

IBM WebSphere Application Server V4.0

# HTTP Server Plug-in



# Agenda



- HTTP Server Plugin
  - ▶ Overview and Functions
- WebSphere V4.0 Plugin
  - ▶ Overview and Features
  - ▶ Configuration of Plugin and Application Server
- Separating the HTTP Server from Application Server
  - ▶ Default WebSphere 4.0 Installation and Configuration
  - ▶ Extended Configuration
    - New Application Server
    - Secure Transport between Plugin and Application Server
- Problem Determination
  - ▶ Hints and Tips



## HTTP Server Plugin



- Enables communications between the HTTP server and the WebSphere Application Server
- Primary plug-in functions
  - ▶ Passing requests between HTTP server and WebSphere
  - ▶ Determining which requests are to be handled by the HTTP server, and which are to be passed to WebSphere
  - ▶ Callbacks and error handling
- Allows separating HTTP Server from WebSphere Application Servers in distributed configurations
  - ▶ HTTP server and Application Server located on different machines



- ▶ The HTTP Server plug-in is a component that handles all communications between the HTTP Server and the WebSphere application servers.
- ▶ It can be configured so that a specific subset of all the HTTP requests are actually forwarded to WebSphere, allowing other requests to be handled by the Web server.
- ▶ The plug-in enables you to locate the HTTP Server and the WebSphere application servers on different systems.

## Options for Separating the HTTP Server

### ■ WebSphere options for separating HTTP server from Application Server

#### ▶ HTTP Transport

- Industry standard protocol
- Non-secure transport (HTTP) and Secure transport (HTTPS)

**New in WebSphere V4.0**

#### ▶ OSE Transport

- Proprietary protocol
- non-Secure transport

**Removed in WebSphere V4.0**

#### ▶ Servlet Redirector

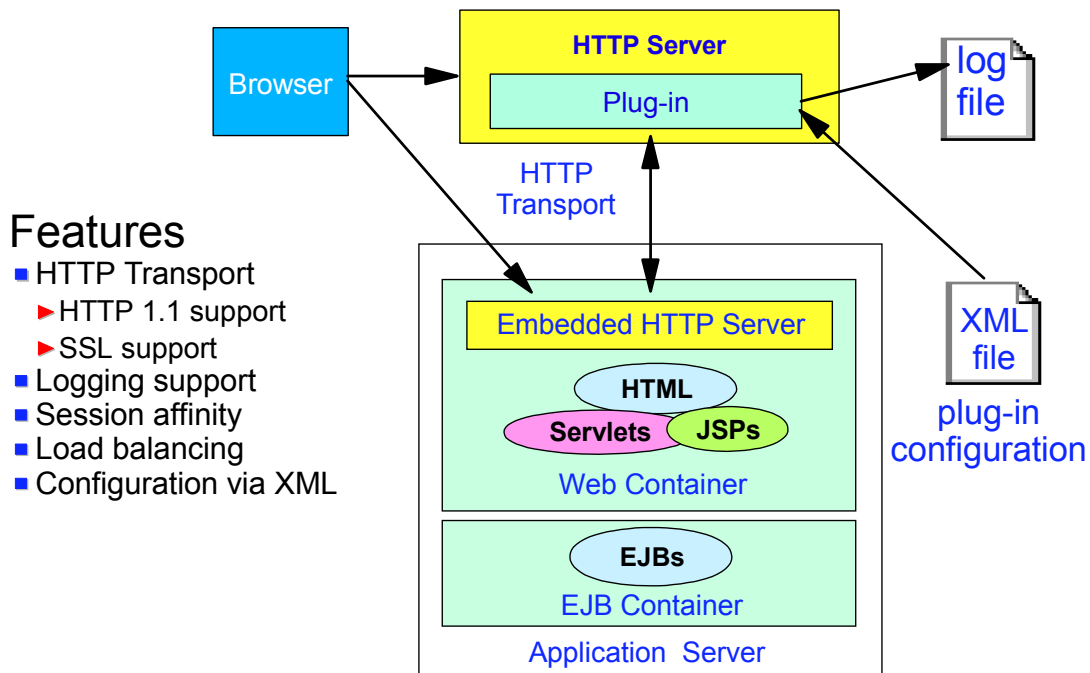
- IIOP (Internet InterOrb Protocol)
- Encrypted flows

**Removed in WebSphere V4.0**



- ▶ In all editions of WebSphere V4.0 the WebSphere plug-in can talk to the application server via HTTP or HTTPS. This new feature makes OSE obsolete and removes the need for the Servlet Redirector. In the past, the only good reason for using the Servlet Redirector was the need for a secure connection between the plug-in and the application server. In V4.0, this need is satisfied by using the HTTPS protocol.
- ▶ In V3.5.x, the plug-in communicated with the application server either through the OSE transport or via the Servlet Redirector, based on the IIOP protocol.
- ▶ The OSE protocol has been completely removed in V4.0
- ▶ The Servlet Redirector has been completely removed in V4.0

## Overview of HTTP Transport Plug-in



### Features

- HTTP Transport
  - ▶ HTTP 1.1 support
  - ▶ SSL support
- Logging support
- Session affinity
- Load balancing
- Configuration via XML

IBM®

- ▶ The new WebSphere V4.0 plug-in fully supports the 1.1 level of the HTTP protocol. It allows for straight HTTP communication with the application server or for secure HTTP connections via SSL. It still provides rich logging facilities.
- ▶ The configuration information is now contained in a XML files that play the same role as the V3.5 vhosts, rules, queues, and bootstrap properties files.
- ▶ In order to enable the HTTP transport between the plug-in and the application server, every WebSphere application server Web container has a built-in HTTP server. Each container HTTP Transport port must be unique. The ports start at 9080 and increment from there. However, ports can be manually configured in the administrative console.
- ▶ An HTTP request could actually be sent directly to and processed by the internal HTTP server. This practice is not recommended in a production environment where you need the full capabilities of the external HTTP Server (workload management, caching, etc.).

## Support for HTTP Transport Plug-in



- Supported HTTP server products
  - ▶ Apache
  - ▶ Microsoft IIS
  - ▶ IBM HTTP Server (IHS)
    - Shipped with WebSphere V4.0
  - ▶ iPlanet
  - ▶ Lotus Domino
  
- Installation of WebSphere V4.0
  - ▶ Automatic configuration of HTTP server
    - ◆ Except for Lotus Domino
    - HTTP server plug-in module
    - Plug-in configuration XML file
      - ◆ Contains configuration information similar to OSE temp directory  
e.g., vhosts.properties, queues.properties, rules.properties, bootstrap.properties
  - ▶ Initial configuration with Default Server (Application Server)
    - Pre-configured to communicate with plug-in
    - Uses port 9080 for communication



- ▶ The HTTP Server is automatically configured upon installation of WebSphere, with the exception of the Lotus Domino Web server.
- ▶ The configuration information is contained in an XML file that contains information that is equivalent to the information contained in the OSE configuration files.
- ▶ Upon creation, an application server Web container is assigned an available port for HTTP communications with the plug-in.

## Plug-in Request Processing

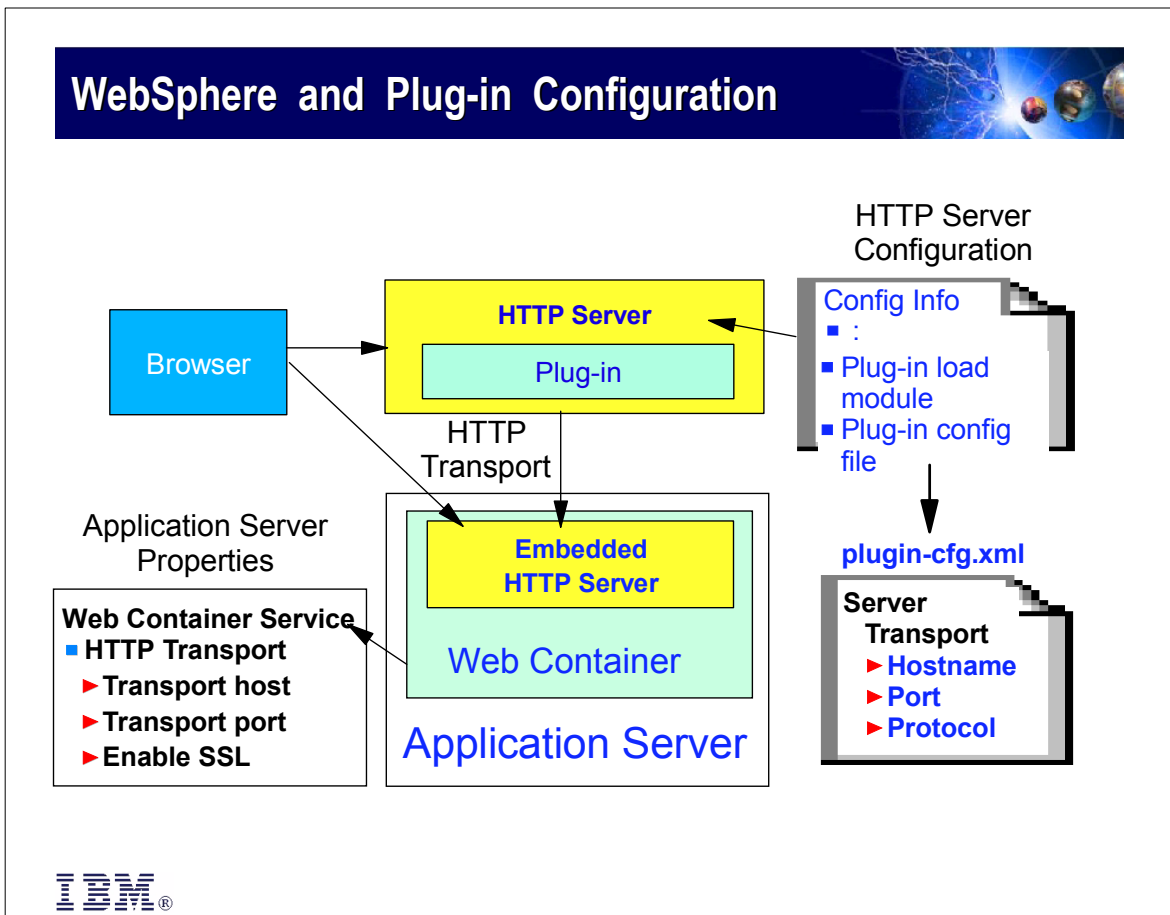


- Determine Server Group
  - ▶ Compare incoming URI and hostname to defined routes
- Determine Application Server within Server Group
  - ▶ Session affinity
  - ▶ Load balancing algorithms
- Determine Transport flow
  - ▶ If only one Web container transport is configured (HTTP or HTTPS), send request using that transport protocol
  - ▶ If both Web container transports are configured, match protocol to transport
- Send request to Application Server and read response
  - ▶ I/O is handled by embedded HTTP Server in Web Container
- Handle callbacks and send response to HTTP server
  - ▶ Return control to HTTP server



- ▶ The first step is about determining which server group the HTTP request needs to be sent to. Typically, the URI determines which server group will process the request.
- ▶ Secondly, if cloning is present, the appropriate load-balancing algorithm is used to determine which application server will process the request.
- ▶ The transport (encrypted vs. unencrypted) is selected based on the configuration of the Web container and the type of request.
- ▶ The request is sent to the embedded HTTP server and the plug-in waits for a response. Housekeeping and error handling are performed, if necessary, before control is given back to the HTTP server.

# WebSphere and Plug-in Configuration



Three areas need to be configured for the WebSphere plug-in to work correctly:

1. The HTTP Server: Business as usual, the plug-in load module and configuration files need to be specified, similarly to what used to happen in V3.5.x.
2. The plugin-cfg.xml file contains the plug-in configuration parameters.
3. The application servers need to be configured through the WebSphere console. The host names, the ports, and whether SSL is enabled or not needs to be specified here.

## HTTP Transport Plug-in Configuration



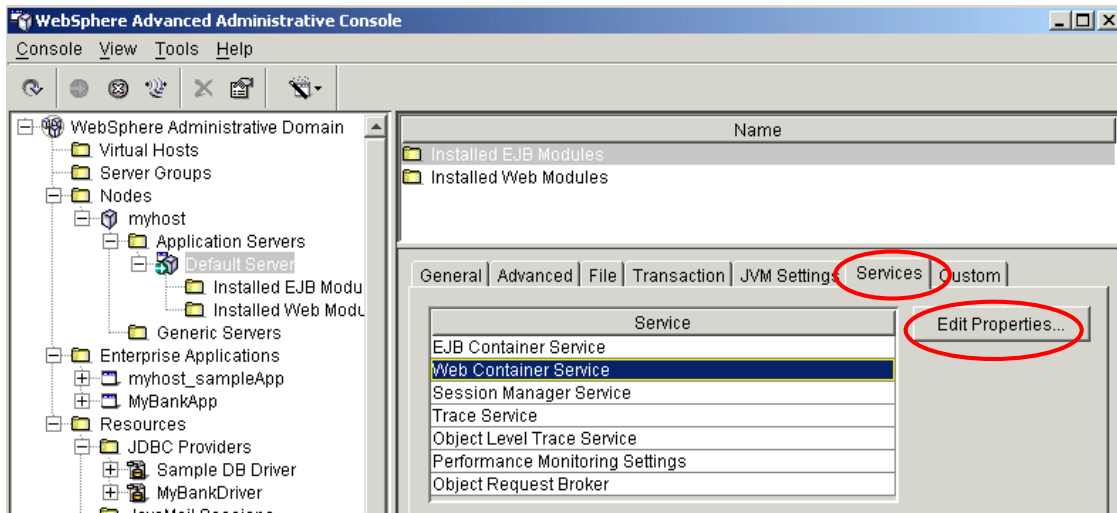
- Plug-in configuration information is contained in XML
  - ▶ Located in \$(WAS\_ROOT)/config/plugin-cfg.xml
  
- plugin-cfg.xml file is loaded and parsed by Plug-in
  - ▶ During HTTP Server initialization
  - ▶ During runtime when changes are made
  
- Procedure for HTTP Transport Plug-in configuration
  - ▶ Using the Administrative Console
    - Create new or modify an existing Application Server
    - Modify HTTP Transport properties for Web Container Service if necessary
  - ▶ Regenerate plugin-cfg.xml file
  - ▶ Or manually edit the plugin-cfg.xml file (not recommended)



- ▶ The plug-in configuration file is an XML file contained in the directory shown. Whenever a change to the environment configuration is made (for instance, whenever a new application server is created, or a new virtual host is created) the plug-in configuration file needs to be updated and reloaded.
  
- ▶ The information in the plug-in file must match the parameters specified in the WebSphere environment configuration (host names, port numbers, protocols)

## WebSphere Configuration for Plug-in

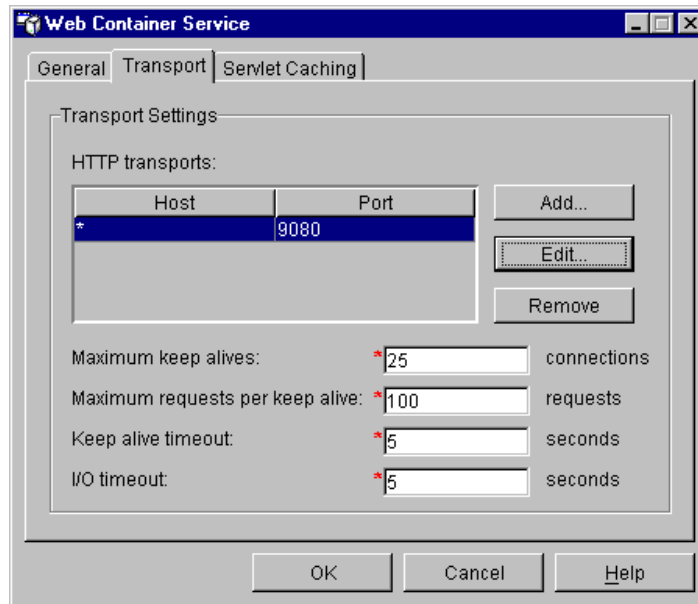
- Select Application Server, Select Services tab
  - ▶ Select Web Container Service, Edit Properties



- ▶ From a WebSphere administrator's standpoint, the Web Container needs to be correctly configured to talk to the WebSphere plug-in in the desired way.
- ▶ There is a significant change in the Admin Console for V4.0 when it comes to configuring the Web Container:
  - ▶ Open the **Application Server** properties and select the **Services** tab. You'll find a list of services, including the **Web Container Service**. Select Web Container and click **Edit Properties**. You'll obtain a dialog that allows you to
    - ▶ Select the transport mechanism (HTTP)
    - ▶ Enable SSL

## Web Container Service Properties

- Multiple HTTP transports may be configured per container
  - ▶ For example, one transport for HTTP and one for HTTPS



- ▶ You can configure multiple HTTP ports and transport (for example to differentiate between unencrypted and SSL communications)
- ▶ If more than one port of the same type is configured on a Web container (multiple HTTP ports or multiple HTTPS ports) only one port of each type will be used by the plug-in - the others will be ignored.

## HTTP Transport Properties

- Property values correspond to plugin-cfg.xml

- ▶ Hostname

- ▶ Port

- ▶ Protocol

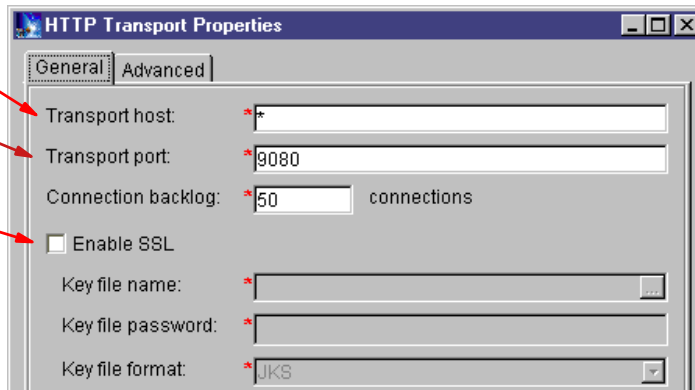
- HTTP

- HTTPS

- Enable SSL for secure transport

- ▶ Requires a key file containing digital certificate

- ▶ Key file format must be JKS



- ▶ This dialog allows the detailed configuration of the virtual host mappings and of SSL.

- ▶ This dialog allows you to specify the key file containing the digital certificate - in JKS format.

## plugin-cfg.xml File

```
<Config>
  <Log Name="$(WAS_ROOT)/logs/native.log" LogLevel="Error"/>
  <ServerGroup Name="default_group">
    <Server Name="default_server">
      <Transport Hostname="localhost" Port="9080" Protocol="http"/>
    </Server>
  </ServerGroup>
  <VirtualHostGroup Name="default_host">
    <VirtualHost Name="*.*"/>
  </VirtualHostGroup>
  <UriGroup Name="default_host_URIs">
    <Uri Name="/servlet/*"/>
    <Uri Name="/webapp/examples/*"/>
    :
    <Uri Name="/WebSphereSamples/*"/>
  </UriGroup>
  <Route ServerGroup="default_group" UriGroup="default_host_URIs"
    VirtualHostGroup="default_host"/>
</Config>
```

Default configuration  
information created  
during WebSphere  
installation



- ▶ This chart shows the initial plug-in configuration file. Notice that each URI can be separately configured. Wildcards can be used at the desired level.
- ▶ ServerGroup relates to queues.properties in V3.5, using OSE Remote protocol
- ▶ VirtualHostGroup relates to vhost.properties
- ▶ UriGroupName relates to rules.properties

## Updating the plugin-cfg.xml File



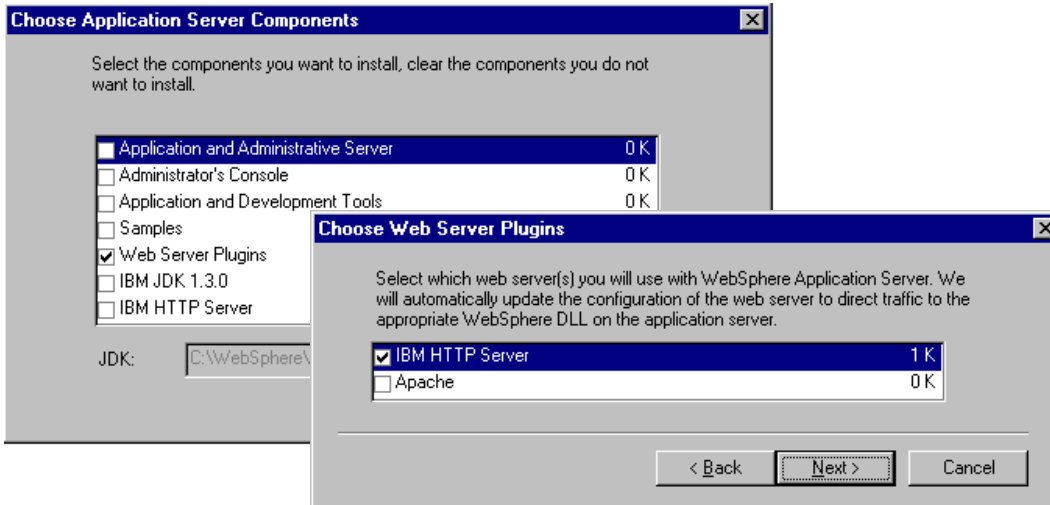
- **Modifying plugin-cfg.xml**
  - ▶ **Automatically**
    - Application Server "Custom" tab --> Service: "Automatic Generation of Plug-in Configuration." Click Edit, check Enabled, click Apply.
  - ▶ **Manually**
    - Right-Click a node, select "Regen Webserver Plugin"
    - Use GenPluginCfg.bat
    - Edit manually (not recommended)
  
- **Updating is required when changes are made to:**
  - ▶ Virtual host
  - ▶ Application Server
  - ▶ Enterprise Application (URIs)
  - ▶ Log file location and message level
  - ▶ Application Server Transport
    - [ Hostname ] [ Port ] [ Protocol ]



- ▶ In order to update the plugin-cfg.xml file, you can either use a manual approach or the GenPluginCfg script.
- ▶ Modification (or regeneration) is required in many circumstances. These circumstance also include changes to the log file location and message level.
- ▶ Automatic Generation of Plug-in Configuration defaults to False. In a production environment, the plug-in configuration should be relatively stable.

## Install Plug-in on Separate Machine

- Custom install only Web Server Plugins on HTTP Server node
  - ▶ Installs Plug-in and configures the HTTP Server



## Configure Plug-in on Separate HTTP Server

- Installation of plug-in and WebSphere components on a separate HTTP server will create a default configuration
- Edit the **plugin-cfg.xml** file and specify Hostname for WebSphere (Application Server)

```
<ServerGroup Name="default_group">  
  <Server Name="default_server">  
    <Transport Hostname="your.host.name"  
      Port="9080" Protocol="http"/>  
  </Server>  
</ServerGroup>
```

Modify *Hostname* attribute to specify fully qualified hostname of WebSphere host

**Note:** Specifies listener port for Application Server. Matches Host *Alias* entry for Virtual Host and *Transport* port value for Web Container Service.



- ▶ Once the installation is complete you will need to modify the default plug-in configuration file to match your server configuration

## Configure Plug-in for New Application Server



```
<Config>
  <Log Name="$(WAS_ROOT)/logs/native.log" LogLevel="Error"/>
  <ServerGroup Name="default_group">
    <Server Name="default_server">
      <Transport Hostname="localhost" Port="9080" Protocol="http"/>
    </Server>
    <Server Name="new_server">
      <Transport Hostname="your.host.name" Port="9081"
        Protocol="http"/>
    </Server>
  </ServerGroup>
  <VirtualHostGroup Name="default_host">
    <VirtualHost Name="*:*/>
  </VirtualHostGroup>
  <UriGroup Name="default_host_URIs">
    :
    <Uri Name="/newURI/*"/>
  </UriGroup>
  <Route ServerGroup="default_group" UriGroup="default_host_URIs"
    VirtualHostGroup="default_host"/>
</Config>
```



- ▶ If you have multiple application servers, you need to configure the additional ones in the plugin-cfg.xml file. The chart shows the relevant xml tags.

## Configure a New Application Server



- Configure Virtual Host aliases
- Create new Application Server
  - ▶ Configure Web Container Service
    - HTTP Transport
      - ◆ Transport Host
      - ◆ Transport Port
    - ▶ Create new DataSource, etc.
    - ▶ Install new Enterprise Application
- Modify the Plug-in Configuration XML File
  - ▶ Specify information for new XML elements
    - Server
      - ◆ Name
      - ◆ Transport
    - URI Names



- ▶ Additional application servers need to be configured for the HTTP/HTTPS protocol.
- ▶ You'll have to define the aliases for the virtual hosts that you want to associate with the new application server.
- ▶ After you have created the new application server, you'll need to configure the Web Container, as we already discussed. In particular, the HTTP transport needs to be configured (the associated virtual hosts need to be specified and the ports too).
- ▶ You are now ready to create your resources (data sources, etc.) and to install any Enterprise Application that you want to associate with the newly created application server.
- ▶ Finally, you'll have to modify the plugin-cfg.xml file.

## Configuring a Secure Transport



- Secure transport between Plug-in and Application Server requires digital certificates
  - ▶ Plug-in certificate
    - Create Key file in KDB format to store certificate
    - Use IBM Key Management Utility (Ikeyman) shipped with IBM HTTP Server to create KDB file
    - Certificate may be self-signed or obtained from Certificate Authority
  - ▶ Application Server certificate
    - Create Key file in JKS format to store certificate
    - Use IBM Key Management Utility (Ikeyman) shipped with WebSphere to create and/or store certificate
    - Certificate may be self-signed or obtained from Certificate Authority



## Configure Plug-in for a Secure Transport



- Configure the plugin-cfg.xml file for secure transport
  - ▶ Specify XML elements attributed for Protocol and key file
    - Protocol = HTTPS
    - Property name and value for key file
    - Property name and value for stash file

```
<Config>
:
<Server Name="new_server">
  <Transport Hostname="your.host.name" Port="9081" Protocol="HTTPS">
    <Property name="keyring"
      value="E:/WebSphere/AppServer/config/pluginKey.kdb"/>
    <Property name="stashfile"
      value="E:/WebSphere/AppServer/config/pluginStash.sth"/>
  </Transport>
</Server>
:
</Config>
```



- ▶ Digital certificates are a good way to prove the identity of a person or another object. Certificates are digital documents that contain the information about the person or object that owns the certificate. A certificate also contains a public key that can be used for encrypting or decrypting messages. The basic idea behind a certificate is that a trusted authority, called a Certificate Authority (CA), has once identified you and proves that you are who you claim to be. The public key contained in such a certificate can be used to encrypt information that is only intended for the certificate owner because the certificate owner is in possession of the secret private-key that is required to decrypt the messages encrypted with his public-key. The certificate and the public key are freely-distributed information. The certificate also contains the digital signature of the signer Certificate Authority, which protects the certificate from tampering.

# Configure WebSphere for a Secure Transport



## ■ Configure the Web Container Service

### ▶ HTTP Transport Properties

#### ● Enable SSL

- ◆ Fully qualified key file name, key file password, and key file format (JKS)

HTTP Transport Properties

General | Advanced

Transport host: \*

Transport port: \* 9081

Connection backlog: \* 50 connections

Enable SSL  Use global SSL default configuration

Key file name: \* C:\WebSphere\AppServer\config\javakey.jks

Key file password: \* \*\*\*\*\*

Confirm password: \* \*\*\*\*\*

Key file format: \* JKS



## Plug-in Problem Determination



### ■ Hints and Tips

- ▶ Analyze the plug-in log for problem messages
  - Log name and location correctly configured in plugin-cfg.xml
  - Settings for LogLevel= Error | Warn | Trace
    - ◆ LogLevel=Error is default value
    - ◆ LogLevel=Trace provides most Problem Determination information
    - ◆ Log file grows quickly
    - ◆ Set LogLevel=Error under normal situations
  - Log file may contain various Return Codes
    - ◆ See Appendix for Return Codes and descriptions
- ▶ Use the embedded HTTP Server to test application
  - e.g., `http://your.hostname.com:<9080>/yourAppURI`
- ▶ Changes made to plugin-cfg.xml are dynamic
  - Switch to LogLevel=Trace for plug-in Problem Determination
- ▶ Application Server stdout and stderr files
  - Specify fully qualified file names
  - Check for application-related problems during initialization or execution



IBM WebSphere Application Server V4.0

# HTTP Server Plug-in Appendix



## HTTP Server Plug-in: Appendix



- Complete default plugin-cfg.xml file
- Configuration information for plugin-Cfg.xml
- Plug-in Return Codes and descriptions



## plugin-cfg.xml File



The plugin-cfg.xml file is located in:

**\$(WAS\_ROOT)/config/plugin-cfg.xml**

The following information represents the complete (default) plugin-cfg.xml file.

```
<Config>
  <Log Name="$(WAS_ROOT)/logs/native.log" LogLevel="Error"/>
  <ServerGroup Name="default_group">
    <Server Name="default_server">
      <Transport Hostname="localhost" Port="9080" Protocol="http"/>
    </Server>
  </ServerGroup>

  <VirtualHostGroup Name="default_host">
    <VirtualHost Name="*:*/>
  </VirtualHostGroup>

  <UriGroup Name="default_host_URIs">
    <Uri Name="/servlet/*"/>
    <Uri Name="/webapp/examples/*"/>
    <Uri Name="*.jsp"/>
    <Uri Name="/ErrorReporter"/>
    <Uri Name="/j_security_check"/>
    <Uri Name="/theme"/>
    <Uri Name="/WebSphereSamples/*"/>
  </UriGroup>

  <Route ServerGroup="default_group" UriGroup="default_host_URIs"
  VirtualHostGroup="default_host"/>
</Config>
```



# Plug-in Configuration Information



## Plug-in XML Configuration Requirements

The plug-in configuration is specified in XML. The parsing of this config file is done using the Xerces parser. Validation for the config file will be turned on to ensure that the config file is proper before the parsing handler function of the plug-in begins.

**Note:** Elements are in **bold**. Attributes are in *italic*.

**Config** - The start of a WebSphere HTTP plug-in configuration file. This is the global config element and all other elements and attributes of the configuration should be contained within this element. A config file should contain only one of these elements.

**Log** - The log describes the location and level of log messages that should be written by the plug-in. A Config should contain either zero or one of these elements. If a Log is not specified within the Config then log messages should be written to the HTTP server's error log.

*Name* - The full path to the log file that the plug-in should write error messages to. A Log should contain exactly one of this attribute.

*LogLevel* - The level of detail of the log messages that the plug-in should write to the log. Values for this attribute are one of the following:

Trace, Warn, or Error. A Log should contain zero or one of these attributes. If a LogLevel is not specified with the Log then the default value is Error.

**ServerGroup** - A group of servers that are generally configured to service the same types of requests. A Config should contain one or more of these elements.

*Name* - The name to be used for this group of servers. This serves as an identifier only. A ServerGroup should contain exactly one of this attribute.

**Server** - A WebSphere Application Server instance that is configured to handle requests routed to it given the routing rules of the plug-in configuration. A ServerGroup should contain one or more of these elements.

*Name* - The name to be used for this server. This serves as an identifier only. A Server should contain exactly one of this attribute.

*SessionID* - The session id that if present in the HTTP Cookie header requires us to route to this particular server provided all other routing rules are met. A Server should contain exactly one of this attribute. If a SessionID is not specified in the Server then Session Affinity is not enabled for this server.

**Transport** - The actual transport to be used for reading and writing requests to a particular WebSphere Application Server instance. A Server should contain one or more of these elements.

*Hostname* - The machine hostname or IP address that the WebSphere Application Server instance is running on. A Transport should contain exactly one of this attribute.

*Port* - The port that the WebSphere Application Server instance is listening on. A Transport should contain exactly one of this attribute.

*Protocol* - The protocol that should be used when communicating over this transport. Values for this attribute should be either HTTP or HTTPS. A Transport should contain exactly one of this attribute.



## Plug-in Configuration Information ...



**Property** - A property to be used in conjunction with the parent element. Currently this element is only supported within a Transport. When the Protocol of the Transport is set to HTTPS, this element is used to supply the various initialization parameters such as keyfile, stashfile, etc. A Transport should contain zero, one, or more of these attributes.

*Name* - The name of the Property being defined. A Property should contain exactly one of this attribute.

*Value* - The value of the Property being defined. A Property should contain exactly one of this attribute.

**VirtualHostGroup** - A group of virtual host names that will be specified in the HTTP Host header. A Config should contain zero, one, or more of these elements.

*Name* - The name to be used for this group of virtual hosts. This serves as an identifier only. A VirtualHostGroup should contain exactly one of this attribute.

**VirtualHost** - The name used for a virtual or real machine used to determine if incoming requests should be handled by WebSphere Application Server or not. A VirtualHostGroup should contain one or more of these elements.

*Name* - The actual name that should be specified in the HTTP Host header in order to match successfully with this VirtualHost. A VirtualHost should contain exactly one of this attribute.

**UriGroup** - A group of URIs that will be specified in the HTTP request line. A Config should contain one or more of these elements.

*Name* - The name to be used for this group of URIs. This serves as an identifier only. A UriGroup should contain exactly one of this attribute.

**Uri** - The virtual path to the resource that will be serviced by WebSphere Application Server. A UriGroup should contain one or more of these elements.

*Name* - The actual string that should be specified in the HTTP request line in order to match successfully with this Uri. A Uri should contain exactly one of this attribute.

**Route** - A request routing rule that the plug-in will look at to determine if an incoming request should be handled by one of the WebSphere Application Servers. A Config should contain one or more of these attributes.

*VirtualHostGroup* - The group of virtual hosts that should be used in route determination. A Route should contain zero or one of these attributes.

*UriGroup* - The group of URIs that should be used in route determination. A Route should contain zero or one of these attributes.

*ServerGroup* - The server group that requests successfully matching this route should be sent to. A Route should contain exactly one of this attribute.



## Plug-in Return Codes



RETURN CODE	RETURN CODE DESCRIPTION
WS_HANDLED	Return code when all things behaved correctly. During the request handler phase, this return code means that the request was handled by the Application Server.
WS_DECLINED	Returned when the request is not handled by the Application Server.
WS_AS_DOWN	Return code when the plug-in has deemed a specific server is no longer responding.
WS_ALLOC_ERROR	Failure to allocate a request off of the heap. If this return code is seen, the resources on the machine are likely all used up and other major problems are going on.
WS_CONFIG_ERROR	Returned when there is an error in parsing of the configuration file.
WS_SYS_ERROR	Generic return code for something having gone wrong on the system.
WS_PARAM_ERROR	A parameter provided for request processing is invalid in some way.

