Servers, nodes and node agents, cells and the deployment manager are fundamental concepts in the administrative universe of the product. It is also important to understand the various processes in the administrative topology and the operating environment in which they apply.

For more information, refer to “Welcome to basic administrative architecture” on page 6.

- **Scripting**

  The WebSphere administrative (wsadmin) scripting program is a powerful, non-graphical command interpreter environment enabling you to run administrative operations in a scripting language. You can also submit scripting language programs to run. The wsadmin tool is intended for production environments and unattended operations.

  For more information, refer to “Introduction: Administrative scripting (wsadmin).”
- **Commands**

  Command-line tools are simple programs that you run from an operating system command-line prompt to perform specific tasks, as opposed to general purpose administration. Using the tools, you can start and stop application servers, check server status, add or remove nodes, and complete similar tasks.

  For more information, refer to “Introduction: Administrative commands” on page 5.
- **Programming**

  The product supports a Java programming interface for developing administrative programs. All of the administrative tools supplied with the product are written according to the API, which is based on the industry standard Java Management Extensions (JMX) specification.

  For more information, refer to “Introduction: Administrative programs” on page 5.
- **Data**

  Product configuration data resides in XML files that are manipulated by the previously-mentioned administrative tools.

  For more information, refer to “Introduction: Administrative configuration data” on page 6.

Introduction: Administrative console

The administrative console is a graphical interface for performing deployment and system administration tasks. It runs in your Web browser. Your actions in the console modify a set of XML configuration files.

You can use the console to perform tasks such as:

- Add, delete, start, and stop application servers
- Deploy new applications to a server
- Start and stop existing applications, 7 certain configurations
- Add and delete Java 2 Platform, Enterprise Edition (J2EE) resource providers for applications that require data access, mail, URLs, and so on
- Manage variables, shared libraries, and other configurations that can span multiple application servers
- Configure product security, including access to the administrative console
- Collect data for performance and troubleshooting purposes
- Find the product version information. It is located on the front page of the console.

See the *Using the administrative clients* PDF for information on how you begin using the console. See also the **Reference > Administrator > Settings** section of the Information Center navigation. It lists the settings or properties you can configure.

Introduction: Administrative scripting (wsadmin)

The WebSphere administrative (wsadmin) scripting program is a powerful, non-graphical command interpreter environment enabling you to run administrative operations in a scripting language. The wsadmin tool is intended for production environments and unattended operations. You can use the wsadmin tool to perform the same tasks that you can perform using the administrative console.

- A CORBA server
- A Remote Method Invocation (RMI) server

After you define a generic server, you can use the Application Server administrative console to start, stop, and monitor the associated non-WebSphere server or process when stopping or starting the applications that rely on them.

For more information, refer to “Creating generic servers” on page 219.

Introduction: Web servers

7

In the WebSphere Application Server product, an application server works with a Web server to handle requests for dynamic content, such as servlets, from Web applications. Go to http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html for the most current information about supported Web servers.

The application server and Web server communicate using “Web server plug-ins” on page 132. “Communicating with Web servers” on page 116 describes how to set up your Web server and Web server plug-in environment and how to create a Web server definition. The Web server definition associates a Web server with a previously defined managed or unmanaged node. After you define the Web server to a node, you can use the administrative console to perform the following functions for that Web server.

If the Web server is defined to a managed node, you can:

- Check the status of the Web server
- Generate a plug-in configuration file for that Web server.
- Propagate the plug-in configuration file after it is generated.

If the Web server is an IBM HTTP Server (IHS) and the IHS Administration server is installed and properly configured, you can also:

- Display the IBM HTTP Server Error log (error.log) and Access log (access.log) files.
- Start and stop the server.
- Display and edit the IBM HTTP Server configuration file (httpd.conf).

If the Web server it is defined to an unmanaged node, you can:

- Check the status of the Web server
- Generate a plug-in configuration file for that Web server.

If the Web server is an IBM HTTP Server (IHS) and the IHS Administration server is installed and properly configured, you can also:

- Display the IBM HTTP Server Error log (error.log) and Access log (access.log) files.
- Start and stop the server.
- Display and edit the IBM HTTP Server configuration file (httpd.conf).
- Propagate the plug-in configuration file after it is generated.

You can not propagate an updated plug-in configuration file to a non-IHS Web server that is defined to an unmanaged node. You must manually install an updated plug-in configuration file to a Web server that is defined to an unmanaged node. Web servers defined to an unmanaged node are typically remote Web servers. Remote Web servers are Web servers that are not located on the same machine as the WebSphere Application Server.

After you set up your Web server and Web server plug-in, whenever you deploy a Web application, you must specify a Web server as the deployment target that serves as a router for requests to the Web application. The configuration settings in the plug-in configuration file (plugin-cfg.xml) for each Web server

    2. Select the normal inbound chain for serving requests. This will usually be named WCInboundDefault, on port 9080.
    3. Click **TCP Inbound Channel (TCP_2)**.
    4. Set **Thread Pools** under Related Items.
    5. Select **WebContainer**.
    6. Enter values for **Minimum Size** and **Maximum Size**.
  - The HTTP 1.1 protocol provides a ″keep-alive″ feature to enable the TCP connection between HTTP clients and the server to remain open between requests. By default WebSphere Application Server will close a given client connection after a number of requests or a timeout period. After a connection is closed, it will be recreated if the client issues another request. Early closure of connections can reduce performance. Enter a value for the maximum number of persistent requests to (keep-alive) to specify the number of requests that are allowed on a single HTTP connection. Enter a value for persistent timeouts to specify the amount of time, in seconds, that the HTTP transport channel allows a socket to remain idle between requests. To specify values for Maximum persistent requests and Persistent timeout:
    1. Click **Servers > Application Servers >***server_name* **Web Container Settings> Web Container > Web container transport chains**.
    2. Select the normal inbound chain for serving requests. This will usually be named WCInboundDefault, on port 9080.
    3. Click **HTTP Inbound Channel (HTTP_2)**.
    4. Enter values for **Maximum persistent requests** and **Persistent timeout**.
- **Tune the EJB container.** An EJB container is automatically created when you create an application server. After the EJB container is deployed, you can use the following parameters to make adjustments that improve performance.
  - Set the **Cleanup interval** and the **Cache size** as described in “EJB cache settings” on page 1033.
  - **Break CMP enterprise beans into several enterprise bean modules** while assembling EJB modules. See ″Assembling EJB modules″ in the information center for more information.

  See also “EJB method Invocation Queuing” on page 1014.
- **Tune the session management.** The installed default settings for session management are optimal for performance. See “Tuning session management” on page 987 and “Tuning parameter settings” on page 988 for more information about tuning session management.
- **Tune the data sources.** A data source is used to access data from the database. The following parameters reveal how the number of physical connections within a connection pool can change performance.
  - Review information on “Connection pooling” on page 1336.
  - Set the **Maximum connection pool** and **Minimum connection pool** as described in “Connection pool settings” on page 1400.
  - Set the **Statement cache size** as described in “Data source settings” on page 1372.

Web services client to Web container optimized communication

7 To improve performance, there is an optimized communication path between a Web services client application and a Web container that are located in the same application server process. Requests from the Web services client that are normally sent to the Web container using a network connection are delivered directly to the Web container using an optimized local path. The local path is available because the Web services client application and the Web container are running in the same process.

This direct communication eliminates the need for clients and web containers that are in the same process to communicate over the network. For example, a Web services client might be running in an application server. Instead of accessing the network to communicate with the Web container, the Web services client can communicate with the Web container using the optimized local path. This optimized local path improves the performance of the application server by allowing Web services clients and Web containers to communicate without using network transports.

1.1 Overview of WebSphere Portlet Factory

7 WebSphere Portlet Factory is a powerful and flexible tool for rapidly building portlets on top of a service-oriented architecture. Developers are able to quickly and easily leverage their company's core assets, automatically assembling them into custom, high-value portlets. Portlets created with WebSphere Portlet Factory are dynamic, robust Java™ 2 Enterprise Edition (J2EE) applications that react automatically to change. They can be further modified by business users in real time to meet changing business requirements, without requiring any coding, or duplicating and versioning of assets. By eliminating the need to code all of these implementations and their variations, WebSphere Portlet Factory simplifies the development, deployment, and change management process, saving companies time and money.

1.2 Capabilities of WebSphere Portlet Factory

The primary features and capabilities of WebSphere Portlet Factory are as follows:

- **Multi-page custom portlets without coding**

  The WebSphere Portlet Factory rapid application development (RAD) capabilities and ease of use enable developers of all skill sets to create multi-page, complex portlets. Developers build portlets by defining a sequence of highly adaptive software components, called *builders*, which have an easy-to-use interface. Developers assemble builders into *models*, which then generate the application code. In this way, developers can capture and automate the process of building dynamic portlets, instead of explicitly coding each portlet.

- **Robust integration capabilities with enterprise applications and data**

  WebSphere Portlet Factory provides automatic integration with existing applications and data, including SAP®, Lotus Domino, PeopleSoft®, Siebel®, Web Services, relational databases, and Excel®. Developers can quickly create composite, high-value portlets that leverage existing investment in your existing applications and data.

- **Automatic integration with WebSphere Portal**

  With WebSphere Portlet Factory, you have direct integration with IBM WebSphere Portal features such as portlet wiring, Click-to-Action, business user configuration, people awareness, WebSphere Portal groups, and the credential vault. Portlets are deployed automatically to WebSphere Portal software.