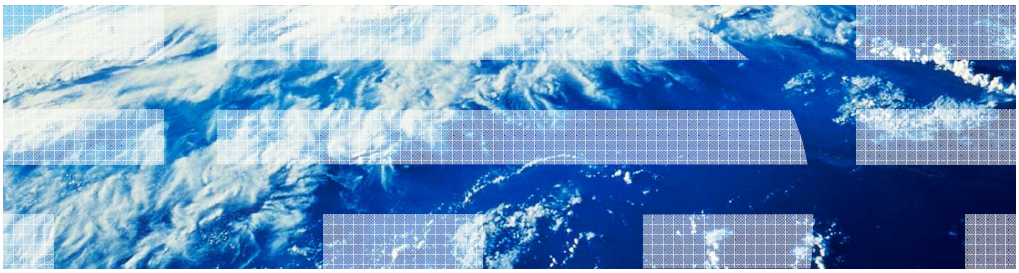
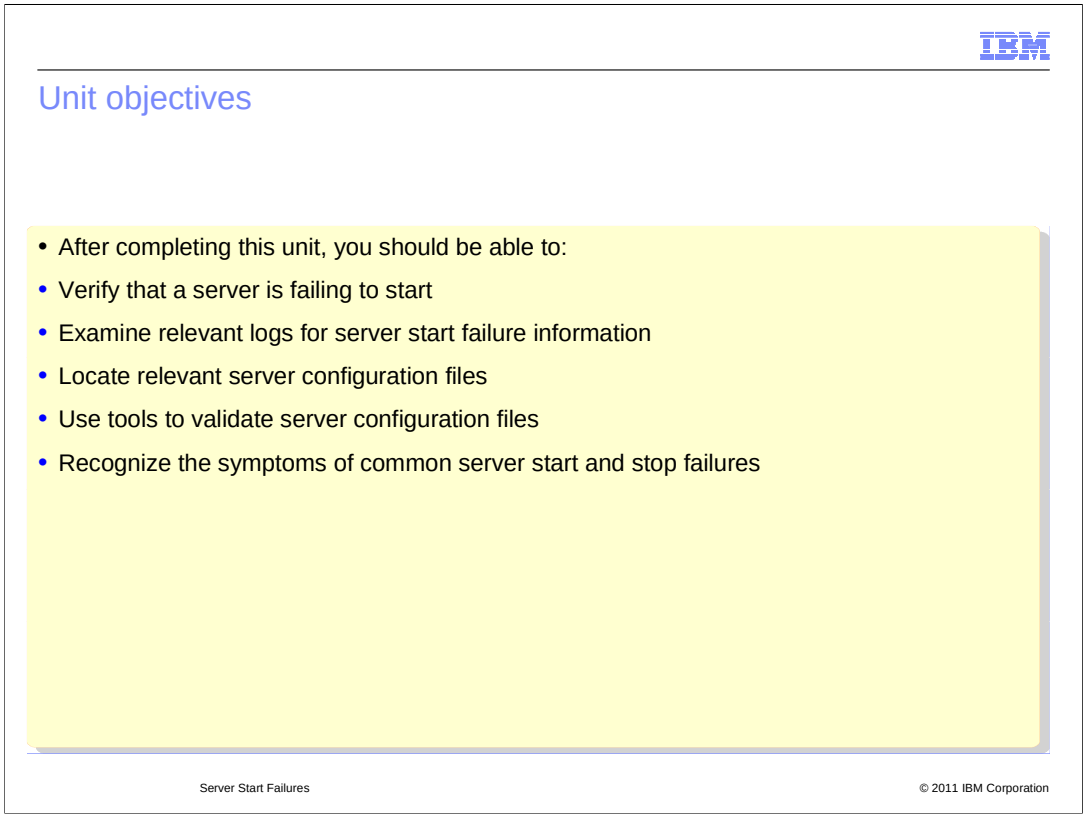
Server start failures



WebSphere software

4.0

This presentation describes diagnosing application server start failures.



After completing this unit, you should be able to detect a server start failure, validate configuration files, determine the root cause of the application server start failure, and be able to resolve the server start failure issue.

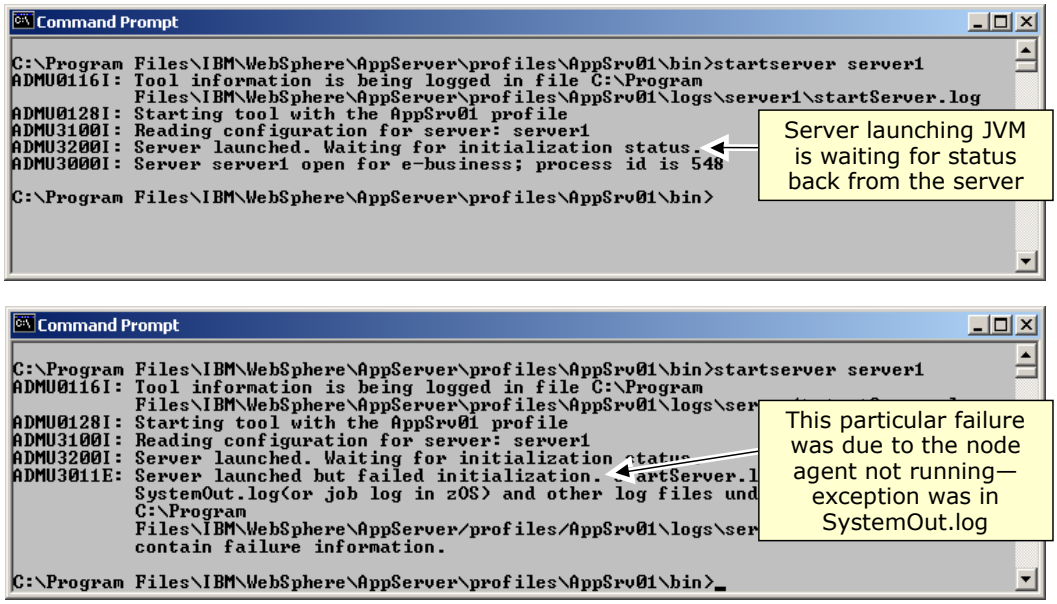
What happens when a server starts

- Issue the command: **startServer server1**
- Two JVMs are actually launched (except on iSeries).
- The first JVM is the Systems Management server launch utility.
  - Locates and reads the appropriate configuration (server.xml)
  - Spawns the second JVM, which is the actual server process
- The server launching process waits until it receives status back from the server process, and then exits.
  - Unless you specified the **-nowait** parameter

When a start command is issued, two JVMs are actually created. The first JVM is a launch utility that locates and reads the configuration required to start the server and then creates the server JVM using the acquired configuration values.

After spawning the server JVM, the launcher typically waits for a good status value to return from the server process.

Server start messages



In the successful server start show above, notice that the console messages (ADMU) are from the launch utility and the process ID for the actual JVM of server1 is reported.

The start failure shown above was caused by the fact that the node agent was not running. This condition is clearly reported in the SystemOut.log and SystemErr.log files of server1.

Server bootup process

- A relatively small amount of WebSphere code is placed onto the class path of the JVM. This includes the bootstrap code.
- The bootstrap code establishes a class loading environment for the rest of the installation image.
  - In V6 and earlier, this meant constructing a class loader that picked up everything in the lib directory (and others).
  - In V6.1 and beyond, this means establishing a proper OSGi environment so that the bundles are loaded.
- The bootstrap code then branches into the “main” of the server.
  - This main performs some preliminary processing (like loading the server.xml document) and then begins starting the various components of the system

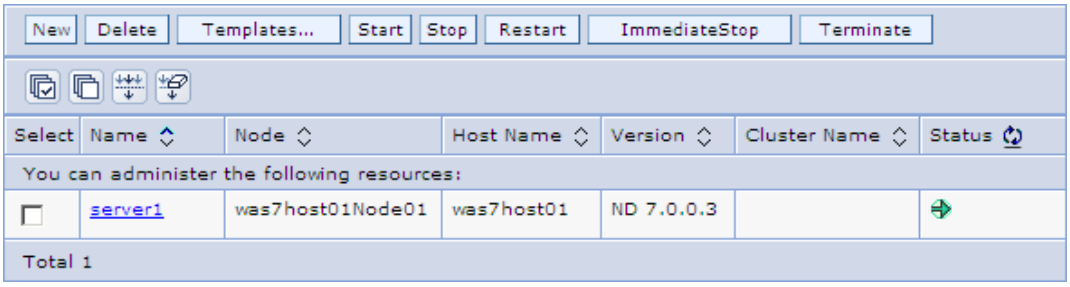
Server Start Failures

During the start of the server process, a small amount of WebSphere code is place into the class path of the JVM. This code is part of the bootstrapping process. The bootstrapping process establishes a class loading environment for the rest of the server’s runtime.

In WebSphere Application Server V7 and earlier, this meant constructing a class loader that picked up everything in the lib directory among other places. In WebSphere Application Server V6.1 and beyond, the bootstrap process establishes a proper OSGi environment so that all the OSGi bundles for are loaded. After this, the bootstrap code begins processing configuration information (like loading the server.xml file) and starts the various components of the server runtime.

Starting a server from the administrative console

- In a Network Deployment cell, application servers can be started from the administrative console.
- This server start process differs from the command line.
  - Does not use a separate launch utility JVM.
  - The node agent creates the server process.
  - No data is streamed to the startServer.log



Starting servers in an Network Deployment cell can be achieved through the use of the administrative console. This method of starting servers differs from the command line in that a new JVM is not created to create the server process. Instead, the Node Agent process creates the server process.

Detecting a failed server start (1 of 2)

- An application server failure during the start process is reported as an error message in the command window or administrative console.
- Further details on the error are logged as exceptions in the server logs.
  - native_stderr.log
  - native_stdout.log
  - startServer.log
  - SystemErr.log
  - SystemOut.log

  } If server fails early, you can *only* see these two logs
- If the server started from the administrative console of the deployment manager, then the errors might be viewable in the Runtime Error message panel.
  - Select **Troubleshooting > Runtime Message > Error > Runtime Events**.
  - Also check the SystemOut and SystemErr log files of the node agent.
- If the application server is started using the wsadmin script interpreter, the startServer.bat (Windows) or the startServer.sh (UNIX) script, the error message is printed in the command window

Server Start Failures

An application server failure during the start process is reported as an error message in the server start panel and the application server logs. (SystemOut.log and SystemErr.log for example)

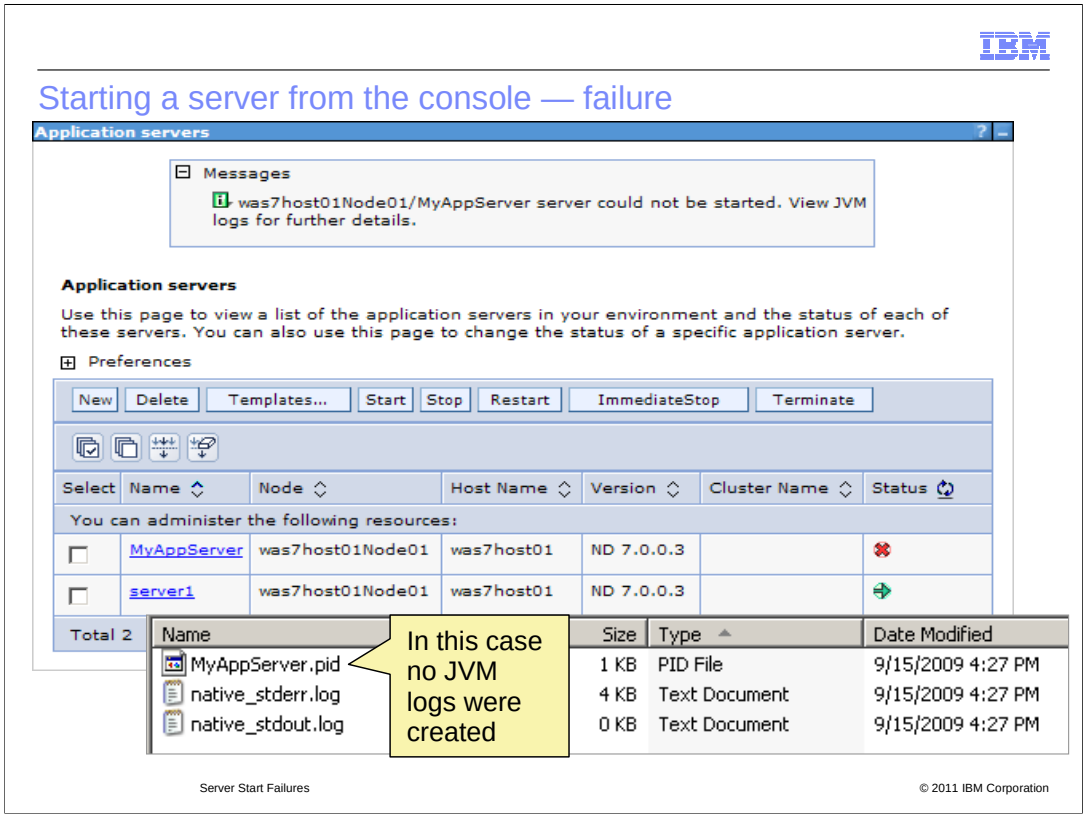
If the server is being started through the administration console, the error can be viewed in the Troubleshooting panel.

If the server is started using the startServer script, the error message is printed to the command line window.

During application server startup, various initialization steps occur, and message types are printed in the SystemOut.log. (WSVR and ADMU to name a few)

On a successful application server start, the message “Server is open for e-business” is displayed.

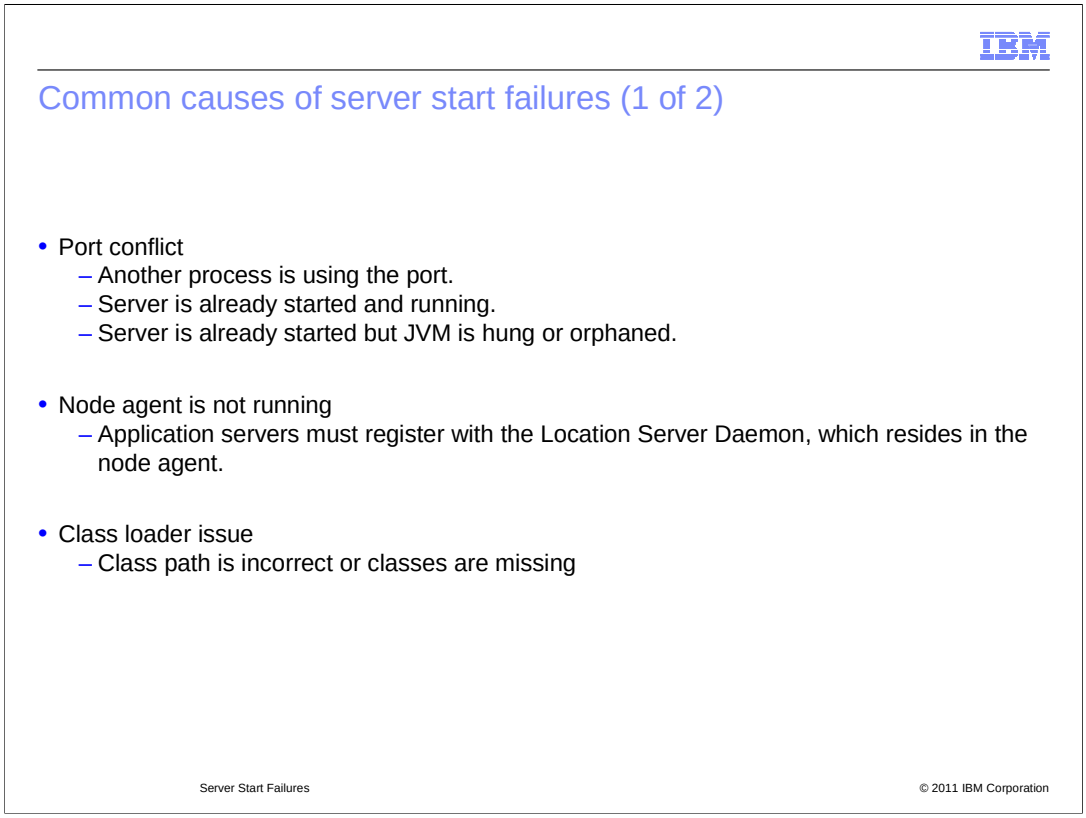
If the server start fails, the message displayed will depend on the cause of the failure.



This sample start failure shown was caused by setting the min and max heap of the server to 1MB (which is too low). Notice that only the native_stderr.log and native_stdout.log files were created.

If you were to look at these files, you would see an out-of-memory exception while the JVM was starting.

Note, that if a server fails to start very early in the process, only the native_stderr and native_stdout log files might have been created and updated.



Some common causes for a server not starting might be a port conflict (meaning the server might already be running or the JVM is a hung state), also if a class path is incorrect then you would run into a class loader issue.

Also, in an Network Deployment environment, the servers might not be able to start because the NodeAgent is not running. Application servers must register with the Location Server Daemon, which resides in the node agent.

Common causes of server start failures (2 of 2)

- Out-of-memory issue
  - Not enough native heap
  - JVM max heap size is too small
- Invalid JVM generic arguments or custom properties
  - JVM generic arguments are case sensitive.
  - Non-IBM JDKs recognize different generic arguments

Some other common reasons for the server not starting might be an Out-of-memory issue, because the JVM max heap size is too small, or a not valid JVM generic argument.

Determining server start issues (1 of 3)

- Begin with investigation of the log files to locate any exceptions that have occurred.
- Common server start failures will log exceptions.
  - Typically, there are multiple exceptions that occur during a server start failure.
  - The initial failure is the general exception accompanied by a more detailed exception regarding the actual cause of the failure.

Determining the root cause of an application start issue begins with investigating the log files for exceptions. There can be multiple exceptions that occur, but the initial failure is the general exception accompanied by a more detailed exception regarding the actual cause of the failure.

Determining server start issues (2 of 3)

- The application server can be started with the trace option specified on the command line.
  - **startServer.[bat | sh] <server name> -trace**
  - Start information will stream to the startServer.log file (otherwise not available).
- Using the **–trace** option is the same as setting the startup trace state to **com.ibm.*=all**
- The startServer.log file is very useful because some failures occur before the SystemOut.log and SystemErr.log trace streams being initialized.
  - If the server is started from the administrative console, there is no startServer.log. In that case, check the **native_std*.log**, and the node agent logs

When starting the application server, a trace option can be specified on the command line, which will stream start information to the startServer.log file.

Sometimes this can be very useful in troubleshooting start server issues, because some failures occur before the SystemOut.log and SystemErr.log trace logs being initialized.

You can also check the native_stdout.log, native_stderr.log, and the node agent logs if there is no startServer.log.

Determining server start issues (3 of 3)

- Enable server verbose modes.
  - Verbose class loading
  - Verbose garbage collection
- Can be enabled from the administrative console.
  - **Application servers > server_name > Process Definition > Java Virtual Machine**
- Trace information is written to native_stderr.log by default.
- If the administrative console is not available, edit the **server.xml** configuration file.
  - **verboseModeClass="true"**
  - **verboseModeGarbageCollection="true"**

Verbose class loading and garbage collection can also be enabled from the WebSphere administration console or in the server.xml file for gathering more diagnostic information to troubleshoot a start server failure.

This will provide debug output for native method invocation, and can be found in the native standard error log file.

Common configuration error messages

- **ADMU3402E**: Server name not specified; must supply the name of a server
- **ADMU3522E**: No server by this name in the configuration
- **ADML0029E**: No configuration defined for server
- **ADML0062W**: Cannot load server.xml for server
- **ADML0003E**: A configuration error occurred in the ProcessDef Umask property
- **ADML0004E**: An exception occurred when attempting to expand variable
- **ADML0006E**: The ${WAS_INSTALL_ROOT} variable is missing.

This slide demonstrates some common configuration error messages you might run into during start server. For example, the first item listed is a result of not supplying a server name when using the startServer.sh script, which is needed to start the server.

Resolving server start issues (1 of 2)

- Port conflict
  - Verify server ports using **serverindex.xml** of the node.
  - Check for used ports with **netstat -a**, **lsof -i** or port scanning tools.
  - Use the **ps –ef** command on UNIX or the **Task Manager** on Windows to determine orphan processes and end them.
- Node agent is not running
  - Start the node agent.
  - Check for orphaned node agent process, kill, and restart node agent.
- Class loader issue
  - Enable verbose class loading and look for ClassNotFound exceptions.
- Out-of-memory issue
  - Locate Java core dump and analyze with a thread analyzer tool
  - Locate heap dump and analyze with a heap analyzer tool

Server Start Failures

Resolving start server issues take a little patience troubleshooting the issue. You will need to view different log files to determine the failure. (startServer, SystemOut, and SystemErr to name a few).

If you determine the failure was caused by a port conflict, you can start by verifying server ports in serverindex.xml file of the node, or run the “netstat –a” command from the command line, or use the Task Manager on Windows to determine orphan processes and end them, to name a few.

If you determine the failure was caused because the Node agent is not running, then start the Node agent or check to make sure the process is not in a hung state, if so, end it and restart the Node agent.

If the failure is do to an Out-of-memory issue locate the javacore (thread dump) or heap dump (system core) and use the proper tool to analyze the problem.

Resolving server start issues (2 of 2)

- Administrator specified not valid JVM generic arguments or custom properties.
  - Check the **server.xml** file for the application server and if you see any generic JVM arguments, try removing them and see if the server starts.
    - Make sure you make a backup to server.xml file.
  - In a network deployment environment make the changes from the administrative console and then synchronize the node.

Server Start Failures

Configuration parameters (JVM arguments for example) or references changed improperly, can cause start server issues. Check the proper configuration files and make sure the arguments or properties are set correctly, if not, always make a backup copy first before making any changes to the file.

Most configuration issues can be resolved through the WebSphere Application Server administrative console.

Server stop failures

- Servers fail to respond to a stop command from the administrative console or the command line.
- If security is enabled, you must supply both the **-user** and **-password** arguments for these commands:
  - **stopServer**
  - **stopNode**
  - **stopManager**
- If using user ID and password in the **soap.client.props** file, verify that the login information is valid for the active user registry.
- Servers can be hung and will not respond to a stop command.
  - Check SystemOut.log files for messages from the ThreadMonitor such as

- ThreadMonitor W WSVR0605W: Thread "SoapConnectorThreadPool : 6" (00000032) has been active for 607027 milliseconds and can be hung. **There is/are 10 threads in total in the server that may be hung**

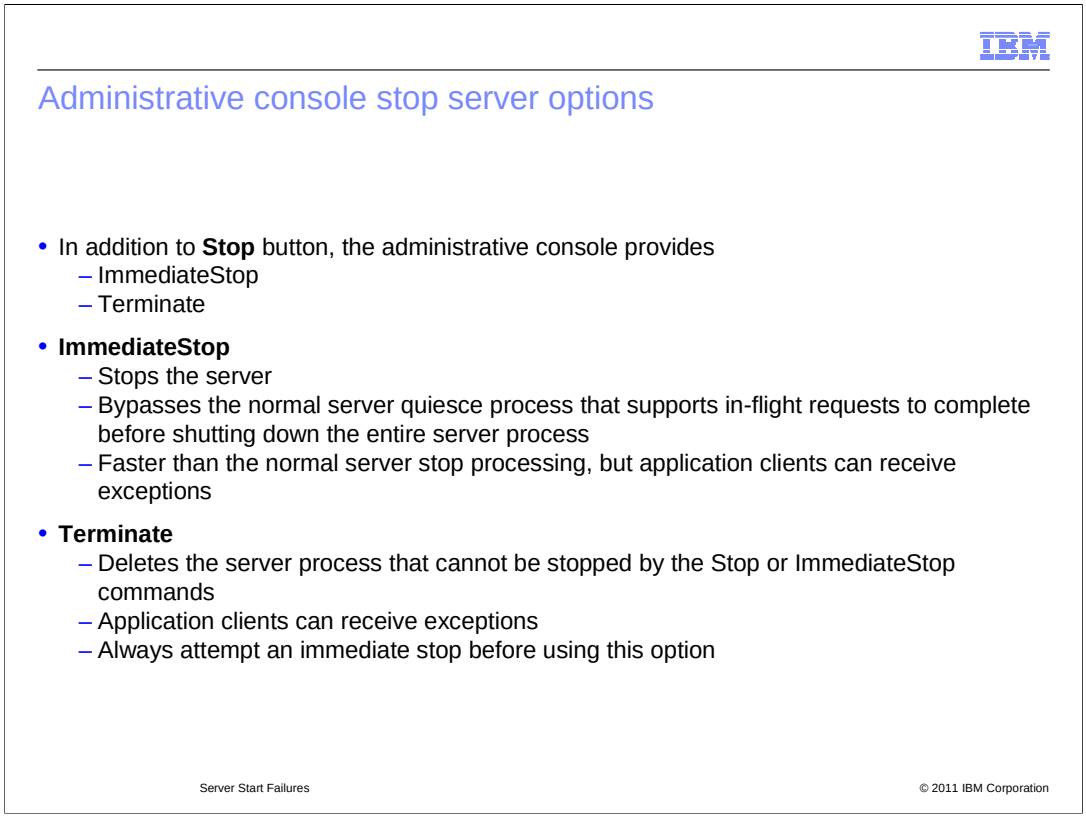
Server Start Failures

Whether you are using the WebSphere administration console or command line to stop the server, you can run into failures. Here are some common causes for stop server failures.

If you have security enabled, and you forget to pass your “user” and “password” ID when issuing the stop command, your server will fail to stop.

One way around having to pass your “user” and “password” ID every time you attempt to stop the server, is to update a few files in the properties directory under WAS_HOME, but if your ID information is incorrect, you will see failures when you try to stop the server.

Also, if you have threads (processes) in a hung state, this can cause your server to become unresponsive, in which if a stop server command was issued, it cannot be received and executed.



In the WebSphere administration console there are options to initiate an immediate stop and a termination of the process.

The immediate stop option bypasses the normal quiesce process that assures that in-flight requests are dealt with before shutting down the server. As a result, clients can experience exceptions.

The terminate option removes the server process completely. This is an abrupt termination of the server process and clients will most likely receive exceptions as well. Always attempt an immediate stop before using the terminate option.

Stop server by killing the Java process

- If an application server is hung, you can stop it by killing the Java process.
  - First try triggering a thread dump — the javacore file will contain useful diagnostic data.
  - You can get the process ID of the server from startServer.log or the <server_name>.pid file in the logs directory for the server.
  - In a Windows environment, use the Task Manager to stop the task.
  - In a UNIX environment, use the **kill -9** command to kill the Java process.
- After you have killed the hung Java process, you need to restart the server.
- Stopping a hung process is a workaround. The real problem is the hang.
  - Examine any heap dumps and Javacore files

Server Start Failures

If necessary, it is possible to forcefully end a server process.

For example, if the server is hung and will not respond to any commands, the server can be killed according the platform being used. On a UNIX-based systems this means using the “kill -9” command from the command line and on Windows platforms this means utilizing the Task Manager to identify and terminate the server process.

In the case of the hung server, it’s best to obtain a thread dump before terminating the server to assist in troubleshooting the problem. (depending on what OS platform your server is running on, a wsadmin command or a kill -3 on the process ID that is hung, will create a thread dump)

After you have killed the Java process, you will need to restart the server, and then analyze the thread dump or heap dump using one of the tools provided by IBM Support Assistant workbench.

Restarting an application server in recovery mode

- When an application server instance with active transactions in progress restarts after a failure, the transaction service can use recovery logs to complete the recovery process.
- If a catastrophic failure occurs that leaves **InDoubt** transactions:
  - Issue the command: **startServer server_name -recovery**
  - You must issue the command from the ***<profile_root>*/bin** directory of the server.
- The application server restarts in recovery mode.
  - Performs transactional recovery
  - And shuts down
- Any resource locks that the application server held before the failure are released.

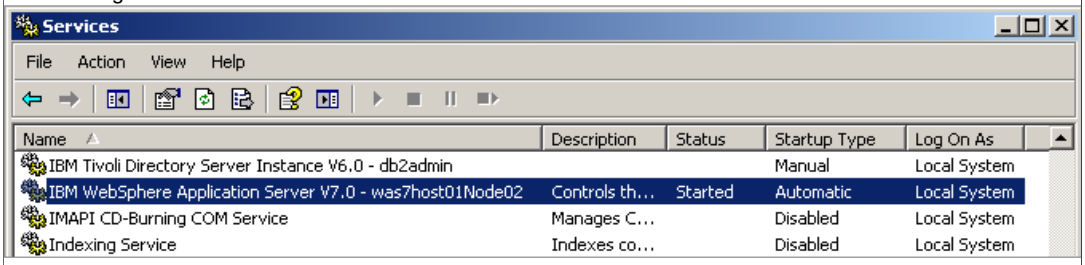
Server Start Failures

When an application server instance with active transactions in progress restarts after a failure, the transaction service uses recovery logs to complete the recovery process. These logs, which each transactional resource maintains, are used to rerun any In-Doubt transactions and return the overall system to a self-consistent state.

When you restart an application server in recovery mode, the transactional resources complete the actions in their recovery logs and then shut down. This action frees up any resource locks that the application server held before the failure.

Windows services: monitoring server processes

- Windows services
  - Optionally, created during defining new profile for stand-alone node, managed node, and deployment manager
  - **addNode** command has option to add node agent services
  - Automatic or manual start
  - Restart
  - Recovery actions
- **WASService.exe** command to create and configure services
  - User must belong to Administrator group
  - User must have advanced rights
    - Act as part of the operating system
    - Log on as a service



On a Windows machine WASService.exe command allows you to create, configure and register WebSphere services (such as AppServer, Node agent process) so the operating system can monitor and if needed restart a WebSphere process.

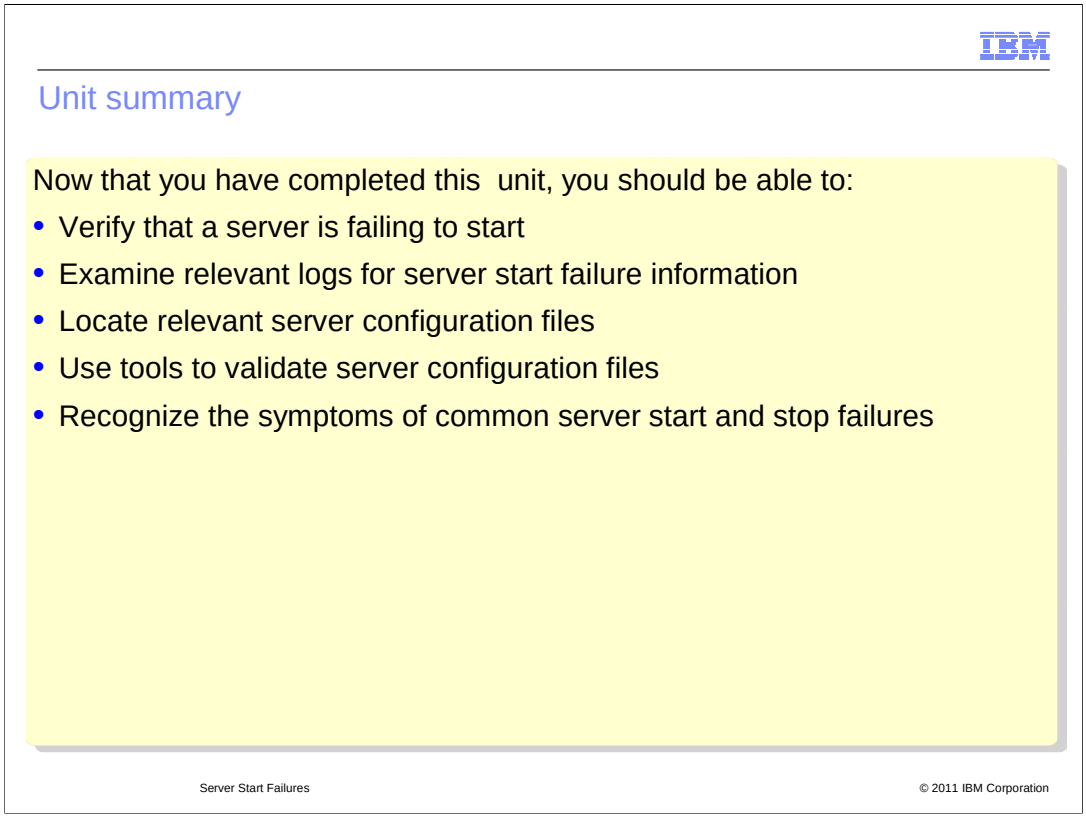
UNIX: Automatic server monitoring and restart

- On a Linux or supported UNIX operating system, you can set up a shell script to automatically monitor and restart server processes.
  - Locate the **rc.was** example shell script, which is in the ***<was_root>*/bin** directory.
  - Create a new shell script for each process that the operating system is to monitor and restart.
  - Edit each shell script according to comments in its header, which provide instructions for identifying a product process.
- Edit the **inittab** file of the operating system, to add an entry for each shell script you have created.
  - Comments in the header of the rc.was file include a sample inittab entry line for adding this script to the inittab table.
- Each inittab entry causes the operating system to call the specified shell script whenever the system initializes.
- As each shell script runs, it monitors and starts the server process you specified

You can also setup a server monitoring and restart server processes on a Linux or supported UNIX operating system.

There is a sample script in the WAS_ROOT/bin directory, it can be used as a guide to setup your own scripts. You will need to create a new shell script for each process that the operating system is to monitor and restart.

Edit the inittab file of the operating system, to add an entry for each shell script you have created. As each shell script runs, it monitors and starts the server process you specified.(must have root authority to edit the inittab file)



Now that you have completed this unit, you should be able to verify that a server is failing to start, examine relevant logs for server start failure information, locate relevant server configuration files, and use tools to validate server configuration files.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_ServerStartFailures.PPT

This module is also available in PDF format at: ../ServerStartFailures.pdf

Server Start Failures

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, disclaimer, and copyright information

IBM, the IBM logo, ibm.com, iSeries, and WebSphere are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of other IBM trademarks is available on the web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. Windows, and the Windows logo are registered 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.

Java, and all Java-based trademarks and logos are trademarks of Oracle and/or its affiliates.

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

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS OR SOFTWARE.