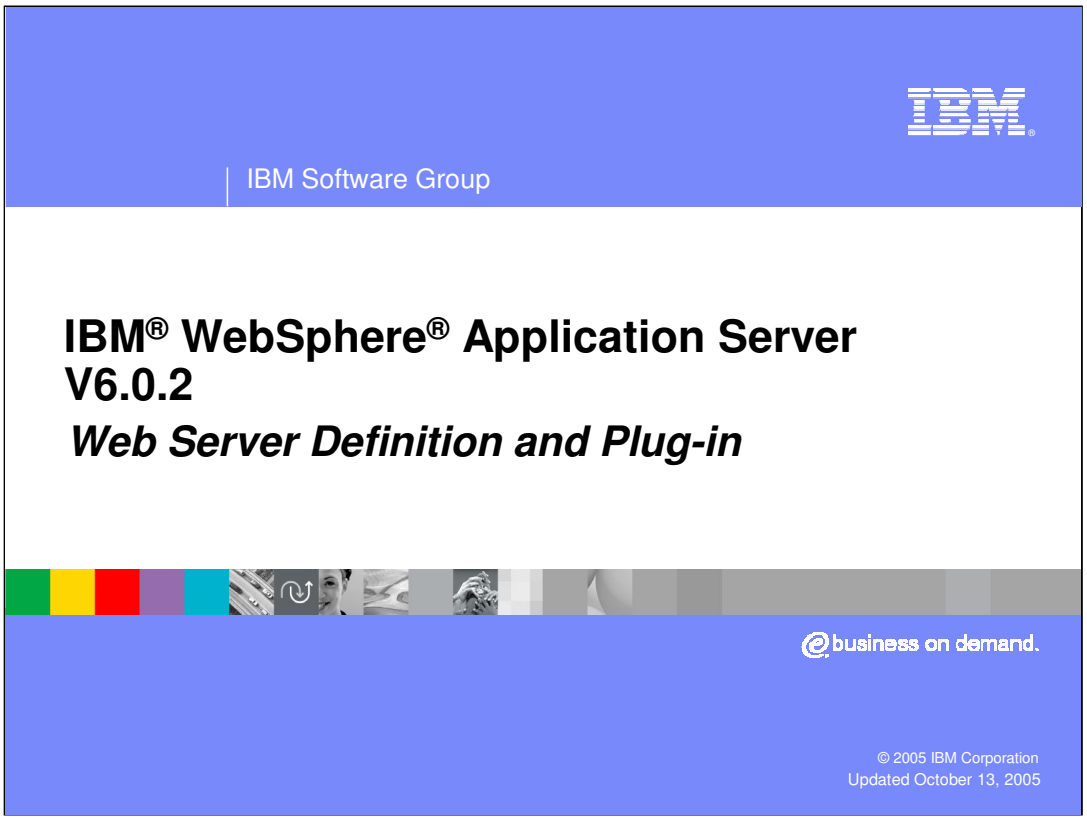


This presentation will focus on support for Web Servers in WebSphere Application Server V6.0.2.

Goals

- Understand Managed and Unmanaged nodes
- Introduce Web Server definitions in WebSphere Application Server topologies
- Understand IBM HTTP Server (IHS) V6.0.2 Administration
- Understand Plug-in file generation and propagation

The goal of this presentation is to introduce the core concepts of Web Servers in WebSphere Application Server V6.0.2. In particular this presentation will cover the new concept of managed and unmanaged nodes within a WebSphere topology. There will also be an explanation of Web Server administrative tasks.

Agenda

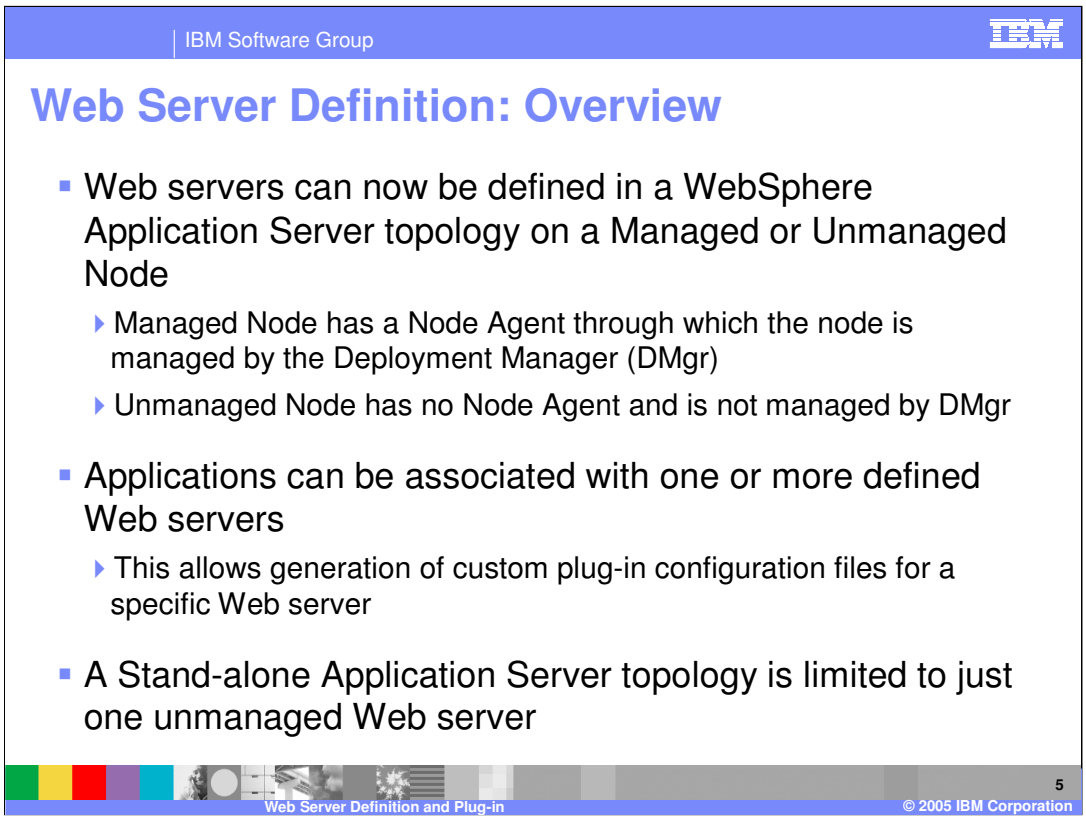
- Web Server definition overview
- Web Server definition for Stand-alone Application Server
- Web Server Definition in a V6 Network Deployment cell
  - Deployment Manager Administrative Console UI to manage Web Servers and the plug-in files
- Create and manage Web Server definitions
- IBM HTTP Server V6 management from WebSphere V6
- Plug-in file generation and propagation

The presentation will begin with an explanation of how to create a Web Server definition in an ND cell. V6 offers the option to create a managed or an unmanaged node for the purposes of managing the Web Server from within a WebSphere topology. The WebSphere administrative console provides the capability to manage both the Web Servers and their plug-in files. Next, Web Server creation in a Stand-Alone or Deployment Manager environment will be discussed as well as how to manage an IBM HTTP Server through the WebSphere Application Server administrative console. Finally, Plug-in file generation features and propagation in WebSphere Application Server V6 will be discussed.

Section

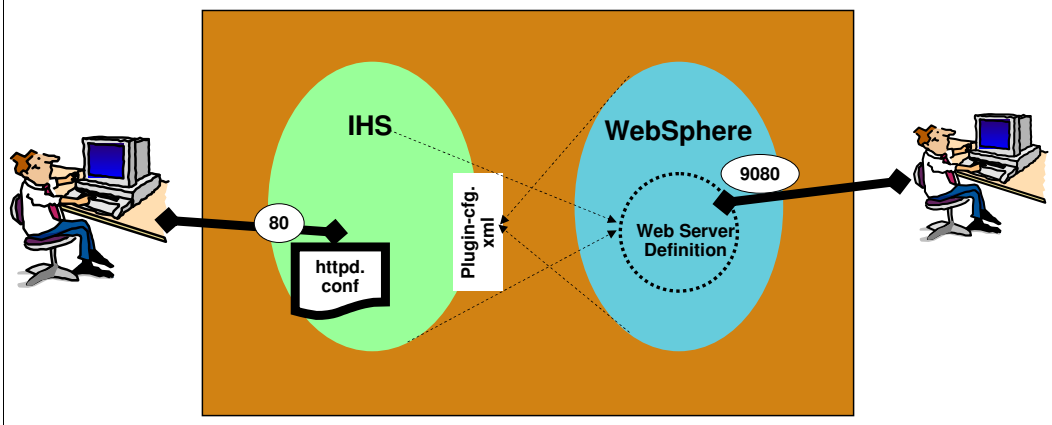
Web Server Definition - Overview

This section will cover the basics of Web Server definitions.



The ability to define a Web server configuration within a node is new in version 6. In the case of an Apache based IBM HTTP Server, the WebSphere administrative console can be used to stop and start the Web server and transparently update the plug-in configuration file on a remote system where the Apache based Web server is installed. WebSphere can also selectively target applications to specific Web servers, so that only those Web servers can route requests to the application. Web Servers can be defined on managed or unmanaged nodes. The concept of a managed node means that a node agent is running on the Web Server system. With an unmanaged node there is no node agent, which can be useful if you want to place the Web Server in a DMZ. IBM HTTP Server on z/OS platforms is based on Domino Go and not Apache, so there are some minor differences.

Web Server Definition



This graphic presents a stylized view of Web servers and their definition within the WebSphere repository.

The plugin-cfg.xml file contains directives that control the forwarding of requests between the HTTP Server and WebSphere Application Server. The plug-in file can be generated from the Administrative Console (using port 9080 in the picture) or through a special script. The plugin-cfg.xml file must be manually defined in the Web server httpd.conf file as a plug in. IBM HTTP Server (IHS) must be installed and configured as usual before the plugin-cfg.xml file definition is added to the httpd.conf file. The Web server is managed by an administrator (using port 80 as shown here).

The new administrative function provides an alternative way to administrate the Web server. It can still be controlled as before by using the Web server administrative interface or manually editing the httpd.conf, but it can now also be controlled from the WebSphere administrative console. Both the httpd.conf and the plugin-cfg.xml files can be modified from the WebSphere Administrative Console.

This picture can now be generalized beyond the stand-alone case. For example, the Web server and the application server could be on separate hosts. The web server could even be on a distributed host.

You can even have multiple definitions corresponding to multiple Web servers.

Web Server Support: At a Glance

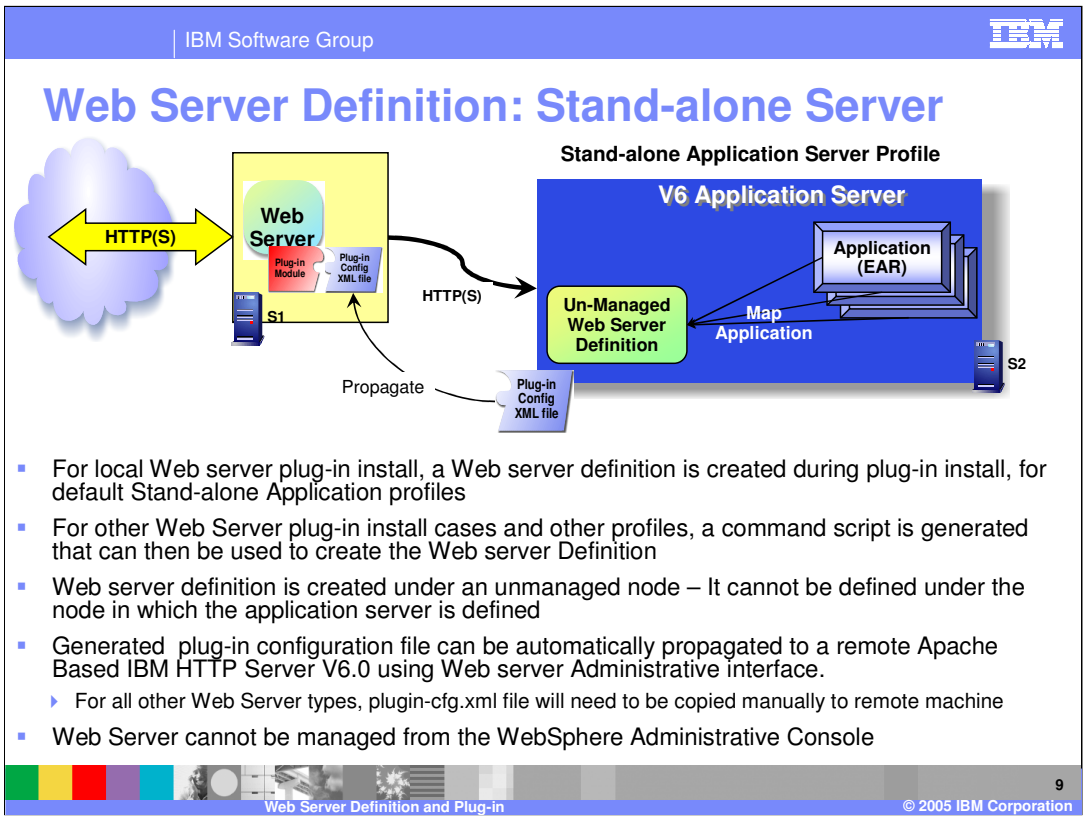
WebSphere Topology Applicability	Web Server Support	Requirement	Web Server Administration Capability
All packages - Stand-alone Application Server (ND or Express) or ND Cell	**Un-managed Web Server Node** This is same as support in WebSphere v5.x	None	None
Network Deployment (ND) Cell only	**Managed Web Server Node**	Requires Node Agent running on the Web Server machine	Auto propagation of plug-in configuration file to any remote Web Server machine If using Apache based IHS, then you can start/stop IHS
	IHS as a special case of Unmanaged Node	Requires Apache Based IHS AdminService to be running	

This slide details the support for managed and unmanaged nodes based on the version of WebSphere you use. The key point to notice here is that in a stand-alone server environment, there can only be a Web Server definition on an unmanaged node.

Section

Web Server Definition for a Stand-alone Application Server

This section will show the support for a Web Server definition in a Stand-alone application server topology.



The Stand-alone application server environment only supports the notion of a Web server in an unmanaged node, which is shown in this diagram. When the plug-in is installed on the system with the application server, a Web server definition is created within the configuration. The plug-in configuration file must be propagated to the system where the Web Server is installed. Propagation is supported for the Apache Based version of IBM HTTP Server V6.0 using IHS administration server. WebSphere V6 has the capability to map applications to specific Web Servers. However, in this case since there can be only a single Web server, all applications will be mapped to the same Web Server.

Section

Web Server Definition in a Network Deployment Cell

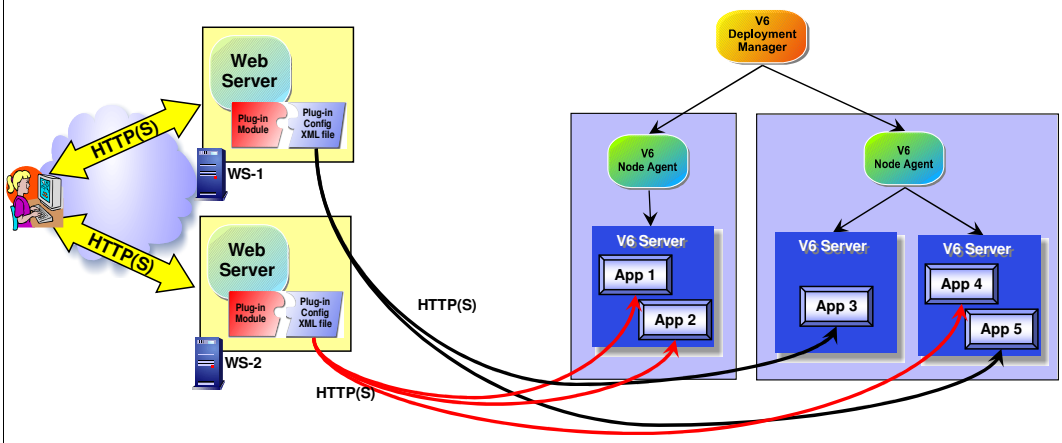
This section will detail a Web server definition in a Network Deployment setting.

Web Server Definition in a ND Cell

- Web servers are defined as Web Server nodes in Cell topology under Managed or Unmanaged Nodes
- Following administrative tasks are supported
  - Create a new Web server definition templates
  - Create new Web server definitions from pre-defined templates for different supported Web servers
  - Manage and administer the managed apache based IBM HTTP Server Web servers
    - Start, Stop, Delete, Modify Web server configuration
  - Customize Plug-in configuration (plugin-cfg.xml) file for each Web Server
    - Map applications to a Web server
    - Configure Plug-in configuration properties (like Error level, etc.)
  - Propagate Plug-in configuration files

In a Network Deployment environment, Web Servers can either be defined as managed or unmanaged nodes. As a managed node, there will be a node agent present that will allow the deployment manager to manage the Apache based Web server. As an unmanaged node or a non Apache-based Web server, there is no node agent so the deployment manager is more limited in managing the Web server. Administrative tasks allow you to create new Web Server definitions from templates that have been defined. A new Web server definition template can be created from an existing Web server definition. The administrative console also provides enhanced functionality to customize the plug-in configuration files for a Web server. The plug-in can then be automatically propagated to Apache-Based managed Web servers or manually copied to unmanaged or Domino Go Web servers.

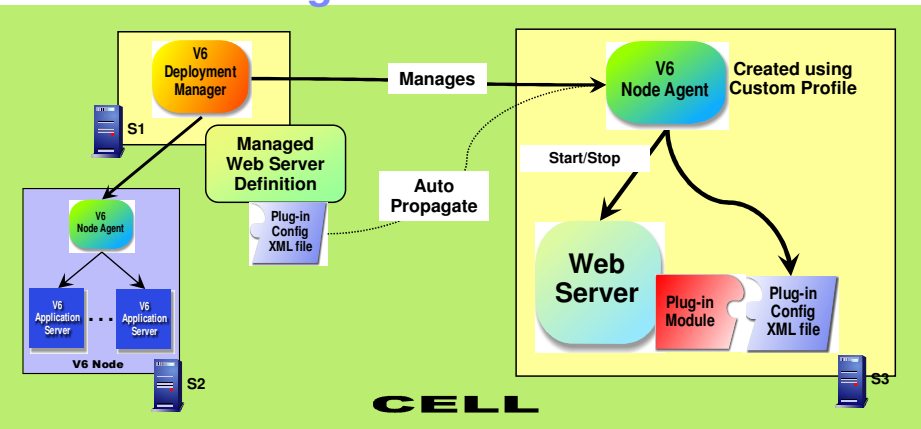
Web Server Topology: Application Centric



- Applications App-3, App-5 are mapped to Web server WS-1
- Applications App-1, App-2, App-4 are mapped to Web server WS-2
- Custom plug-in files will automatically be generated for Web servers to send requests for only their mapped applications

This slide shows how applications can be mapped to specific Web servers. These Web servers will then be responsible for handling the requests for the application that are mapped to that Web server only. These mappings result in the Deployment Manager creating the custom plug-in files. The information contained in the plug-in can also contain other customizations, such as caching and balancing features.

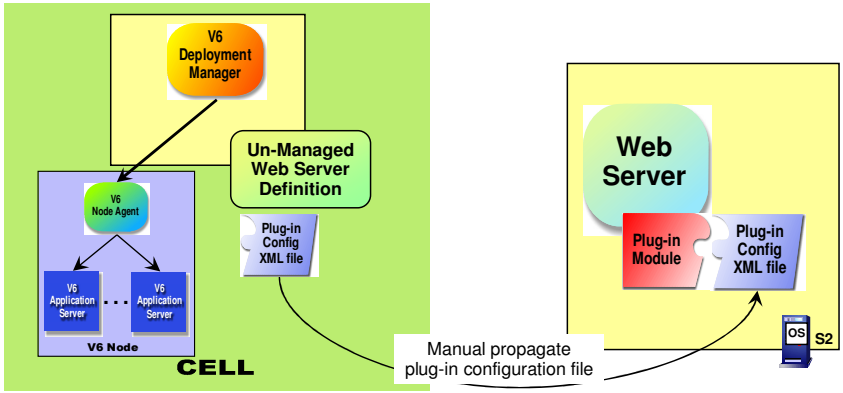
Web Server: Managed Node in a ND Cell



- Manage Web server (defined on a Managed node) from WebSphere DMgr
- Requires a WebSphere Node to be created on the Web server machine. Node Agent receives commands from DMgr to administrate the Web Server
- Provides ability to start and stop the Web Server and automatically push the plug-in configuration file to the Web Server
- Common for web servers installed behind a firewall where a WebSphere Node can be installed

This slide shows an example of a Web server in a managed node. A node agent is present on the system where the Web server is installed. The Apache-based Web server is managed by the WebSphere Deployment Manager through the node agent. The node agent provides the capability to start and stop the Web Server as well as automatically propagate the plug-in configuration file to the Web Server. This scenario is most common for a Web server installed behind a firewall where a WebSphere Node can be installed without any security concerns.

Web Server: Unmanaged Node in a ND Cell



- Web server not managed by WebSphere (defined on a unmanaged node)
- Manually propagate the plug-in configuration file from the DMgr system to the Web server system . The Apache based IBM HTTP Server allows for more management in an unmanaged node, which will be explained later in this presentation.
- Allows system administrator to create custom plug-in files for specific Web server
- Common for Web servers installed outside firewall or in DMZ, where no Node Agent is installed

In this example the Web server is defined in an unmanaged node and registered as an Unmanaged Node in WebSphere configuration. This allows a WebSphere System Administrator to create custom plug-in files for that Web server. This is covered in more detail in the plug-in presentation. When the plug-in is created it must be manually copied to the Web server. The deployment manager has no capability to direct the Web server. A Domino Go- based Web server falls into this category as well. The Apache-based IBM HTTP Server allows for more management of an unmanaged node, this will be explained later in this presentation.

Section

Creating and Managing Web Server Definitions
and
Mapping Applications to Web Servers

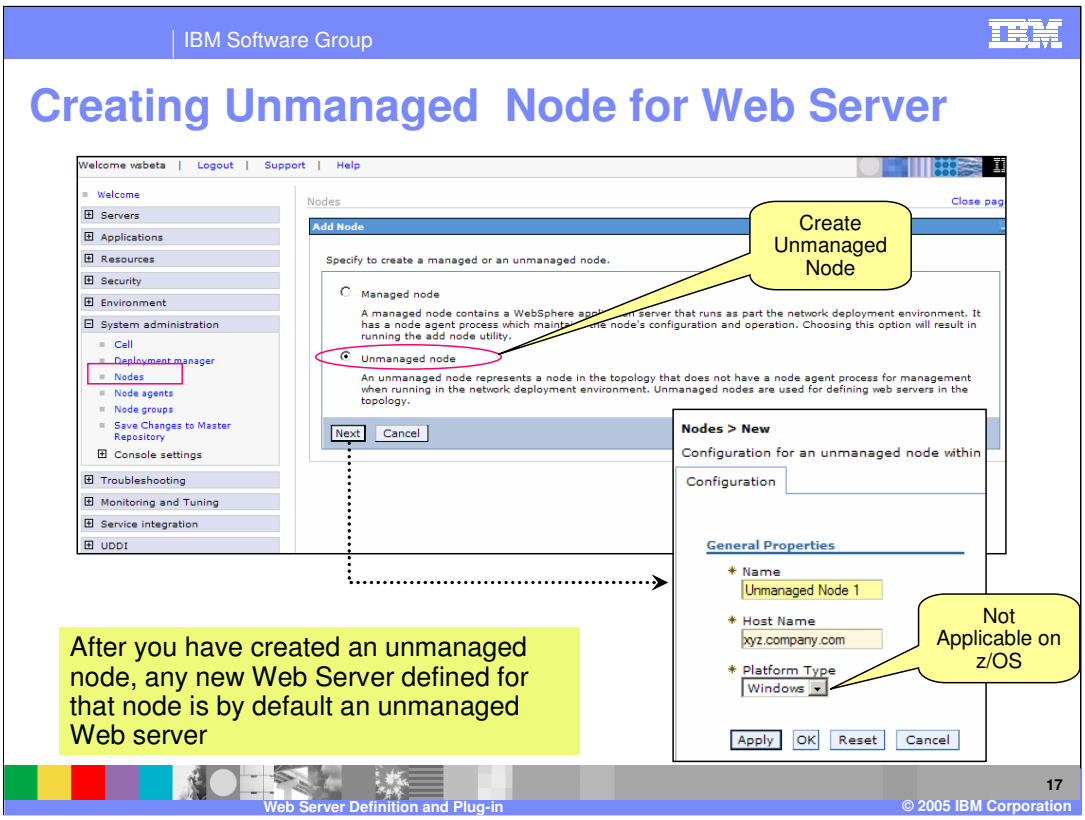
This section will demonstrate how to create and manage Web server definitions .

Steps for creating Web Servers and Mapping Applications

1. Create Web server definition using Administrative console or Wsadmin command or system provided JACL script file
   - Web Server is defined on a managed or unmanaged node
   - Managed or Unmanaged node must exist in the topology prior to creating a Web Server definition
2. Map applications to one or more defined Web Servers
   - Allows creating custom plug-in configuration files for specific Web server

This slide shows the steps for creating Web servers and Mapping Applications to them. A Web Server is defined on either a managed or an unmanaged node. You can use the Administrative console or wsadmin commands to create the Web server definition. In addition, there is a script called GenPluginCfg.sh, that can be used to create the Web server definition.

Once the Web servers are defined, applications can be mapped to them. This allows creating the custom plug-in files for specific Web servers, containing only those applications that are mapped to the Web server. It is possible to map an application to multiple Web servers. In that case, the plug-in files for those Web servers will contain entries for the application. This allows multiple Web servers to route requests to the same application.



From the system administration panel, a new node can be created. This node can be either managed or unmanaged. The panels here show how to create a unmanaged node.

A managed Node is created either through this panel, when creating a Custom profile, or when federating a Stand-alone application server.

Creating Web Server Definition

- Web Server definitions created during Web Server Plug-in install (on distributed platforms)
  - For Local Web Server Plug-in install and if the profile is an Express Stand-alone Application Server profile, the installer will create the Web Server definition within the Application Server configuration
  - For all other Scenarios, installer creates a JACL script, called configure<WebServerName>.jacl, which can then be modified and run for a specific WebSphere installation.
- For Network Deployment Cell, Web Server definitions can be created using DMgr Administrative console
- Using wsadmin: “createWebServer” administrative task can be used to interactively create Web Server definition
  - wsadmin $AdminTask createWebServer -interactive

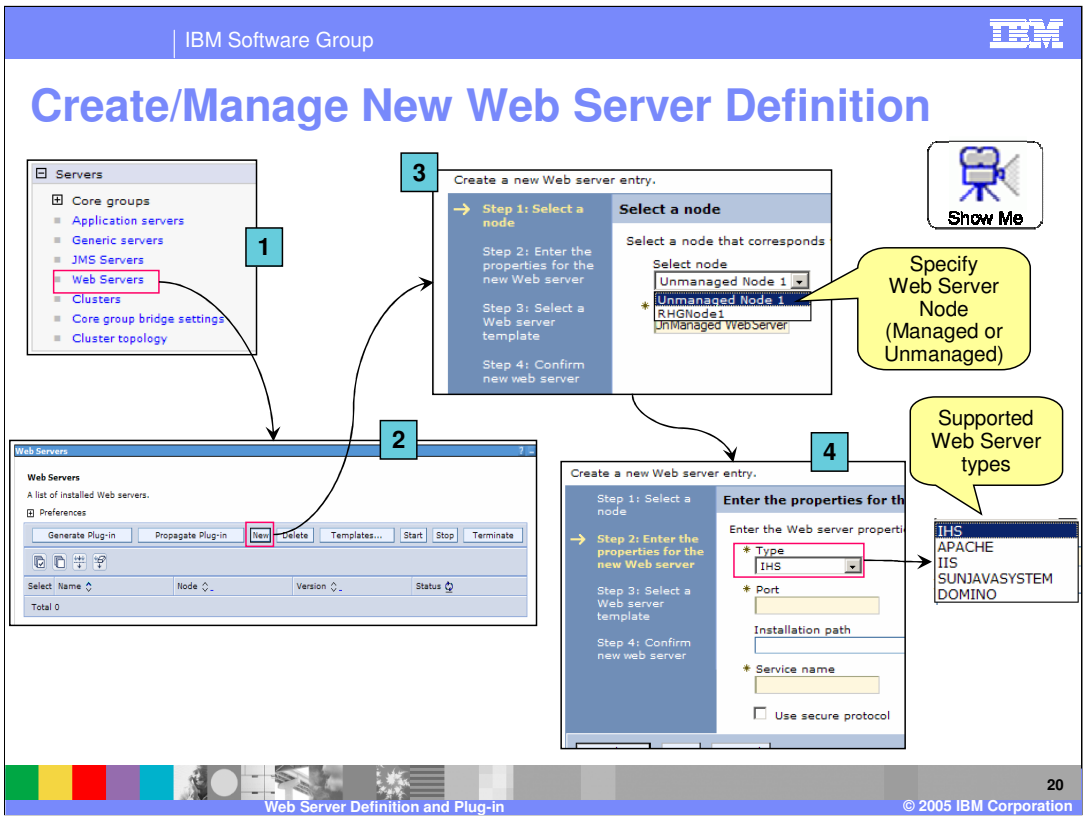
Web server definitions can be created during Plug-in install on distributed platforms, using the Administrative Console in a Network Deployment environment, or using the wsadmin command line tool. The installation on a Domino Go-based z/SO IHS is always manual.

The plug-in install either creates the Web Server definition or a JACL script called configureWebserverDefinition.jacl that enables you to create the Web Server definition later. The JACL script is useful for remote Web Server plug-ins, where the installer may not be able to access the remote machine.

configureWebserverDefinition JACL script

- Plug-in installation wizard under the covers use configureWebserverDefinition JACL script to create Web Server definition.
- configureWebserverDefinition script can be found in the <WAS_INSTALL_ROOT>/bin directory
  - This can be copied to the WebSphere install machine
- ConfigureWebServerName.sh script requires all the parameter necessary to create Web Server definition
- configureWebserverDefinition script will
  - Create unmanaged node if it does not exist
  - Create Web Server definition under the specified node
  - Map all the applications to the newly created Web Server
- Please see the usage documented in the configureWebserverDefinition.jacl script for complete detail on the options

On distributed platforms, the plug-in installer wizard uses the configureWebserverDefinition JACL script to create Web Server definition in the background. This slide describes the details of the JACL script, which is saved in the Web Server plug-in install bin directory. There is a special script created in the os390 directory on the distributed platform that can be used to create a Web server definition on a WebSphere for z/OS version 6.0.1 or later deployment manager.



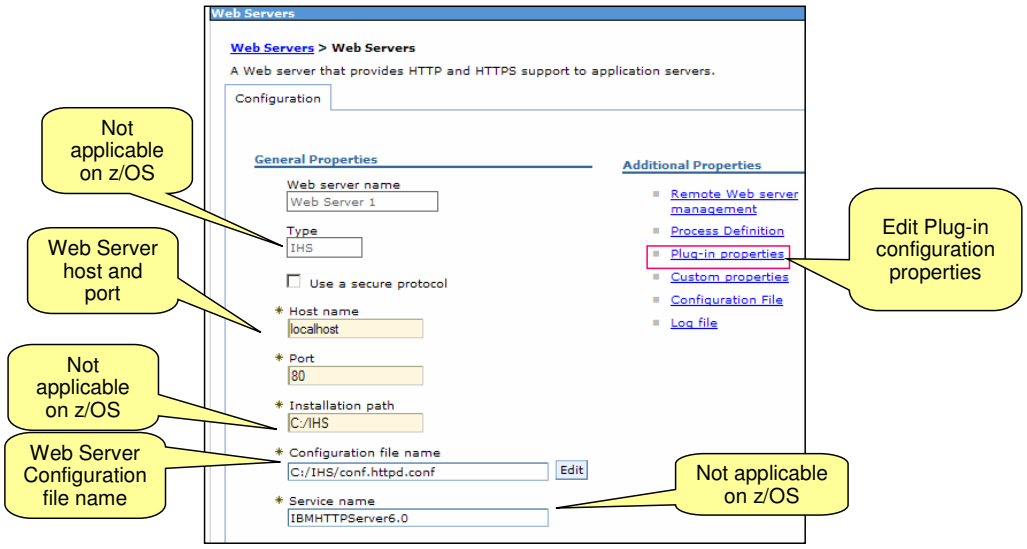
The steps for creating a Web server definition using the Administrative console is shown on this page:

Panels (1) and (2) show how to start creating the Web Server definition. When a new Web server definition is created, it must be associated with an existing node.

In Panel (3), the managed or unmanaged node on which the Web server will be defined is selected. In the example shown here, there are two possible nodes, both a managed and an unmanaged node have already been created in this topology. After a node is selected, the properties for the new Web server are entered. In panel (4), details for the Web server are provided. Click on the Show-me for a demonstration of creating a Web Server definition and mapping applications to the Web Server.

Managing Defined Web Server

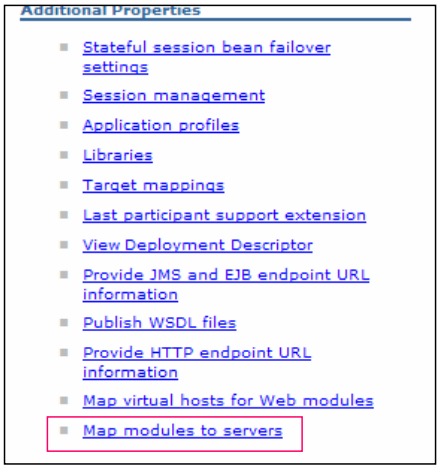
- Select Web Server to configure the Web Server and its plug-in configuration file



Once a Web server has been added, an administrator can access the plug-in for that Web server through the administrative console. From this section an administrator can configure the plug-in.

Mapping Applications to Web Server

- Select application to map the modules to Web servers
- Select the Application Server and one or more Web servers
- Can also be done using the following wsadmin script:
  - configureWebserverDefinition .jacl



Within the configuration for each application various application modules can be mapped to defined Web servers. An application can be mapped to more than one Web server as well.

Section

WebSphere Application Server V6

and

IBM HTTP Server (IHS) V6

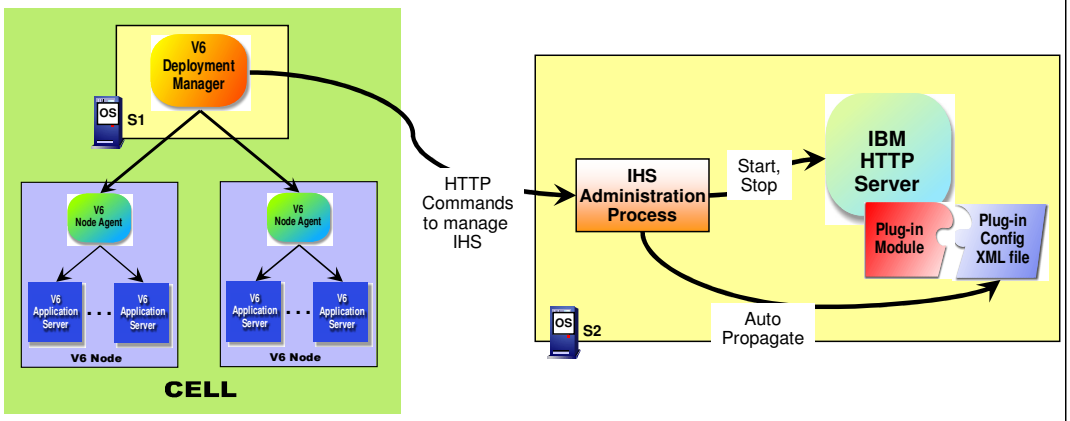
This section will detail the enhancements for the IBM HTTP Server in V6.0.2.

IHS Administration Overview

- Administrative functions for Apache based IBM HTTP Server (IHS) 1.3.X are handled by IHS Administrative Server (port 8008) through the IHS Administrative console
- Apache based IHS 2.0.X does not have any console administrative function.
  - Administration is done from the command line
- Apache based IHS V6 is bundled with WebSphere Application Server V6
  - Distributed platforms are based on Apache 2.0.47
  - Z/OS platforms are built on Domino Go
  - Administrative functionality is integrated into WebSphere Application Server V6 Administrative Console
  - Enhanced functionality with WebSphere Application Server

Earlier versions of the Apache based IBM HTTP Server were administrated through a separate administrative interface. Since version 2.0.x of the Apache based IBM HTTP Server this is no longer the case. Instead, administration is done from the command line. On Distributed platforms IHS V6 is bundled with WebSphere V6. More specific information on IHS can be found in the separate presentation for that product. On z/OS platforms, IHS is bundled with the z/OS operating system and can still be administrated through the Web based interface or through the WebSphere for z/OS v 6.0.1 or later administrative console.

IHS as Unmanaged (Remote) Node: Enhancements



- Provides administrative functions for IHS from WebSphere Application Server using the IHS Administrative process
- Provides ability to start and stop HIS and automatically propagate the plug-in configuration file to the IHS system and make configuration changes to httpd.conf
- Does not require Node Agent on the Web server machine

IBM HTTP Server based on Apache provides enhanced functionality for managing a Web server on an unmanaged node. This is accomplished through the use of a separate IHS Administrative process that runs on the same system as the Web server. The Deployment Manager communicates with the administrative process to manage the Web server. It might help to think of this separate administrative process as a smaller version of a node agent.

IHS Administration Server

- The IHS Administrative server runs as a separate instance of the IHS Web Server
- IHS Administrative Component for IHS 6.0 includes:
  - IHS Administration module (mod_ibm_adm.so or IBMModuleAdmin.dll).
  - IHS Administration configuration file (admin.conf)
    - Default port for the IHS Administration server is 8008.
- IHS Administrative authentication password file (htpasswd.admin)
  - Initially blank, which prohibits access to IHS Administration
  - Administrator updates IHS Administration password file using the htpasswd utility program
  - At a command prompt for /<IHS-Home>/bin enter the command:
    - ./htpasswd -m ../conf/admin.passwd *<user_name>*
- To start and stop the administrative server
  - <IHS-Home>/bin/adminctl start
  - <IHS-Home>/bin/adminctl stop
  - Administer as a Windows Service

This slide details various commands and files that are needed to administrate the Apache based IBM HTTP Server. In particular, information is offered here on the separate administrative process as a reference.

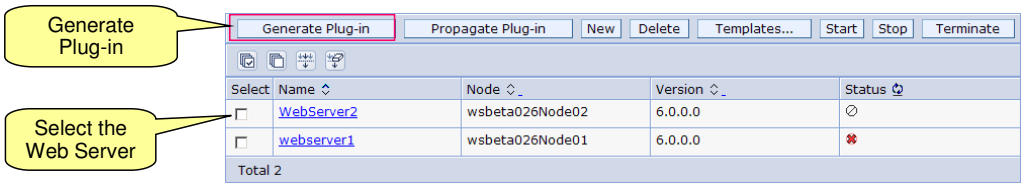
Section

Web Server Plug-in Configuration file:

Generation, Customization and Propagation

This section will detail the enhancements for IBM HTTP Server in V6.0.2.

Generating Plug-in with Administrative Console



- Plug-in configuration file can be generated using the Administrative Console after defining Web Server and mapping the Applications.
- Only Applications mapped to the Web Server will be added to the plug-in file (Application Centric)
- Plug-in files are generated automatically for any Application and Virtual host modifications. The default behavior can be changed by modifying plug-in property.
- Generated Plug-in files are always saved in the repository under Web Server defined directory.
  - config\cells\CellName\nodes\NodeName\servers\webServer1\plugin-cfg.xml

Once a Web server has been added, an administrator can generate a plug-in for it through the administrative console. Plug-in files generated for Web Servers will contain all the URIs for Applications mapped to the Web Server. By default, the plugin-cfg.xml files are generated automatically for any change to the Application or virtual host setting. Generated plug-in files are always saved in the master repository under the Web Server defined directory.

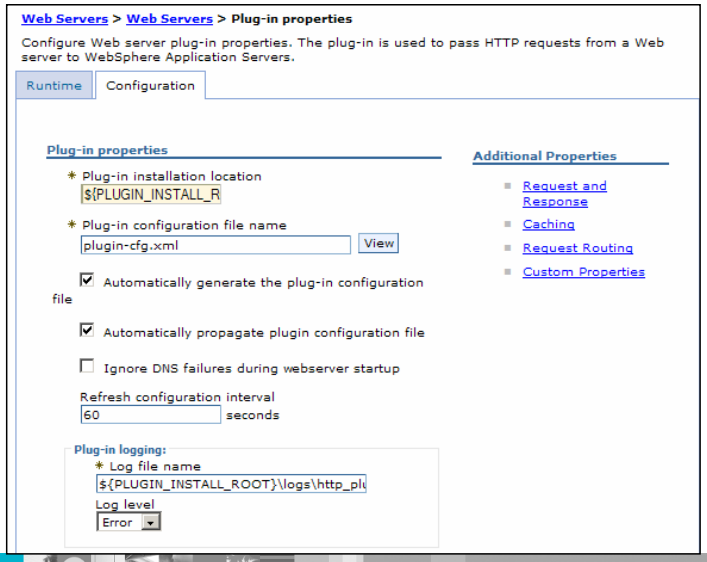
Generating Plug-in using command

- **GenPluginCfg.sh** command generates Web server plug-in configuration file, plugin-cfg.xml
- **Application centric** plug-in file generation
  - Used to generate plugin-cfg.xml file only for applications mapped to specific Web Server
  - Use –webserver.name option
    - Example: GenPluginCfg.bat -webserver.name webServer1
      - plug-in file for webServer1 and saved under webServer1 defined server directory in the master repository
- **Topology centric** plug-in file generation
  - Same as V5.x
  - Used to generate plug-in file for the entire cell, node or Application Server
  - Example: GenPluginCfg -cell.name CellName -node.name appServerNode -server.name server1
    - Above command will generate plug-in file for all the applications installed on the server1

The GenPluginCfg.sh command is used to regenerate the WebSphere Web server plug-in configuration file, plugin-cfg.xml. When the GenPluginCfg.sh command is issued with the -webserver.name webservrName option, the plug-in configuration file is created for the Web server. This setting in this generated configuration file are based on the list of applications that are deployed on the Web server. When this command is issued without the -webserver.name webservrName option, the plug-in configuration file is generated based on topology within the application server.

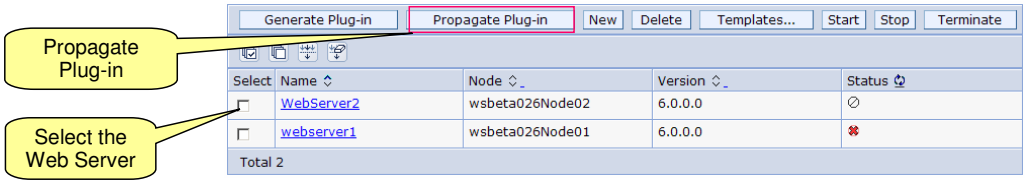
Customize Web Server Plug-in Properties

- Select Web Server to configure its plug-in configuration file



This is the administrative console screen used to configure the plug-in. Various additional properties can also be configured for the plug-in from the additional properties on the right. This should allow an administrator to perform most customizations without having to manually edit the plug-in configuration.

Propagation of the plug-in file



- Plug-in file can now be propagated to the remote Web server system
- For a Web Server defined on a managed node, propagate action will invoke “nodeSync” operation. Node Agent will handle replicating the repository on the remote machine
- For a Web Server defined on an Unmanaged node , propagation is possible only with IBM HTTP Server
  - DMgr Administrative Console communicates with IHS Administrative Server configured on remote machine
- HTTP protocol is used for transferring the plug-in file

Once the plug-in configuration file has been created, it must be copied to the appropriate Web server. For an Apache-based Web server on a managed node this can be done by invoking the node sync operation through the WebSphere for z/OS v 6.0.1 or later administrative console. The node agent on the managed node will handle replicating the plug-in configuration file stored in the repository to the destination Web server on the remote machine. For a Web server defined on an unmanaged node, automatically copying the plug-in configuration file is only possible through the use of the separate administrative process of an IBM HTTP Server version 6 Web server. For other Web servers, including the Domino Go-based Web server for z/OS platforms, the plug-in configuration file must be manually copied into the appropriate directory on the remote Web server system.

Plug-in Propagation : At a Glance

Web Server Scenario	Requirement on remote Web Server machine	Destination after propagation
Z/OS (Domino Go) Remote Non-IHS Unmanaged Node	None	Files must be copied manually Recommended location: <Plug-in Install>/config/WebServerName/plugin-cfg.xml
Local Web Server Unmanaged	Configure Web Server directly where plug-in file are generated under master repository[1]	Propagation not required
Remote IHS Web Server Unmanaged Node	Requires IHS AdminService to be running, with write permission for destination directory	Files are propagated under Plug-in install directory on remote machine. <Plug-inInstall>/config/WebServerName/plugin-cfg.xml
Any Web Server under Managed Node	Node Agent should be running on the Managed Node	Files are replicated under node configuration directory. <WAS_Profile_Home>/config/cells/<CellName>/nodes/<NodeName>/Servers/<WebServerName>/plugin-cfg.xml

[1] Plug-in file for Web Server are generated under Web Server definition in the master repository

<WAS_Profile_Home>/config/cells/CellName/nodes/NodeName/Servers/WebServer1/plugin-cfg.xml

This slide details the plug-in automatic propagation and destination location on a remote system. Local Web Servers are configured to allow updates when the plug-ins are generated so no manual propagation is necessary. Local Web Servers on z/OS are configured by manually copying the plug-in configuration file to the appropriate location for the Web sever. If the web server has been configured to look for the plug-in configuration file in the WebSphere repository, no copying is required. If you are working with an Apache-based IHS Web Server configured as an unmanaged node , the files are propagated under the plug-in installation directory <Plug-in Install>/config/WebServerName/plugin-cfg.xml file. For Managed or Custom nodes the node sync operation takes care of replicating the configuration.

Section

z/OS Current Points
and
Work Arounds

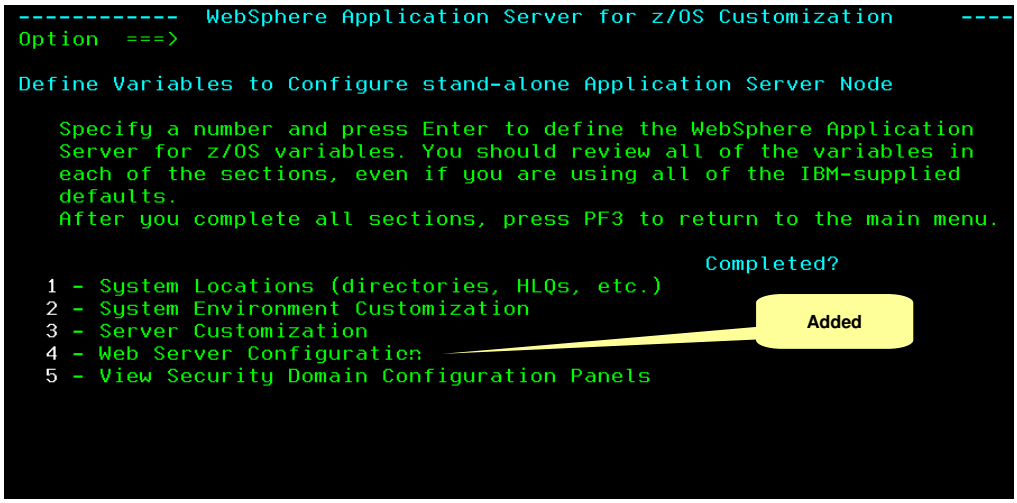
This section will show how to create and manage Web server definitions .

Web Server Definition: ISPF Panels

- Web Servers can now be defined in WebSphere cell topology as a Web Server node
  - Managed or unmanaged
    - Managed nodes contain a node agent
  - Name used during Web Server install is orthogonal to the name used in Web Server definition
- Association of an application to one or more Web Servers
- Generation of custom plug-in configuration files (plugin-cfg.xml) for a specific Web Server
  - Web Servers target specific applications running in a cell
  - Automatically generated by the Deployment Manager

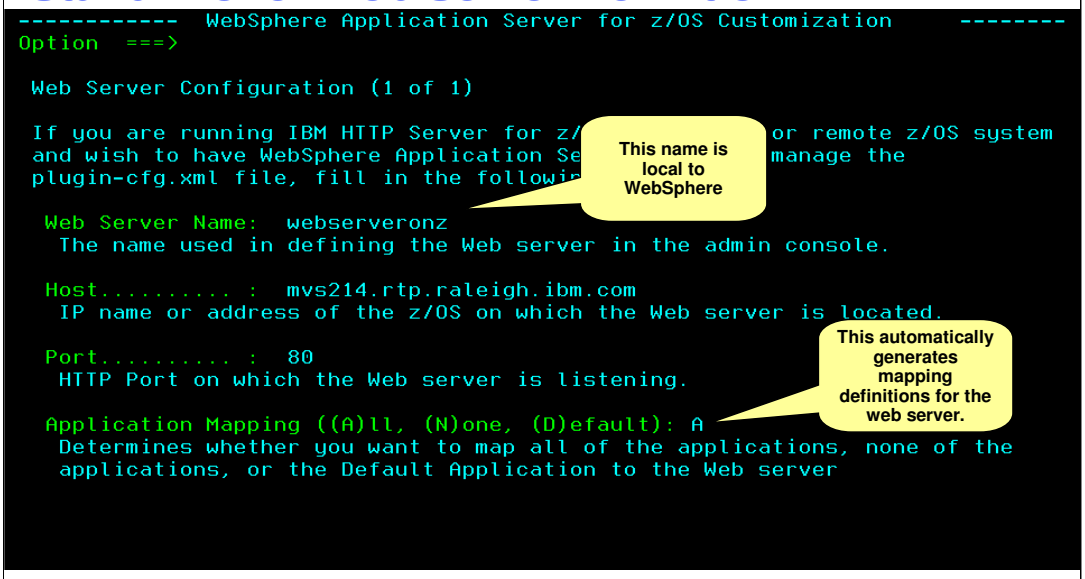
Web servers can be defined as managed or unmanaged nodes in the WebSphere topology. A managed Web Server node can be administered from the WebSphere administrative console. Applications can be associated with specific Web Server nodes so that only those associated Web servers will handle requests for the application. Custom plug-in configuration files can be generated for specific Web servers.

Stand Alone: New Configuration Panels



There are new ISPF configuration panels to specify Web server definitions. The panel shown here is for Deployment Manager configuration. All the configuration panels for **Web Server Configuration** are optional. Their purpose is to allow you to create Web server definitions in the deployment manager using batch jobs. The same definitions can also be created using the Administrative Console. In version 6.0.2, these panels for the deployment manager no longer allow creating a Web Server definition. For a deployment manager, the definitions must be created with the administrative console or scripts.

Stand Alone: Web Server Definition

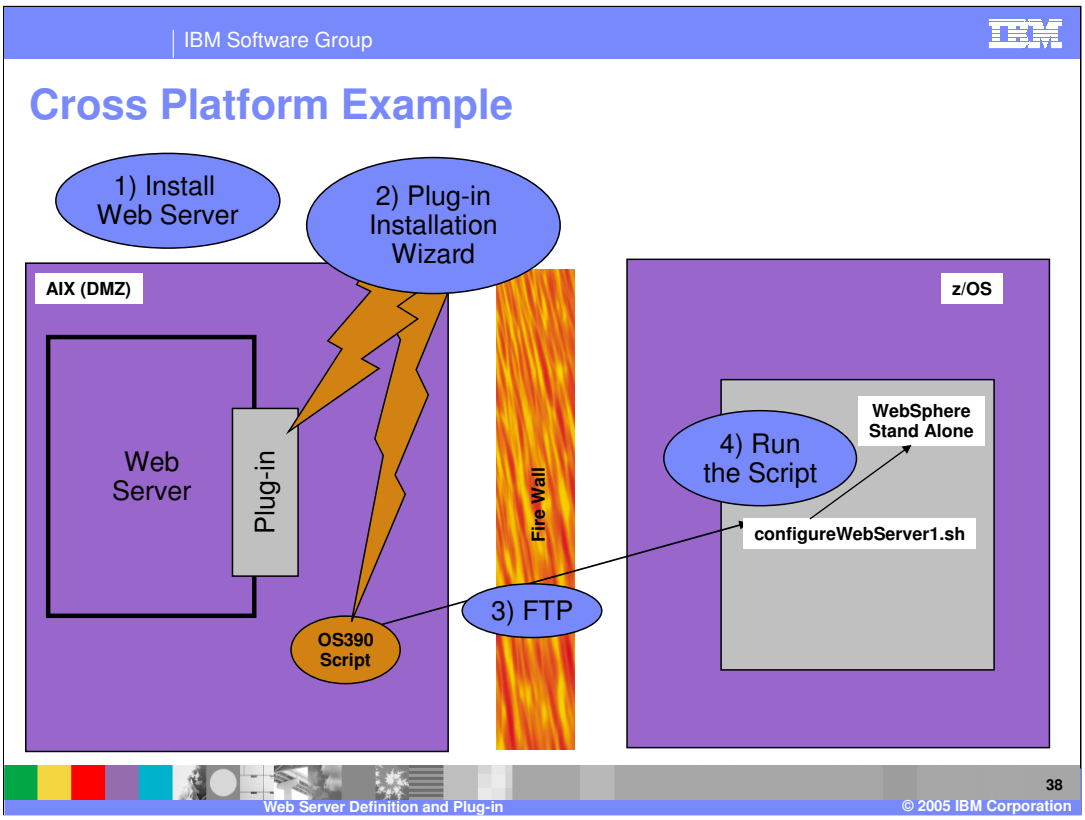


This slide shows the information required for a Web server definition in **Stand Alone.** There are similar panels in the **Federate a Node** panels. Some of the panels have different fields. The name here (webserveronz) is not related to any names or aliases inside the Web Server.

Cross Platform: a Special Case

- The Plug-in installation wizard creates a script that is used to create a Web Server definition for a remote Web Server
  - configure*Web_Server_Name*
  - Located in plugin_install_root/bin
- Copy to system with WebSphere Application Server and run the script
  - Copy to *WAS_HOME*/bin
  - For cross-platform support scripts are available in *install_root*/Plugins/bin/crossPlatformScripts

A special case is the ability to manage a Web Server running on a distributed platform from a WebSphere administrative console running on z/OS. From a high level, a normal plug-in install is performed on the distributed platform. Part of this installation creates a special script file that can be used to define the remote Web Server to the z/OS based WebSphere. The details are shown in the next slide.



There are a couple of big advantages to the configuration shown in this graphic:

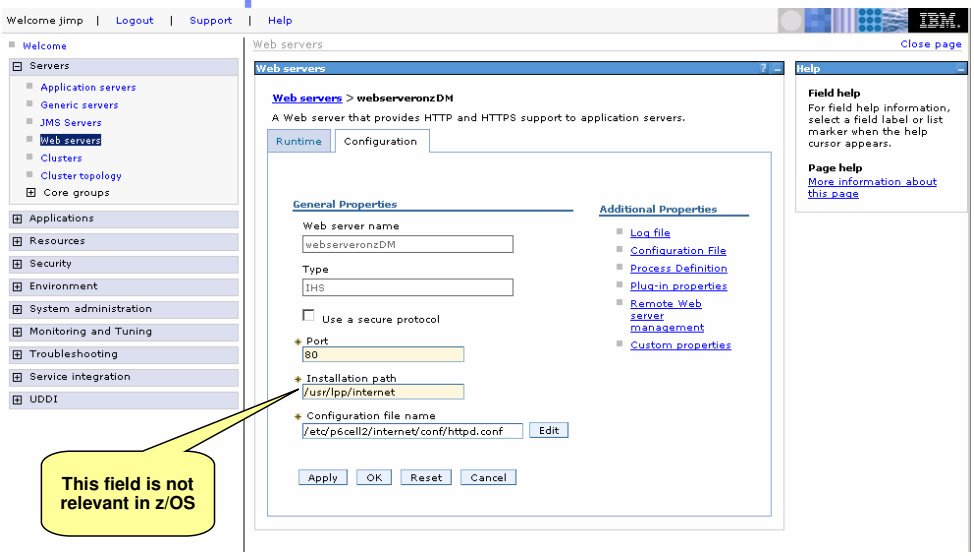
1) The Web Server can be located in the DMZ on a less expensive system and firewall security is not compromised. The other firewall and the internet are off to the left.

2) When you run the script, all the applications are mapped.

The administrative screens will be discussed along with Web Servers under the Deployment Manager.

This is a bonus feature because the AIX platform does not need a Node Agent running. The Administrative Application can administer the plugin-cfg.xml file for the Web Server using scripting or the administrative console and push it over to a plug-in agent, which saves the new xml file and restarts the Web Server, causing it to read the new configuration.

General Properties



The installation path is there so WebSphere can execute the Web Server programs, which are currently only important for the Apache-Based version of IHS. The Configuration File Name presently defaults to <config root>/<AppServer name>, so you will probably want to change this. In this example, the path is into a <config root> tree for an application server. In the case of a Deployment Manager with a federated node, the add node process causes a glitch in a variable. You can edit the https.conf file by clicking the **Edit** or **Configuration File** file buttons.

Add the plug-in directives to the httpd.conf file

The best place to put the plug-in directives is beneath the comment in the httpd.conf file that says:

*** WAS Directives ***

Do not forget to comment out the Pass /* directive in the httpd.conf file; otherwise, the Service directive might not be processed (depending on the order of directives in the httpd.conf file).

Each of these directives must be entered entirely on one line. The lines are split on this slide due to space limitations.

```
ServerInit  /usr/lpp/zWebSphere/V6R0/bin/ihs390WAS60Plugin_http.so:init_exit
                                        <very_long_pathname>plugin-cfg.xml

Service     /SuperSnoopWeb/*
            /usr/lpp/zWebSphere/V6R0/bin/ihs390WAS60Plugin_http.so:service_exit

ServerTerm  /usr/lpp/zWebSphere/V6R0/bin/ihs390WAS60Plugin_http.so:term_exit
```


This slide demonstrates manually installing a plug-in configuration file. Adding the plug-in directives to the httpd.conf file is **not** done automatically. Be careful when performing this edit as it is easy to make a mistake. Include the full path to the plugin-cfg.xml file.

In this example, *SuperSnoopWeb* is the context root of the application for which requests will be forwarded to WebSphere Application Server.

Here is the location of the plugin-cfg.xml file on the system:

/wasV6config/lwcell/lwnoded/AppServer/profiles/default/config/cells/lwcellnd/nodes/lwnoded/servers/mywebserver/plugin-cfg.xml

The easiest way to find the plugin-cfg.xml file on your system is by issuing the find command from the Unix shell:

find . –name plugin-cfg.xml

The Web server must be stopped and restarted to process the ServerInit, Service, and ServerTerm directives. It must also be stopped and restarted if you make any changes to the plugin-cfg.xml file, since the ServerInit directive is executed only when the Web server is started. The Service directive is executed on each Web server request cycle, unless a preceding directive catches and processes the request first.

If you want all requests to be handled by the plug-in, then the first parameter of the Service directive can be specified as simply /* . If this technique is used, then be sure to specify that Service directive after all Pass, Exec, Fail, and Map directives in the httpd.conf file so that the Pass, Exec, Fail, and Map directives will also be processed.

Section

Summary and Reference

This section includes the summary and references.

Summary

- A new concept in unifying administration of Web Servers
- WebSphere Application Server V6.0.2 supports defining Web servers as a managed or unmanaged node
- Supports creating custom plug-in configuration files for each Web server

This presentation explained the enhancements in WebSphere Application Server V6.0.2 support for Web servers, detailed the concept of managed and unmanaged nodes for Web servers, and explained the capabilities provided by the Apache-based IBM HTTP Server V6.

Reference

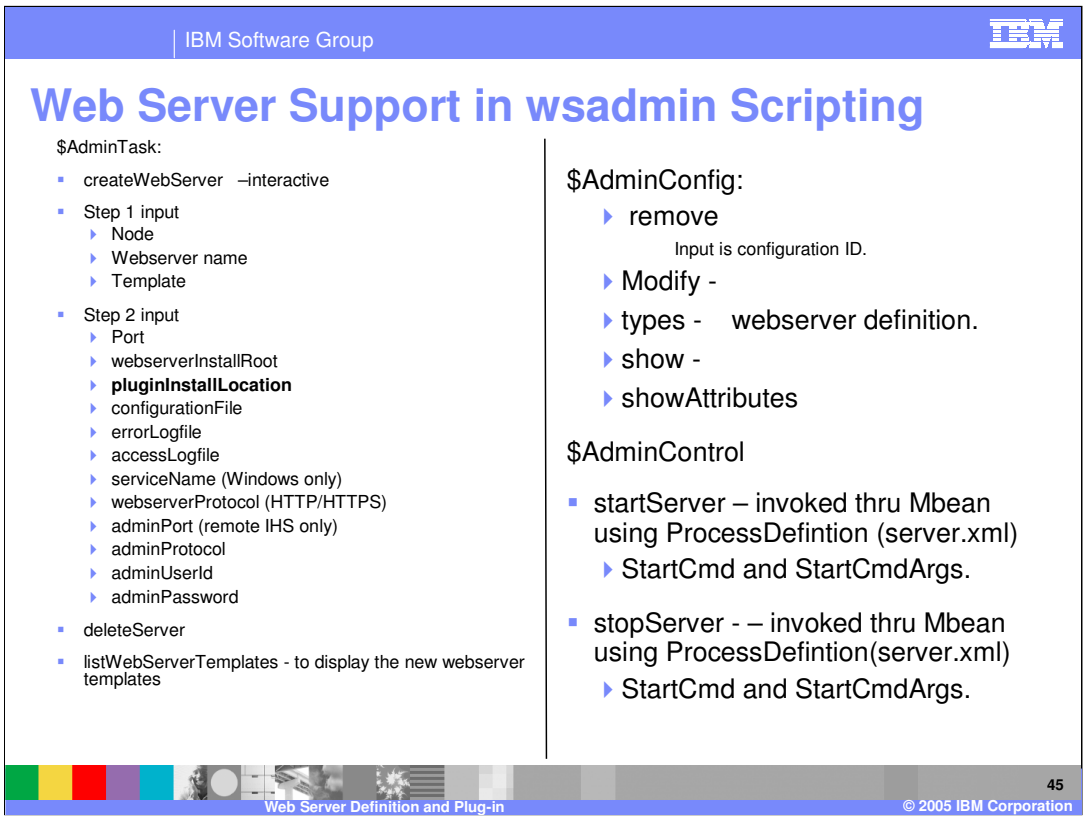
- Configuring and Troubleshooting the WebSphere for z/OS Version 5 HTTP Server Plug-in
  - http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS829
- IBM HTTP Server
  - HTTP://www-306.ibm.com/software/webservers/HTTPservers/library/
- Understanding the WebSphere Application Server Web server plug-in
  - HTTP://www-106.ibm.com/developerworks/websphere/library/techarticles/0310_cocasse/cocasse.html

Shown here are some additional references.

Appendix

Wsadmin commands to create Web server definition

Appendix provides details of wsadmin tasks to create and manage Web server definitions.



wsadmin tasks to create Web server and modify the configuration are shown here for reference.

configureWebserverDefinition Usage

wsadmin.sh -f \${WAS_HOME}/bin/configureWebserverDefinition.jacl

- webserverName : Name of the Web server
- webserverType : Type of the web server.
- webserverInstallLocation : Location of the Web server
- webserverConfigFile : Web server configuration file
- webserverPort : Listening port of the web server (80)
- mapApplications : Specifies whether the existing applications are to be mapped to the web server.
  Valid values are: MAP_NONE MAP_DEFAULT MAP_ALL
- pluginInstallLocation : Location of the plug-in install
- webserverNodeType : Type of the node holding Web server
  Valid values are : managed, unmanaged
- webserverNodeName : Node at which web server should be defined (created unmanaged if does not exist)
- webserverHostName : Host name of the web server (Needed only for unmanaged node)
- webserverOS : Operation system of the web server machine (Needed only for unmanaged node)
- Values are : windows solaris aix hpux linux os400 os390

wsadmin tasks to create Web server definition, using configureWebserverDefinition.jacl script is shown here for reference.

Template Revision: 11/02/2004 5:50 PM

Trademarks, Copyrights, and Disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	CICS	IMS	MQSeries	Tivoli
IBM(logo)	Cloudscape	Informix	OS/390	WebSphere
e(logo)business	DB2	iSeries	OS/400	xSeries
AIX	DB2 Universal Database	Lotus	pSeries	zSeries

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

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

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel 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.

Linux is a registered trademark of Linus Torvalds.

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

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided AS IS without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED AS IS WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.