site that hosts an extensive collection of best practices for administering the WebSphere Application Server Web site is also available at <http://www.software.ibm.com/wsdd/zones/bp>. IBM best practices are reviewed and updated for new versions of WebSphere. It is recommended you stay informed and follow the best practices published by IBM.

25.1.3.1 Application Best Practices

Application performance and scalability is heavily influenced by the design of the application, database, and other resources. The importance of following application best practices is underestimated in many cases. A good application design for performance and scalability follows certain fundamental best practices and avoids common mistakes. Following the WebSphere application best practices is a good starting point for designing applications with great performance and scalability. Performance, in particular, suffers from poorly designed and implemented applications.

25.1.3.2 Code Reviews

Code reviews are an extremely effective way to find code bugs and, thus, prevent problems, but many times they are not fully completed due to project deadlines or other time constraints. Code reviews are performed by programmers who are not the authors of the code that is investigated. In many cases code reviews are as effective as or even more effective than testing.

Code reviews include simple things like checking for naming conventions, code indenting or adding Javadoc comments, and more complex tasks like code optimization or logic checking. The tasks in the first category make your code easier to maintain and reuse, while tasks in the second category check the correctness and improve the performance of your application. Code optimization like caching data, efficient use of database connections, or eliminating unnecessary object instantiation can significantly increase performance.

The good news is that you can use Java code analyzers to automate much of the code review process. Most Java code analyzers detect things like unused or duplicate imports, unused private and local variables, missing Javadoc comments, violation of naming conventions, and empty “catch” blocks or “if” statements. You will still have to check for logic errors and code optimization, but most of these simple (but time-consuming) code checking tasks are automated.

**TIP**

You can integrate Java code analyzers with Ant. In this way you can check your code any time you run a unit test or complete a build.

25.1.4 WebSphere Fix Packs and Interim Fixes

In many cases, the solution to your problem may be as simple as applying a fix pack or interim fix. For this reason, we recommend that you stay up to date with fix packs for your version of WebSphere and check for interim fixes when problems occur. Although applying a fix pack requires careful planning, this operation pays off in most cases by fixing or preventing problems. It is recommended that you also keep current supporting products such as DB2, and so on.

**TIP**

Fix packs should be tested in nonproduction environment first. Occasionally features in a fix pack may not be compatible with existing applications.

Interim fixes are WebSphere code fixes created for known individual problems and should be applied when you have a critical problem without a valid workaround. Interim fixes are individually tested and are integrated with the next WebSphere fix pack. Make sure that you check the WebSphere Application Server fix packs and interim fixes Web site <http://www.ibm.com/software/webservers/appserv/was/support/> when a problem occurs in order to determine if this is a documented problem. Each released version of WebSphere details the defects that were fixed in its accompanied release notes. You can also create a “My support” profile to receive weekly e-mail notifications about IBM product updates. For WebSphere, go to <http://www-306>

[.ibm.com/software/webservers/appserv/was/support/](http://ibm.com/software/webservers/appserv/was/support/) and select My Support from the right menu.

25.2 Problem Determination Methodology

Often, when an error occurs in WebSphere, the user knows a problem has occurred but doesn't know what to do about it or what it means—the user just knows that something is broken. This section takes you through WebSphere problem determination methodology, a series of rules to follow when you encounter a problem, to help pinpoint its origin and take it to resolution.

Why is methodology important? The foundation of any problem determination process is good methodology. Knowing how to go about determining if you have a problem, where the problem exists, and the basics of how to solve that problem are important to any enterprise project.

You might be asking, “Why not tell us how to troubleshoot WebSphere?” Troubleshooting a product the size of WebSphere could fill a book on its own. WebSphere is a very large and complex piece of software and to try to address in detail how to debug each specific component is beyond the scope of an administration book. While this subsection does not cover detailed troubleshooting of the application server, it does lay the foundation for troubleshooting your environment if and/or when a problem occurs. This methodology section lays out a set of rules you can employ when doing in-depth problem determination. Following these rules when a problem occurs can help expedite locating the problem (the first step in any problem determination process) and then resolution.

25.2.1 Locating the Error in a Complex Environment

When a problem occurs, pinpointing its origin can take some detective work, especially in a complex environment where multiple tiers with multiple products are integrated with one another. Knowing where each product's log files are located is an important step in knowing your environment, since the log files often provide very useful information. We have detailed where to locate and how to read WebSphere's log files in chapter 24, WebSphere Problem Determination Tools—Logging and Tracing. Often, when troubleshooting in a complex environment, problem determination becomes a team effort, involving administrators and application developers from each component in the environment.

This is especially true in the initial stages of determining where the problem resides (e.g., which tier, which component/product). Rarely is one person an expert in each component that makes up the environment (e.g., the back-end, the Web server, the edge component, the authentication server, the application, etc.). It is often necessary to involve people from various roles to assist in determining where the problem is or is not located. For example, if you are experiencing the problem when testing a new build of an application, make sure the necessary application developers are available to help determine if the error could be originating from the application code. Or, if the error is occurring when accessing the back-end data store, involve the database administrator to assist in determining if the error is originating from the database.

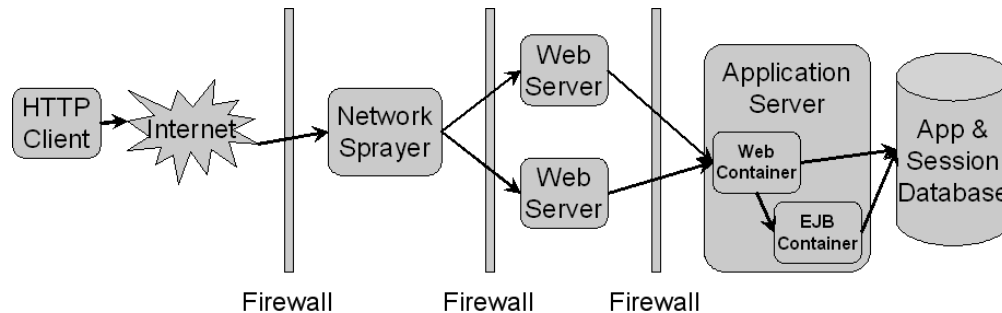


Figure 25.1 Example request path through enterprise environment.

To help pinpoint the component where the problem exists, it is useful to create a diagram of the path a request would take, assuming it did not fail, beginning with the client all the way to the back-end, depicting each component the request touches. For example, assume that a request for a simple servlet is failing. If you were to map the flow of a request in a simple environment beginning from a browser, it might look something like the servlet request goes through the proxy firewall to the network sprayer; from the network sprayer through the domain firewall to the Web server; from the Web server through an additional firewall to WebSphere Application Server, which then responds back to the Web server that responds to the browser.

At any one of these points, the request might fail. However, the request can be tracked by looking at access and error logs of each of the components involved. Additionally, this might require enabling trace on the various components to track the requests. For example, the access log on WebSphere's embedded HTTP Server might be enabled so you can verify that the request from the Web server made it into WebSphere. Possibly, one of the components is throwing error messages into the log files, or there are communication issues between two or more of the components involved. Once the failing component(s) is located, a more thorough examination of the error can begin. Problem determination of products outside the WebSphere Application Server or products installed and running in WebSphere Application Server (like WebSphere Portal or a custom J2EE application) is out of the scope of this section. While this section is mainly geared toward problem determination with regard to the WebSphere Application Server run time, some of this methodology still applies to external products and applications.

25.2.2 Could the Error Be Valid?

Once the error is isolated to a particular component(s) within the environment, one of the first things to evaluate is the error code, message, and any associated stack trace that appears. Often these error codes and/or messages provide useful information as to what went wrong. Most products and protocols also have guides that provide additional information on particular error codes. WebSphere has a Message Reference guide that is a subsection of the InfoCenter documentation. This Message Reference section has a description of each error code that WebSphere can log in

Table 25.1 IBM Product and Protocol Message Reference Guides

Product or Protocol	Message Reference Guide	Where to Locate
WebSphere Application Server	Message Reference Section of Info Center Documentation	Online InfoCenter Documentation: www.ibm.com/software/webservers/appserv/infocenter.html
DB2 Universal Database	Message Reference, vol. 1 and 2 Guides	Online DB2 Core Documentation: www.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/v8pubs.d2w/en_main
IBM Http Server	Troubleshooting Section of InfoCenter Documentation	Online Infocenter Documentation: www.ibm.com/software/webservers/httpservers/doc/v1326/manual/ibm/index.html
Apache Server	Online Documentation and FAQ	Online Apache Documentation Project: httpd.apache.org/docs-project/
Hypertext Transfer Protocol (HTTP)	Status Code Section of RFC2616	Online RFC 2616: www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

its trace or log files. For information on how to read or locate WebSphere error codes and messages, please refer to chapter 24. Also, in Table 25.1, we are providing information on where to locate additional commonly used IBM product documentation, as well as protocol error codes.

Sometimes, pertinent information can be located in different product log files, which can be aligned by timestamp. For example, an exception occurring in the database could provoke error messages to be logged in the application server log files, as well as the Web server logs. Once you locate one error message, you can use its associated timestamp to cross-reference the database, application server, Web server, log files, and so on. This is also a good technique to use when locating the root of the error.

**TIP**

It is important to have the system clocks synchronized on each machine such that time stamping cross-referencing can be easily used. If the system clocks are not synchronized, the logs can still be cross-referenced; however, the times must be skewed appropriately.

When tackling a problem, it is always good to first assume that the error is valid before declaring that there is a bug or a defect in the running product. The error code and associated message can often be enough for you to diagnose and fix the problem.

For example, let us take a look at a fix pack installation problem scenario. Upon installing a fix pack onto an existing WebSphere configuration, an error occurs preventing the installation. The initial reaction to such a failure could be to assume that there is a defect with the installation or the WebSphere run time. Listing 25.1 depicts a portion of a reproduced log file with the problem.

Listing 25.1 Portion of log file from failing installation of a WebSphere fix pack.

```
...
Results:
=====
Time Stamp (End)      : 2003-07-15T17:08:42-04:00
EFix Component Result : failed
EFix Component Result Message:
=====
WUPD0239E: Fix removal failure: The processing of fix
WAS_WSADIE_ND_01_16-2003_5.0_cumulative, component prereq.wsadie
failed. See the log file
C:\\WebSphere\\DMgr\\properties\\version\\log\\20030715_210842_WAS_WSADIE_ND
_01-16-2003_5.0_cumulative_prereq.wsadie_uninstall.log for processing
details.
=====
EFix Component Installation ... Done

Exception: WUPD0223E: Fix uninstall failure: The update for component
{1} for fix pre-req.wsadie could not be installed .

...
```

As you can see from the log file excerpt, an exception occurred that prevented the installation. This log file also referenced an additional file for more information (see the highlighted portion of the log file above). Upon investigation of the referenced log file, 20030715_210842_WAS_WSADIE_ND_01-16-2003_5.0_cumulative_prereq.wsadie_uninstall.log, additional information about the problem is uncovered. Listing 25.2 shows a reproduced portion of the referenced log file, 20030715_210842_WAS_WSADIE_ND_01-16-2003_5.0_cumulative_prereq.wsadie_uninstall.log.

Listing 25.2 Portion of log file from failing installation of a WebSphere fix pack.

```
...

2003-07-15T17:08:42-04:00 Applying entry 1 of 5 20% complete
2003-07-15T17:08:42-04:00 Preprocessing entry (restore):
```

```
2003-07-15T17:08:42-04:00 No EAR processing noted.
2003-07-15T17:08:42-04:00 Next entry name: lib/jdom.jar
2003-07-15T17:08:42-04:00 entry path: C:\WebSphere\DMgr\lib\jdom
.jar
2003-07-15T17:08:42-04:00 Error 16--File could not be deleted:
C:\WebSphere\DMgr\lib\jdom.jar
2003-07-15T17:08:42-04:00 Fetching entry ...
2003-07-15T17:08:42-04:00 Preprocessing entry (restore):
2003-07-15T17:08:42-04:00 No EAR processing noted.
2003-07-15T17:08:42-04:00 Next entry name: lib/marshall.jar
2003-07-15T17:08:42-04:00 entry path: C:\WebSphere\DMgr\lib\
marshall.jar
2003-07-15T17:08:42-04:00 Error 16--File could not be deleted:
C:\WebSphere\DMgr\lib\marshall.jar
2003-07-15T17:08:42-04:00 Fetching entry ...
2003-07-15T17:08:42-04:00 Preprocessing entry (restore):
...
```

Again, we have highlighted the errors in the log file excerpt—you can see that some of the jar files being replaced during the installation of the fix pack could not be removed.

**TIP**

A log file can have a tremendous amount of information in it. Sometimes searching for “rro” or “xception” can help pinpoint problems easily. Notice in both search string the “E” was left off such that capitalization does not limit the search.

Since these jar files could not be removed, the installation was failing. With this information in hand, we can begin to diagnose the problem—first assuming that the error was valid. Why couldn't the files be removed? The following options could all be valid possibilities:

- The jar files did not exist in the first place.
- The person running the installation did not have the appropriate permissions to remove files on the operating system.
- The files were locked by a running process.

After validating that the jars did exist and the installer had the appropriate permissions, the last option was investigated. A quick check of all running processes uncovered that WebSphere was still running while the fix pack was attempting to be installed. Since WebSphere was still running, it had locked the jar files to prevent run time corruption. So, in fact, the problem was not a defect or bug in WebSphere's installation of the fix pack; instead it was a valid response to an invalid operation (note that the fix pack installation directions require all WebSphere processes to be stopped before running the installation program). Once all WebSphere processes were stopped, the fix pack installation was successful.

25.2.3 What Has Changed?

When an error occurs, another technique to help pinpoint the root of the problem is to determine what might have changed to invoke the error. For example, did the error occur just after you ran a new test scenario, or did the error begin after you adjusted some TCP configurations on your operating system? Rarely does an error “just begin happening” if nothing was changed in the environment (network, operating system, application, server configuration, etc.). Therefore, it is an important part of problem determination to uncover what might have changed to invoke the problem at hand.

In an earlier section, Change Control Best Practices, a change control process was detailed as a best practice for problem prevention. If this system is in place and adhered to, determining if something in the environment has changed becomes much easier. Also, note that when we say “environment” we do not just mean WebSphere administration. The environment encompasses much more than this—it includes items such as the operating system settings, application configuration and code, test cases, configuration of supporting products either running on WebSphere or communicating with WebSphere, such as the Web server, back-end, authentication server, portal server, edge components, and so on. Determining if something has changed can be more than asking yourself if you have recently altered a configuration setting (unless you are the only one with administrator privileges to every server in the environment). Communication, therefore, is the key, especially in a complex environment. We are often surprised how, in some environments, communication between the administrators of various components (WebSphere, Database, network, etc.), as well as with application development, is minimal. Often, an administrator will call a product support line before calling a coworker in a different department to see if they might have altered a configuration.

Pinpointing the change does not mean that a bug does not exist, nor does it mean that the problem is now solved. Often there is a good reason for the change that has been effected. However, knowing what the change is and how it affects the system is important in finding a solution (whether it is a product bug fix, a configuration tweak, etc.).

25.2.4 Simplify, Simplify, Simplify

When you are running in a complex environment and an error occurs, finding the problem can sometimes be equated to finding a “needle in a haystack.” With so many different products and configurations involved, solving the problem becomes like solving a multiple variable algebra problem: the greater the number of variables involved, the more complex it is to solve.

Also, there is not always just *one* item that causes a problem to occur. Rather, it can be a combination of settings, coupled with a particular path through running application code, that triggers the error. To help limit some of the variables in the problem, it is wise to strip the environment back to the simplest possible running environment in which the error still occurs.

The best problem determination environment is one where the error can be reproduced with the *simplest* test scenario, running the *simplest* application code, deployed in the *simplest* environment. In this environment, not only is it easier to describe the problem to support (if

necessary), but also it limits the number of variables involved in the problem, making it easier to determine a solution.

25.2.4.1 The Simplest Test Scenario

Evaluate the test scenario that prompts the error to occur. If the test scenario is testing multiple conditions, can the test be limited to only the condition that fails? By narrowing what is tested until you have located the simplest test scenario that still causes the failure, you can save time when rerunning the test scenario, as well as narrow the number of variables when reproducing the test. You might discover that it is the sequence in which the tests are run that causes the problem, and/or eliminate the components that appear to not related to the failure.

Additionally, if the failure is occurring during load testing, try to find the minimum amount of load that still reproduces the problem. For example, if the test does not fail with a single user, but fails with two users, there might be a thread synchronization error. If the tests only fail under high load, it might be that your application or environment needs to be tuned for performance (please see chapters 21 through 23 of this book to learn about performance as relates to WebSphere).

25.2.4.2 The Simplest Application

When running a complex application, it is often difficult to determine whether the source of the error resides in the running application, in the WebSphere run time, or in some other area. If you can eliminate the running application in the simplified environment by reproducing the error with an alternate, much simpler application, that is very beneficial. You can eliminate the enterprise application as the source of the problem by attempting to reproduce the error with one of the IBM WebSphere sample applications that are installed with WebSphere or by creating a very simple application that forces the error to occur.



TIP

The technique of using an IBM WebSphere sample application or a simple sample application that can reproduce the problem is especially beneficial when working with IBM support. If using a created simple application, include it with any documentation that is provided to WebSphere support, with a description of what it does and the error it causes. This can help expedite support's interaction in determining the problem.

25.2.4.3 The Simplest Environment

To locate the origin of the problem, either a product or a WebSphere component level, it is best to reproduce the problem with the simplest configuration possible. For example, if the problem is occurring with a Web application, remove the Web server from the environment by accessing the application directly via WebSphere's embedded HTTP server. If the problem is with persistent Enterprise Java Beans (EJBs), try to manually invoke some of the update or select queries on the back-end to validate that they run correctly. Some other suggestions for simplifying the problem determination environment include

- Disabling work load management (distributed only)
- Disabling security
- Disabling the JIT compiler
- Moving the test clients onto a machine that has a direct route to WebSphere (rather than having to go through firewalls or edge components)

25.2.5 Do You Have Enough System Resources?

When failures begin to occur during load tests, sometimes it is not due to a run time failure, but rather a potential performance issue. Please refer to the Part 5, WebSphere Performance, of this book for additional information on performance monitoring and tuning. However, do remember that every machine will have its limits. In some situations, additional hardware will be needed to support particular load requests.

Performance monitoring can also uncover problems such as memory leaks that can severely impact application performance. It is highly recommended to tune your application before releasing it in a production environment.

25.2.6 What to Do If the Problem Is in Production

When a failure occurs in a production environment, it is often a critical situation. If you believe the problem to be a WebSphere run time defect, it will be important to contact IBM WebSphere support (1-800-IBM SERV) immediately so they can begin investigating the problem. It is also important to make sure that no information surrounding the failure is lost. It is a best practice to backup all log files, including database and Web server logs, if applicable, so they can be referred to later, if necessary. Until a solution is found, rollback or disable any change or update that might have invoked the problem. In parallel, it is pertinent to try to reproduce the problem in a test environment. A problem that can be reproduced in test will lend itself to easier problem determination since detailed traces and logging can be enabled without fear of affecting performance or up-time in the production environment. It also provides an experimental environment for being able to freely alter configuration parameters, as well as providing a simpler, less complex problem determination environment.



TIP

When the cause of the problem is determined, use it as a lesson learned. It is important that the failing scenario works its way back into the test suite that is run before any application is released in production. This way, the problem can be prevented in the future. Be sure to update procedures and test cases to avoid this problem in the future.

25.2.7 Where to Go for Help

IBM has an extensive WebSphere support Web site that contains self-help and problem submission information. This page should always be used before contacting IBM support. The WebSphere support Web site is accessible at <http://www.ibm.com/software/webservers/appserv/was/support/>.

The self-help section of the WebSphere support Web site contains links to several online resources meant to help you troubleshoot a WebSphere problem. Using this site, you can search on keywords to find Frequently Asked Questions (FAQs), Technotes, Hints and Tips, and other documents that address existing WebSphere problems. FAQs document common problems and solutions. Hints and Tips contain information about installing, configuring, and troubleshooting WebSphere. Technotes are documents containing customer-reported problems and solutions. You can also download WebSphere tools and utilities, as well as WebSphere fix packs and interim fixes. The support page also contains links to educational material such as IBM online publications, redbooks, and white papers.

The WebSphere InfoCenter is another resource for self-help. The InfoCenter is available online at <http://www.ibm.com/software/webservers/appserv/infocenter.html> or it can be downloaded as a PDF file. The local version of the InfoCenter is also available as an Eclipse documentation plug-in and can be downloaded from <http://www.ibm.com/software/webservers/appserv/infocenter.html>. To view the local documentation, you also need to install the IBM WebSphere Help System, which is a viewer for displaying product or application information developed as Eclipse documentation plug-ins. The IBM WebSphere Help System is built on open source software developed by the Eclipse Project. The InfoCenter contains a problem determination section, and you can also search the InfoCenter using keywords. The developerWorks Web site contains very good information for WebSphere developers in the section dedicated to WebSphere, which is available at <http://www.ibm.com/developerworks/websphere/>. The WebSphere developerWorks is a great source for articles and best practices related to WebSphere products. The site also provides other features like code downloads, technology previews, and forums.

Other helpful WebSphere resources are the WebSphere newsgroups and WebSphere user-group forums. There are several such newsgroups and forums, and they usually contain very useful information provided by WebSphere users. Some of these newsgroups are monitored by IBM personnel, helping to ensure the integrity of information included within those newsgroups.

WebSphere Studio Application Developer and Site Developer V5.1 have a new feature that allows you to search on keywords for several products, including WebSphere Application Server. This new feature is provided in the form of several product-specific plug-ins. The search is performed on resources like the WebSphere support Web site, the WebSphere InfoCenter, and Google newsgroups. Access to these resources is provided from one central product-specific page. Besides search capabilities, the plug-ins also provide a collection of local documents for self-support. These documents are copies of FAQs, Technotes, Hints and Tips, and other resources that are frequently used by WebSphere support personnel. One advantage of having these local documents is that they are searched when you perform a search through the WebSphere Studio Application Developer or Site Developer Help menu. To access the product plug-ins, select Help → Help Contents from the main menu, and then click on Support information of the left side of the page. Please see Figure 25.2.

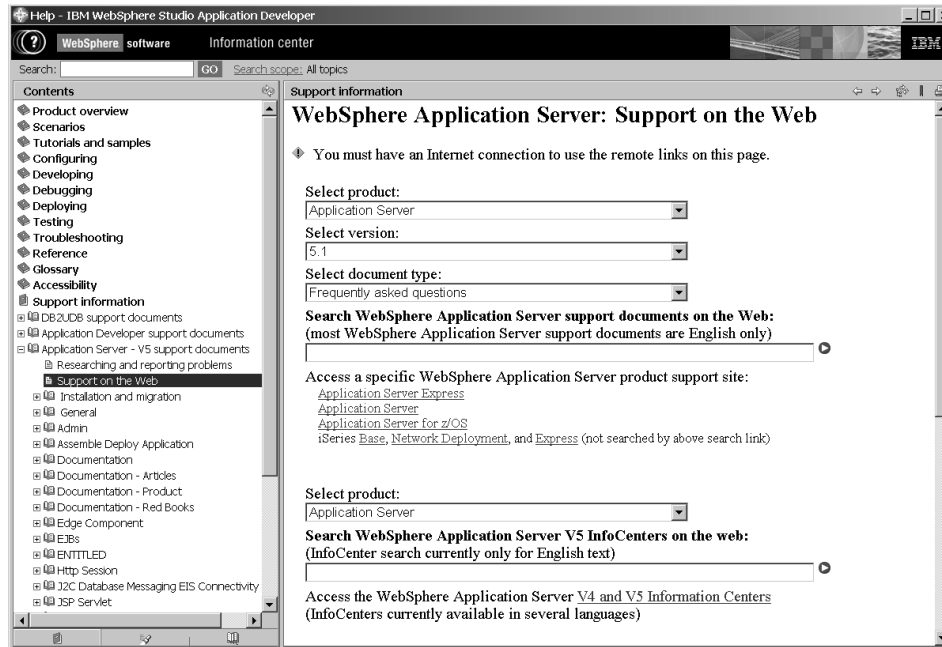


Figure 25.2 WebSphere Studio Application Developer Support information.

25.3 Working with IBM WebSphere Support

There are times when you need to work with IBM WebSphere support personnel to troubleshoot a WebSphere problem. The goal of this section is to familiarize you with the process of resolving a WebSphere problem with the help of IBM WebSphere support. We will tell you how to open a Problem Management Record (PMR), what information to have ready, and when you should involve IBM support to resolve a WebSphere problem.

25.3.1 When to Involve WebSphere Support

You have already tried to solve the problem by yourself. By now, you have searched the WebSphere support Web site, newsgroups, WebSphere Developer Domain, and other resources in order to determine if this is a known problem. You have also checked the latest WebSphere fix packs and interim fixes. Unfortunately you haven't found a match. It's time to engage IBM WebSphere customer support.

25.3.2 How to Open a PMR

There are two ways to open a Problem Management Record (PMR). You can submit an electronic PMR using the Electronic Service Request (ESR) tool, or you can call IBM customer support (1-800-IBM-SERV). ESR is a Web-based problem submission tool and is available from the

problem submission page at <http://www.ibm.com/software/support/probsub.html>. In order to use the ESR tool, you have to be enrolled in the IBM Passport Advantage Program and be registered as an authorized caller. Authorized callers are registered by the site technical contact, a person in your company who is responsible for maintaining the list of persons (in your company) authorized to use the ESR tool for problem submission. Once your site technical contact adds you as an authorized caller, you will be able to use the problem submission page to create a user ID and password. The user ID and password will be required to access the ESR tool. The ESR tool allows you to open new PMRs, as well as work with your existing PMRs.

**TIP**

The IBM Customer Number or Customer ID consists of 7 or 10 digits and is used to identify an IBM Passport Advantage support contract for a customer. Your company may have several contracts. Check with your site technical contact to determine the correct IBM Customer Number or Customer ID to use.

The following steps are required to submit a new PMR:

1. Click on your customer number.
2. Click on the Report a New Problem button.
3. Select the product for which you want the PMR to be created.
4. Select a component from the drop down list.
5. If necessary, edit the contact information in the Report a New Problem page.
6. Complete the rest of the fields in the Report a New Problem: Environment, Severity, Problem description, etc.
7. Click on Submit Problem Report. A page confirming that your PMR has been submitted to a support queue will be displayed.

Once the queued PMR is processed, a PMR number will be assigned to it and e-mailed to you. Keep this PMR number handy since you will need it every time you update the PMR or talk to IBM support.

**TIP**

The list of products and the list of components for which you can open PMRs can be quite long. You can use the Product Search and Component Search functions to find a specific product or component.

**TIP**

You can create and update a user profile in the ESR tool, with personal information like your phone number, e-mail, pager, and so forth. This profile will be used when you submit a PMR via the ESR tool. To access your profile, go to Update Maintenance Agreements from the main support page.

Figure 25.3 illustrates the ESR page that allows you to create a new PMR.

IBM

[Home](#) | [Products & services](#) | [Support & downloads](#) | [My account](#)

--Select a country

- ← Products & services
- IBM Software
- All software products
- Trials and betas
- News
- How to buy
- Software Support
- My Support
- Submit & track problems
- How to buy software support
- Help
- Site tours
- Feedback
- Related links:

Report a New Problem

You can create PMRs 24 hours a day, 7 days a week.

Contact Information

0123456 Customer Number

xxx Customer Branch

848 Customer Country

Contact Name (Required)

Contact Phone Number (Required)

Alternate Phone Number

Contact Email Address (Required)

Contact Pager Number

Contact Fax Number

Preferred Response Method [Help?](#)

Problem Details

Select Operating System (Required)

You are now at the Report a New Problem Page.

Figure 25.3 ESR page for creating a new PMR.

The other way to submit a PMR is to call IBM customer support (1-800-IBM-SERV). When you call IBM customer support, you will also need information such as your company's customer number, product, component, and so forth, similar to the information required by the ESR tool.

25.3.3 What Information to Have Ready

No matter how you open the PMR, electronically or by phone, you will need to provide information that will help the support personnel resolve your problem. Needless to say, you should be as specific as possible when describing the details of the problem. If you can re-create the problem, make sure you document the steps for IBM support. Also, gather as much background information as possible. For example, don't forget to specify what your environment is, what WebSphere fix packs you have installed, what Operating System and version you run, what changes you made to the system, and so on. If any messages or errors were logged, send these logs to IBM support. Any relevant information will help the IBM support personnel resolve your problem as quickly as possible. Basic background information is often missing from the PMR description, and this can significantly increase problem resolution time.

Another important piece of information you will be required to provide is the severity of the problem. Use the business impact as measure for determining the problem severity. For example, the most severe problems, those having a critical business impact, should be assigned a severity 1, while minor problems should have a severity 4.

It may also be useful to run the collector tool and have the output jar file ready to send to the IBM support personnel. For more information about the collector tool, refer to chapter 7, *Getting Started with WebSphere—An Overview*.

25.3.4 What to Expect

If a WebSphere defect is found, then an Authorized Program Analysis Report (APAR) is created for this problem. If a workaround can be implemented while the APAR is resolved, and before the fix is delivered, then IBM Support will provide instructions on how to do this. Note that making a fix available requires time-consuming operations such as comprehensive testing, packaging, and so on.

25.4 Summary

You should now be familiar with the following concepts:

- How to create an appropriate test. Specifically, it should be a scale model of the production environment.
- Which test types should be run to minimize the risk of encountering a problem in the production environment. These test types are correctness tests, performance tests, stress tests, and endurance tests.
- How to generate realistic test scenarios and proper workload mixes. These should be derived from real-world users.
- How to enact proper change control by restricting administrative privileges, logging administrative accesses, regularly creating backups of your configuration, keeping a log history, and documenting procedures.
- When locating an error in a complex environment, it is useful to diagram the path of the request, beginning with the client all the way to the back-end.
- Once an error has been located, first assume that the error is valid until it can be verified that it happened under erroneous circumstances.
- When an error occurs, try to determine what, if anything, might have changed in the environment that could have contributed to the condition which promoted the error.
- Try to re-create the error with the simplest test scenario, the simplest application, and the simplest environment.
- WebSphere interim fixes and fix packs.
- What resources are available for WebSphere support.
- Working with IBM WebSphere support to resolve a WebSphere problem.