WebSphere Service Registry and Repository v8.5 Production Deployment Scenario

Authors:

Chris Dudley (chris.dudley@uk.ibm.com)
Anna Pendrich (anna.pendrich@uk.ibm.com)
Darren Sullivan (DARRENSU@uk.ibm.com)
Joshua Naylor (JOSHUANA@uk.ibm.com)
Daniel Howden (danhowden@uk.ibm.com)

Table of Contents

1 Scenario

IBM WebSphere Service Registry and Repository (WSRR) supports several different modes of installation and deployment. There are both GUI and command line or silent methods.

It is very common for production type environments to be UNIX based servers with no local GUI. In these environments it is easiest to make use of the silent install interface for both installation and deployment. The silent interface also opens up the possibility of scripting the entire install and deployment process if desired.

This document will describe the steps necessary to install a new WSRR cluster environment on clean AIX servers. It will assume that new copies of WAS are being installed and a new cluster created.

The scenario covered here also assumes the installation is being done as a non-root user. In these circumstances slightly different installation commands are used. Please see the main product Information Center for the equivalent commands for root.

It is assumed the non-root user already exists and has write access to a file system large enough to install the required software. This document will use the same variable name throughout to represent the non-root user but it is not necessary for the non-root user to have the same name on all nodes.

An operating system user and group will also need to be created for the IHS web server.

For this scenario it is assumed that there is no internet access from the target servers. This is a fairly common restriction on production servers and so is included here too.
It will not cover how to install IBM DB2, instructions on how to do that can be found in the DB2 Information Center. In practice the database server is normally controlled by DBAs and so is outside the remit of this document.

2 Topology

The topology chosen for this document is a cluster with a deployment manager on a separate node. It is fronted by a separate web server node.

The database is on another separate machine.

For the purposes of this document the cluster has two nodes, but it could easily have more if desired, simply repeat the appropriate steps on the extra nodes.

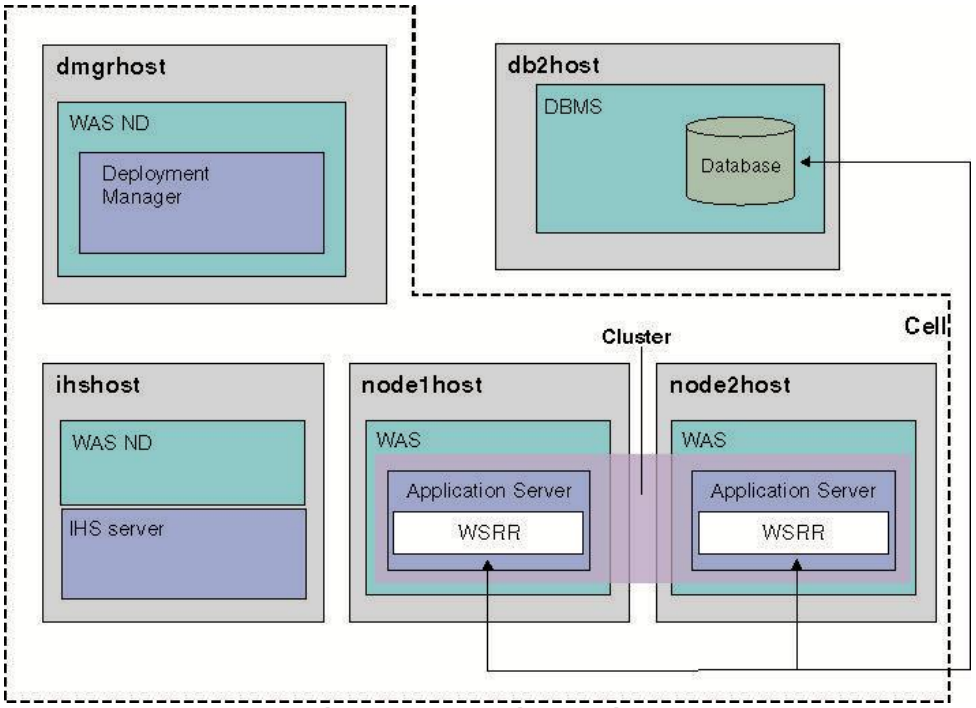

Figure 1: Scenario Topology

This document will assume the nodes are different physical machines. It is also possible to combine multiple nodes on a single machine, e.g. the deployment manager on the same machine as the IHS web server; however that is outside the remit of this document.

3 Prerequisites

Operating System

AIX v6.1 (Maintenance package 6100-00-04 or greater) or
AIX v7.1 (Maintenance package 7100-00-02 or greater)

Disk space

Space for the install images:	3 Gb
Temporary disk space:	0.1 Gb (on each node)
WebSphere Application Server:	1 Gb
WSRR:	1 Gb
Deployment Manager profile:	0.2 Gb
Managed profile:	0.1 Gb
IBM HTTP Server:	0.3 Gb
Database:	2.0 Gb

So space required for the different nodes in our scenario would be:

Deployment manager:	2.2 Gb
Cluster node:	2.1 Gb
IBM HTTP Server node:	1.4 Gb
Database node:	2.0 Gb

Database

WSRR supports several different database providers:

- IBM DB2
- IBM DB2/zOS
- Microsoft SQL Server
- Oracle 11g & 12c

Please see the WSRR System Requirements pages for detailed information on minimum database versions: http://www-304.ibm.com/support/docview.wss?uid=swg27024968#AIX_tab

For the purposes of this document we will assume the use of IBM DB2 v10.5. IBM DB2 WSE v10.5 is supplied with the product. Fixpacks for it can be obtained from the IBM DB2 support website.

WebSphere Application Server

WSRR v8.5 requires WebSphere Application Server v8.5.5.1 or greater. This is supplied with the product and can be installed concurrently.

GNU Tar

On AIX the WSRR download package must be extracted using the GNU tar command. GNU tar for AIX can be downloaded from the IBM AIX Toolbox website: http://www.ibm.com/systems/power/software/aix/linux/toolbox/download.html

Unzip Command

Extracting the IBM HTTP Server images will require the `unzip` command. It is also possible to use the `jar` command to extract a zip file on AIX, as follows:

```
jar –xvf example.zip
```

Umask

Select a umask that allows the owner to read/write to the files, and allows others to access them according to the prevailing system policy. For root, a umask of 022 is recommended. For non-root users, a umask of 002 or 022 could be used, depending on whether or not the users share the group. To verify the umask setting, issue the following command:

```
Umask
```

To set the umask setting to 022 for the current shell, issue the following command:
`umask 022`

Ulimit

It is necessary to increase the default process file descriptor limit.
For AIX:

1. Open the file `/etc/security/limits`
2. Edit or add the default section and include this line:
   `nofiles = 20000`
3. Save and close the file.
4. Log off and log in again.

Page Size

The default AIX page size is only 512Mb and this can cause deployment problems in WebSphere Application Server. It is recommended to increase this to at least 4Gb using the normal AIX administration tools (`smitty lvm`).

4 Variables

This is an explanation of variables used in this document.

<DVD_ROOT>	Root directory of DVD media or extracted download package
<INSTUSER>	non-root user install is being run as
<WAS_HOME>	WebSphere Application Server installation directory, must be somewhere writable by the <INSTUSER>, e.g. /devel/ServiceRegistry
<WAS_ADMIN>	WebSphere Application Server Administration User ID
<WAS_PWD>	Password for the user specified by <WAS_ADMIN>
<LOG_FILE_PATH>	Directory for the Installation Manager log file e.g. /tmp
<DMGR_PROFILE_NAME>	Deployment manager profile name e.g. Dmgr01
<NODE1_PROFILE_NAME>	1st custom profile name e.g. Custom01
<NODE2_PROFILE_NAME>	2nd custom profile name e.g. Custom02
<NODE3_PROFILE_NAME>	3rd custom profile name e.g. Custom03
<DMGR_PROFILE_ROOT>	Deployment manager profile root directory e.g. /devel/ServiceRegistry/profiles/Dmgr01
<NODE1_PROFILE_ROOT>	1st Custom profile root directory e.g. /devel/ServiceRegistry/profiles/Custom01
<NODE2_PROFILE_ROOT>	2nd Custom profile root directory e.g. /devel/ServiceRegistry/profiles/Custom02
<NODE3_PROFILE_ROOT>	3rd Custom profile root directory e.g. /devel/ServiceRegistry/profiles/Custom03
<CELL_NAME>	WebSphere Application Server cell name e.g. WSRRCell
<PLUGIN_HOME>	Web server plug-in installation directory

All names in <> throughout this document refer to the values that you specify for the variables given in the table above. So, for example, <WAS_HOME> might refer to "/devel/ServiceRegistry".)

5 Access WSRR Package

You can obtain the product code in either of the followings ways:

- From the product media packs which include CD-ROM and DVD media.
- From the Passport Advantage site, where licensed customers can download installation images. For more information about the images available for download, see the Passport Advantage download document.

For the download images from Passport Advantage, ensure that you use GNU Tar to extract the images. Using the version of tar that comes with AIX will result in installation errors. For example:

```
gunzip WSRR_V8.5_AIX_ML.tgz
gtar -xvf WSRR_V8.5_AIX_ML.tar
```

6 Deployment Manager

6.1 Install WSRR

To install silently, you create a response file that contains the information for your system, and then run an installation command that calls that response file. WSRR supplies a template response file for you to edit. A sample response file for this scenario is provided in Appendix B: Example WSRR Installation Manager Response File on page 52.

6.1.1 Creating a response file

The response file provides input to the Installation Manager (IM) when it is running a silent installation. You edit the response file to control the silent installation.

The response file is located in the `responsefiles` directory at the top level of your WSRR product DVD or download directory. All the default paths specified in the response file are relative to the `responsefiles` directory. The file is named `template_response.xml`. The response file is called from the silent installation script and should not be moved or renamed (although you can make a copy of the original response file before you modify it so that you can revert to the default version if required).

The response file passes parameters to the IM silent installation command. The response file completes the following tasks:

- Defines the repositories that IM uses to locate install or upgrade files for IM, WebSphere Application Server, and WSRR.
- Upgrades IM and WebSphere Application Server, if required.
- Installs WSRR into a location relative to the WebSphere Application Server.

You must modify certain fields in the response file before you run the script file that calls it.

a) Specify the repository locations. A repository is a location that stores data for installing, modifying, rolling back, updating, or uninstalling packages. You modify the response file to specify the location of the following repositories, or you can leave the fields unchanged to accept the default values.

b) **IM Repository Location**
Specify the location of the repository that holds installation or update data for IM itself. By default this is set to the IM directory at the top level of your WSRR product DVD or download directory. For example:

```
<repository location='../IM/' temporary='true'/>
```

WSRR Repository Location
Specify the location of the repository that holds installation data for WSRR. By default this field is set to the repository directory at the top level of your WSRR product DVD or download directory. For example:

```
<repository location="../repository/" />
```

You can also add a repository entry for the location that holds fix packs for your version of WSRR. If you add this repository in addition to the installation repository for WSRR, then the silent installation will install the latest version, including available fix packs. For example:

```
<repository location="../8501_fp" />
```

c) Specify the location of Installation Manager (IM). You specify the location where IM is installed. The response file contains a placeholder path which you must edit. You can specify the standard installation location for IM:

```
<profile kind='self'
installLocation='/home/<INSTUSER>/IBM/InstallationManager/ecl
ipse' id='IBM Installation Manager'>
  <data key='eclipseLocation'
value='/home/<INSTUSER>/IBM/InstallationManager/eclipse'/>
</profile>
```

You should also edit the Eclipse cache location to match the path that you specified for the IM installation.

```
<preference value="/home/<INSTUSER>/IBM/InstallationManager
/eclipseCache"
name="com.ibm.cic.common.core.preferences.eclipseCache" />
```

Note: You should not edit the `<install>` section which instructs the silent install script to install IM. If IM is already installed, this step is skipped.

d) Specify the location of WebSphere Application Server (this directory is also used to install WSRR). Replace `<WAS_HOME>` in the example below.
Example:

```
<profile installLocation='<WAS_HOME>' id='IBM WebSphere
Application Server - ND'>
  <data key='eclipseLocation' value='<WAS_HOME>'/>
  <data key="cic.selector.nl" value="en" />
</profile>
```

The WebSphere Application Server profile name is set to "IBM WebSphere Application Server - ND" by default. If you have an existing WebSphere Application Server installation that uses that profile id, then you must edit the response file to specify an alternative name.

6.1.2 Run the silent installation

After you have created your response file, you run a command that calls the file.

Since `<INSTUSER>` is a non-root user, use the following command to start a silent installation:

```
<DVD_ROOT>/IM/userinstc -input
<DVD_ROOT>/responsefiles/template_response.xml -log
<LOG_FILE_PATH>/iminstall.log –acceptLicense
```

You must also ensure that the non-root user has write access to the installation directories specified in the response file, `template_response.xml`. They also require write access to the IM subdirectory of the install media and its contents.
The installation command reads the response file, installs packages as directed and writes a log file to the directory you specified. You should specify a temporary directory to write the log file to.

The following log entry constitutes a successful installation log:

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
</result>
```

Anything between the `<result>` tags points to an error during installation, and needs further investigation.

6.2 Create Deployment Manager Profile

You create the profile by running the manageprofiles command with a response file that defines the profile. Template response files can be found in the following directory:

```
<WAS_HOME>/WSRR/install/profile-responsefiles
```

A sample response file for the deployment manager is also shown in Appendix E: Example Deployment Manager Manageprofiles Response File on page 62.

6.2.1 Create a deployment manager response file

Edit the `create-dmgr.txt` file to specify values for the following parameters (or accept the default values):

profileName

The name of the profile. The default profile name is based on the profile type and a trailing number, for example: `Dmgr01` The profile name must not contain spaces or characters that are not valid such as: *, ?, ", <, >, ,, /, \, and |. The profile name that you choose must not already be in use.

profilePath

The fully qualified path to the profile. `<DMGR_PROFILE_ROOT>` For example, `/devel/ServiceRegistry/profiles/Dmgr01`.

templatePath

The fully qualified path to the dmgr.wsrr template file in the installation root directory. For example, `/devel/ServiceRegistry/profileTemplates/dmgr.wsrr`.

enableAdminSecurity

Leave as true to enable administrative security, or set to false to disable administrative security.

adminUserName

Specifies the user ID that is used for administrative security (not required if you set `enableAdminSecurity` to false).

adminPassword

Specifies the password for the administrative security user ID. (not required if you set `enableAdminSecurity` to false).

6.2.2 Run manageprofiles

Run the manageprofiles command to create the deployment manager profile:

```
<WAS_HOME>/bin/manageprofiles.sh -response
<WAS_HOME>/WSRR/install/profile-responsefiles/create-dmgr.txt
```

The command displays status as it runs. Wait for it to finish. Normal syntax checking on the response file applies as the file is parsed like any other response file. Individual values in the response file are treated as command-line parameters.

You can see that your profile creation completed successfully if you receive the message "INSTCONFSUCCESS: Profile creation succeeded.", and you can check the following log file:

<WAS_HOME>/logs/manageprofiles/profilename_create.log

6.3 Start the deployment manager

You can start the deployment manager from the command line.

Change directory to <DMGR_PROFILE_ROOT>/bin, for example:
/devel/ServiceRegistry/profiles/Dmgr01/bin
Enter the command:
./startManager.sh

After starting the deployment manager status messages are displayed.

7 Cluster Member #1

7.1 Install WSRR

To install silently, you create a response file that contains the information for your system, and then run an installation command that calls that response file. WSRR supplies a template response file for you to edit. A sample response file for this scenario is provided in Appendix B.

7.1.1 Creating a response file

The response file provides input to the Installation Manager (IM) when it is running a silent installation. You edit the response file to control the silent installation.

The response file is located in the `responsefiles` directory at the top level of your WSRR product DVD or download directory. All the default paths specified in the response file are relative to the `responsefiles` directory. The file is named `template_response.xml`. The response file is called from the silent installation script and should not be moved or renamed (although you can make a copy of the original response file before you modify it so that you can revert to the default version if required).

The response file passes parameters to the IM silent installation command. The response file completes the following tasks:

- Defines the repositories that IM uses to locate install or upgrade files for IM, WebSphere Application Server, and WSRR.
- Upgrades IM and WebSphere Application Server, if required.
- Installs WSRR into a location relative to the WebSphere Application Server.

You must modify certain fields in the response file before you run the script file that calls it.

a) Specify the repository locations. A repository is a location that stores data for installing, modifying, rolling back, updating, or uninstalling packages. You modify the response file to specify the location of the following repositories, or you can leave the fields unchanged to accept the default values.

b) **IM Repository Location**
Specify the location of the repository that holds installation or update data for IM itself. By default this is set to the IM directory at the top level of your WSRR product DVD or download directory. For example:

```
<repository location='../IM/' temporary='true'/>
```

WSRR Repository Location
Specify the location of the repository that holds installation data for WSRR. By default this field is set to the repository directory at the top level of your WSRR product DVD or download directory. For example:

```
<repository location="../repository/" />
```

You can also add a repository entry for the location that holds fix packs for your version of WSRR. If you add this repository in addition to the installation repository for WSRR, then the silent installation will install the latest version, including available fix packs. For example:

```
<repository location="../8501_fp" />
```

c) Specify the location of Installation Manager (IM). You specify the location where IM is installed. The response file contains a placeholder path which you must edit. You can specify the standard installation location for IM:

```
<profile kind='self'
installLocation='/home/<INSTUSER>/IBM/InstallationManager/ecl
ipse' id='IBM Installation Manager'>
  <data key='eclipseLocation'
value='/home/<INSTUSER>/IBM/InstallationManager/eclipse'/>
</profile>
```

You should also edit the Eclipse cache location to match the path that you specified for the IM installation.

```
<preference value="/home/<INSTUSER>/IBM/InstallationManager
/eclipseCache"
name="com.ibm.cic.common.core.preferences.eclipseCache" />
```

Note: You should not edit the `<install>` section which instructs the silent install script to install IM. If IM is already installed, this step is skipped.

d) Specify the location of WebSphere Application Server (this directory is also used to install WSRR). Replace `<WAS_HOME>` in the example below.
Example:

```
<profile installLocation='<WAS_HOME>' id='IBM WebSphere
Application Server - ND'>
  <data key='eclipseLocation' value='<WAS_HOME>'/>
  <data key="cic.selector.nl" value="en" />
</profile>
```

The WebSphere Application Server profile name is set to "IBM WebSphere Application Server - ND" by default. If you have an existing WebSphere Application Server installation that uses that profile id, then you must edit the response file to specify an alternative name.

7.1.2 Run the silent installation

After you have created your response file, you run a command that calls the file.

Since `<INSTUSER>` is a non-root user, use the following command to start a silent installation:

```
<DVD_ROOT>/IM/userinstc -input
<DVD_ROOT>/responsefiles/template_response.xml -log
<LOG_FILE_PATH>/iminstall.log –acceptLicense
```

You must also ensure that the non-root user has write access to the installation directories specified in the response file, `template_response.xml`. They also require write access to the IM subdirectory of the install media and its contents.
The installation command reads the response file, installs packages as directed and writes a log file to the directory you specified. You should specify a temporary directory to write the log file to.

The following log entry constitutes a successful installation log:

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
</result>
```

Anything between the `<result>` tags points to an error during installation, and needs further investigation.

7.2 Create managed profile

You create the profile by running the manageprofiles command with a response file that defines the profile. Template response files can be found in the following directory `<WAS_HOME>/WSRR/install/profile-responsefiles`
A sample response file for a managed node is also shown in Appendix F: Example Custom Profile Manageprofiles Response File on page 63.

Creating a custom profile creates a managed node. This is federated to a deployment manager as part of profile creation (this requires the deployment manager to be running).

7.2.1 Create a managed node response file

Edit the `create-managed.txt` file to specify values for the following parameters (or accept the default values):
cellName
The name of the cell. This parameter is not required and can be left at the default value or removed entirely.
profileName
The name of the profile. The default profile name is based on the profile type and a trailing number, for example: `Custom01` The profile name must not contain spaces or

characters that are not valid such as: *, ?, ", <, >, ,, /, \, and |. The profile name that you choose must not be in use.

profilePath

The fully qualified path to the profile. `<NODE1_PROFILE_ROOT>` For example: `/devel/ServiceRegistry/profiles/Custom01`

templatePath

The fully qualified path to the managed.wsrr template file in the installation root directory. For example: `/devel/ServiceRegistry/profileTemplates/managed.wsrr`

nodeName

The node name for the managed server that you are creating, for example, `Custom01Node`.

hostName

The host name of the server that you are creating the profile on.

dmgrHost

The name of the host of the deployment manager.

dmgrPort

The SOAP port number of the deployment manager e.g. the default port is `8879`

dmgrAdminUserName

The administrative user ID for the deployment manager (not required if you set `enableAdminSecurity` to false).

dmgrAdminPassword

The password for the deployment manager (not required if you set `enableAdminSecurity` to false).

If you are unsure of the `dmgrPort` of the deployment manager then it can be found by looking in `<DMGR_PROFILE_ROOT>/logs/AboutThisProfile.txt`. It's the value listed as "Management SOAP connector port". Other values such as the `dmgrHost` can also be found in the same file.

This is a sample deployment manager `AboutThisProfile.txt` file showing the values that can be found in it:

```
Application server environment to create: Management
Location: /devel/ServiceRegistry/profiles/Dmgr01
Disk space required: 30 MB
Profile name: Dmgr01
Make this profile the default: True
Node name: sampleCellManager01
Cell name: sampleCell01
Host name: sample.example.com
Enable administrative security (recommended): True
Administrative console port: 9060
Administrative console secure port: 9043
Management bootstrap port: 9809
Management SOAP connector port: 8879
Run Management as a service: False
```

7.2.2 Run manageprofiles

Run the manageprofiles command to create the custom profile:

```
<WAS_HOME>/bin/manageprofiles.sh -response
<WAS_HOME>/WSRR/install/profile-responsefiles/create-managed.txt
```

The command displays status as it runs. Wait for it to finish. Normal syntax checking on the response file applies as the file is parsed like any other response file. Individual values in the response file are treated as command-line parameters.

You can see that your profile creation completed successfully if you receive the message "`INSTCONFSUCCESS: Profile creation succeeded.`", and you can check the following log file:

```
<WAS_HOME>/logs/manageprofiles/Custom01_create.log
```

7.3 Start managed node

If the node is not already running, you can start the managed node from the command line.

Change directory to `<NODE1_PROFILE_ROOT>/bin`, for example:
`/devel/ServiceRegistry/profiles/Custom01/bin`
Enter the command:
`./startNode.sh`

After starting the managed node status messages are displayed.

8 Cluster Member #2

8.1 Install WSRR

To install silently, you create a response file that contains the information for your system, and then run an installation command that calls that response file. WSRR supplies a template response file for you to edit. A sample response file for this scenario is provided in Appendix B.

8.1.1 Creating a response file

The response file provides input to the Installation Manager (IM) when it is running a silent installation. You edit the response file to control the silent installation.

The response file is located in the `responsefiles` directory at the top level of your WSRR product DVD or download directory. All the default paths specified in the response file are relative to the `responsefiles` directory. The file is named `template_response.xml`. The response file is called from the silent installation script and should not be moved or renamed (although you can make a copy of the original response file before you modify it so that you can revert to the default version if required).

The response file passes parameters to the IM silent installation command. The response file completes the following tasks:

- Defines the repositories that IM uses to locate install or upgrade files for IM, WebSphere Application Server, and WSRR.
- Upgrades IM and WebSphere Application Server, if required.
- Installs WSRR into a location relative to the WebSphere Application Server.

You must modify certain fields in the response file before you run the script file that calls it.

a) Specify the repository locations. A repository is a location that stores data for installing, modifying, rolling back, updating, or uninstalling packages. You modify the response file to specify the location of the following repositories, or you can leave the fields unchanged to accept the default values.

b) **IM Repository Location**
Specify the location of the repository that holds installation or update data for IM itself. By default this is set to the IM directory at the top level of your WSRR product DVD or download directory. For example:

```
<repository location='../IM/' temporary='true'/>
```

WSRR Repository Location
Specify the location of the repository that holds installation data for WSRR. By default this field is set to the repository directory at the top level of your WSRR product DVD or download directory. For example:

```
<repository location="../repository/" />
```

You can also add a repository entry for the location that holds fix packs for your version of WSRR. If you add this repository in addition to the installation repository for WSRR, then the silent installation will install the latest version, including available fix packs. For example:

```
<repository location="../8501_fp" />
```

c) Specify the location of Installation Manager (IM). You specify the location where IM is installed. The response file contains a placeholder path which you must edit. You can specify the standard installation location for IM:

```
<profile kind='self'
installLocation='/home/<INSTUSER>/IBM/InstallationManager/ecl
ipse' id='IBM Installation Manager'>
  <data key='eclipseLocation'
value='/home/<INSTUSER>/IBM/InstallationManager/eclipse'/>
</profile>
```

You should also edit the Eclipse cache location to match the path that you specified for the IM installation.

```
<preference value="/home/<INSTUSER>/IBM/InstallationManager
/eclipseCache"
name="com.ibm.cic.common.core.preferences.eclipseCache" />
```

Note: You should not edit the `<install>` section which instructs the silent install script to install IM. If IM is already installed, this step is skipped.

d) Specify the location of WebSphere Application Server (this directory is also used to install WSRR). Replace `<WAS_HOME>` in the example below.
Example:

```
<profile installLocation='<WAS_HOME>' id='IBM WebSphere
Application Server - ND'>
  <data key='eclipseLocation' value='<WAS_HOME>'/>
  <data key="cic.selector.nl" value="en" />
</profile>
```

The WebSphere Application Server profile name is set to "IBM WebSphere Application Server - ND" by default. If you have an existing WebSphere Application Server installation that uses that profile id, then you must edit the response file to specify an alternative name.

8.1.2 Run the silent installation

After you have created your response file, you run a command that calls the file.

Since `<INSTUSER>` is a non-root user, use the following command to start a silent installation:

```
<DVD_ROOT>/IM/userinstc -input
<DVD_ROOT>/responsefiles/template_response.xml -log
<LOG_FILE_PATH>/iminstall.log –acceptLicense
```

You must also ensure that the non-root user has write access to the installation directories specified in the response file, `template_response.xml`. They also require write access to the IM subdirectory of the install media and its contents.
The installation command reads the response file, installs packages as directed and writes a log file to the directory you specified. You should specify a temporary directory to write the log file to.

The following log entry constitutes a successful installation log:

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
</result>
```

Anything between the `<result>` tags points to an error during installation, and needs further investigation.

8.2 Create managed profile

You create the profile by running the manageprofiles command with a response file that defines the profile. Template response files can be found in the following directory `<WAS_HOME>/WSRR/install/profile-responsefiles`
A sample response file for a managed node is also shown in Appendix F: Example Custom Profile Manageprofiles Response File on page 63.

Creating a custom profile creates a managed node. This is federated to a deployment manager as part of profile creation (this requires the deployment manager to be running).

8.2.1 Create a managed node response file

Edit the `create-managed.txt` file to specify values for the following parameters (or accept the default values):
cellName
The name of the cell. This parameter is not required and can be left at the default value or removed entirely.
profileName
The name of the profile. The default profile name is based on the profile type and a trailing number, for example: `Custom02` The profile name must not contain spaces or characters that are not valid such as: *, ?, ", <, >, ,, /, \, and |. The profile name that you choose must not be in use.
profilePath

The fully qualified path to the profile. <NODE2_PROFILE_ROOT> For example: /devel/ServiceRegistry/profiles/Custom02

templatePath

The fully qualified path to the managed.wsrr template file in the installation root directory. For example: /devel/ServiceRegistry/profileTemplates/managed.wsrr

nodeName

The node name for the managed server that you are creating, for example, Custom02Node.

hostName

The host name of the server that you are creating the profile on.

dmgrHost

The name of the host of the deployment manager.

dmgrPort

The SOAP port number of the deployment manager e.g. the default port is 8879

dmgrAdminUserName

The administrative user ID for the deployment manager (not required if you set enableAdminSecurity to false).

dmgrAdminPassword

The password for the deployment manager (not required if you set enableAdminSecurity to false).

If you are unsure of the variable values such as dmgrHost or dmgrPort for the deployment manager then they can be found by looking in <DMGR_PROFILE_ROOT>/logs/AboutThisProfile.txt.

8.2.2 Run manageprofiles

Run the manageprofiles command to create the custom profile:

```
<WAS_HOME>/bin/manageprofiles.sh -response
<WAS_HOME>/WSRR/install/profile-responsefiles/create-managed.txt
```

The command displays status as it runs. Wait for it to finish. Normal syntax checking on the response file applies as the file is parsed like any other response file. Individual values in the response file are treated as command-line parameters.
You can see that your profile creation completed successfully if you receive the message "INSTCONFSUCCESS: Profile creation succeeded.", and you can check the following log file:
<WAS_HOME>/logs/manageprofiles/Custom02_create.log

8.3 Start managed node

If the node is not already running, you can start the managed node from the command line.

Change directory to <NODE2_PROFILE_ROOT>/bin, for example:
/devel/ServiceRegistry/profiles/Custom02/bin
Enter the command:

```
./startNode.sh
```

After starting the managed node status messages are displayed.

9 Create Database

You must create a database for WSRR. You create the database by running scripts on your database server.

Before you can run the scripts to create a DB2 database manually, you must customize them.

The scripts for creating the WSRR database and tables are located in the following directory:

`<WAS_HOME>/WSRR/dbscripts/db2`

Note that you must have execute permission on all these files.

WSRR can be configured using up to three databases. They are the WSRR database, the activity logging database, and the service integration bus database. These databases can be separate or the activity logging, and service integration bus tables can be created in the WSRR database. These instructions assume that all tables are created in the WSRR database.

Note: Using WSRR without a service integration bus is not supported.
The user ID that you use to create the database and tables must have CREATETAB, CREATE PROCEDURE, and CREATE ANY SEQUENCE authority on the database, and USE privilege on the table space, as well as CREATEIN privilege on the schema. The user ID also must have EXECUTE privileges on the PMAPURI and PMAPSTR stored procedures. The user ID should also have IMPLICIT_SCHEMA authority in order for the required schema to be created.

The table below shows the different variables that are used in configuring the database for WSRR.

Table 1. Values used to create DB2 database in WSRR

Name	Description	Example value
`<DB_NAME>`	Database name. Uppercase and 8 characters or less.	`WSRRDB`
`<DB_SCHEMA>`	WSRR database schema. Uppercase and 8 characters or less.	`WSRR`
`<DB_USER>`	Database user ID. Must be lowercase. The user ID must exist.	`wsrr`
`<DB2INST>`	The DB2 instance.	`db2inst1`
`<DB2INSTOWNER>`	The DB2 instance owner	`db2inst1`
`<DB2PATH>`	The DB2 installation directory.	`/opt/IBM/db2/v10.5`

Table 1. Values used to create DB2 database in WSRR

Name	Description	Example value
<DB2TSDIR>	The directory to store the DB2 table spaces in. The <DB2TSDIR>directory must exist.	/home/db2inst1
<SIBDB_SCHEMA>	Service integration bus database schema. Uppercase and 8 characters or less.	WSRRSIB
<WSRR_SQLPATH>	Path to the SQL files. This is the directory on the database server to which you copy the scripts for creating the WSRR database and tables.	/tmp/wsrr_sql

(All names in <> refer to the values that you specify for the variables given in the table above. So, for example, <DB_NAME> might refer to "WSRRDB".)

9.1 Configure the database instance for WSRR

a) Make sure that your DB2 instance is configured for TCP/IP access. For example, as your DB2 instance owner (<DB2INSTOWNER>), and replacing <DB2INST> with your WSRR DB2 instance name, run:

```
db2set DB2COMM=tcpip
db2 update dbm cfg using SVCENAME db2c_<DB2INST>
```

b) Ensure that the db2c_<DB2INST> service is defined in /etc/services. An example of the definition in the /etc/services file is:

```
db2c_db2inst1 50000/tcp
```

c) After running the db2 update command, stop and restart the database (note that this will affect any other applications you are running that use DB2):

```
db2stop
db2start
```

d) To check that the value has been set correctly, run the following command and check the value for TCP/IP Service name:

```
db2 get database manager configuration
```

For example, after setting SVCENAME to db2c_db2inst1, the database manager configuration would include this line:

```
TCP/IP Service name (SVCENAME) = db2c_db2inst1
```

9.2 Configure the database and schemas

Configure the database and create the SIB schema before you create WSRR table space and tables:

a) Copy the files from `<WAS_HOME>/WSRR/dbscripts/db2` to your database server to the `<WSRR_SQLPATH>` directory.

b) Edit the `createWsrrDb.sql` file. Uncomment all lines starting -- db -- and replace `__DBNAME__` with `<DB_NAME>` (the name of the database you want to use). Replace all instances of `__DBSYSUSER__` with `<DB2INSTOWNER>` (the DB2 instance owner).

c) Edit the `createSibDb.sql` file. Uncomment the line beginning with `-- schema --`, and replace `__DBNAME__` with `<DB_NAME>` (the name of the database you wish to use), `__DBSCHEMA__` with `<SIBDB_SCHEMA>`, and `__DBSYSUSER__` with `<DB2INSTOWNER>` (the DB2 instance owner).

d) As your DB2 instance owner (`<DB2INSTOWNER>`) run:

```
db2 connect to <DB_NAME>
db2 -tf <WSRR_SQLPATH>/createWsrrDb.sql
db2 -tf <WSRR_SQLPATH>/createSibDb.sql
```

e) Run the following script as your database instance owner (`<DB2INSTOWNER>`) to configure some essential database settings and to restart the database:

```
<WSRR_SQLPATH>/configureDb2.sh
```

9.3 Create the WSRR table spaces and the WSRR schema

a) Edit the file `<WSRR_SQLPATH>/createWsrrTablespace.sql`
Replace all instances of `__DBNAME__` with the value of `<DB_NAME>`
Replace all instances of `__DBSCHEMA__` with `<DB_SCHEMA>`
Replace all instances of `__DBSYSUSER__` with `<DB2INSTOWNER>`

b) As your DB2 instance owner run:

```
db2 connect to <DB_NAME>
db2 -tf <WSRR_SQLPATH>/createWsrrTablespace.sql
```

c) The schema creation is implicit as long as `<DB2INSTOWNER>` has IMPLICIT_SCHEMA authority so it is worth ensuring that your user ID does have that required permission.

9.4 Create the database tables for the WSRR database

a) In each of `createWsrrTables1.sql`, `createWsrrTables2.sql` and `createWsrrProcs.sql`:
Replace all instances of `__DBNAME__` with `<DB_NAME>`

Replace all instances of __DBSCHEMA__ with <DB_SCHEMA>
Replace all instances of __DBUSER__ with <DB_USER>

b) As <DB2INSTOWNER> run:

```
db2 connect to <DB_NAME>
db2 -tf <WSRR_SQLPATH>/createWsrrTables1.sql
db2 -tf <WSRR_SQLPATH>/createWsrrProcs.sql
db2 -tf <WSRR_SQLPATH>/createWsrrTables2.sql
db2 terminate
```

9.5 Create the database tables for the activity logging database

a) In <WSRR_SQLPATH>/createActTables.sql
Replace all instances of __DBSCHEMA__ with <DB_SCHEMA>
Replace all instances of __DBUSER__ with <DB_USER>

b) As <DB2INSTOWNER> run:

```
db2 connect to <DB_NAME>
db2 -tf <WSRR_SQLPATH>/createActTables.sql
db2 terminate
```

9.6 Create the database tables for the service integration bus database

a) In <WSRR_SQLPATH>/createSibTables.sql
Replace all instances of __DBSCHEMA__ with <SIBDB_SCHEMA>
Replace all instances of __DBUSER__ with <DB_USER>

b) As <DB2INSTOWNER> run:

```
db2 connect to <DB_NAME>
db2 -tf <WSRR_SQLPATH>/createSibTables.sql
db2 terminate
```

10 Create Cluster

You create an empty cluster on the deployment manager, and add the managed nodes to that cluster.

You can create the cluster and add cluster members, by using a script file. You can run the scripts individually from the wsadmin prompt, see http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/index.jsp?topic=%2Fcom.ibm.websphere.nd.multiplatform.doc%2Fae%2Ftxml_clusters.html for more details.

10.1 Create a new Cluster

Follow the instructions in http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/topic/com.ibm.websphere.nd.multiplatform.doc/ae/txml_nomember.html to create a new cluster without members.

10.2 Add Cluster Members

Follow the instructions in http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/topic/com.ibm.websphere.nd.multiplatform.doc/ae/txml_addcluster.html to create cluster members.

11 Deploy WSRR

The next step is to deploy WSRR to the cluster (the cluster must be stopped, see chapter 18 Stop Cluster on page 52 for instructions). WSRR is deployed by running two separate scripts on the deployment manager node:

```
<WAS_HOME>/WSRR/install/jython/deploywsrrbus.py
<WAS_HOME>/WSRR/install/jython/deploywsrrapp.py
```

The `deploywsrrbus.py` script deploys the service integration bus components, the `deploywsrrapp.py` script deploys the WSRR application. Note that you must run both deployment scripts to achieve a functional WSRR system. You must run the script to deploy the bus first. The following sections will explain how to run these scripts in more detail.

11.1 Cluster deployment parameters

The following table gives details of the script parameters:

Parameter	Description
`-wsrrhome`	The `<WAS_HOME>` directory.
`-cluster`	The name of the cluster to which to deploy WSRR. This parameter is only required when deploying to a cluster. The cluster must exist.
`-node`	The node name of the federated server to which to deploy WSRR. This parameter is only required when deploying to a federated server. The federated server must exist. Not used when deploying to a cluster.
`-server`	The server name of the federated server to which to deploy WSRR. This parameter is only required when deploying to a federated server. The federated server must exist. Not used when deploying to a cluster.
`-actdbname`	Activity logging database name
`-actdbuser`	User name to authenticate with the activity logging database
`-actdbpassword`	Password for activity logging database authentication
`-actdbserver`	The name of the server hosting the activity logging database
`-actdbport`	Activity logging database server port
`-actdbschema`	The database schema name for the activity logging database.
`-actdbconnectionlocation`	Specifies the activity logging database connection location. (Only required for DB2ZOS.)
`-sibdbuser`	User name to authenticate with the service integration bus database

Parameter	Description
-jmsuser	Specifies the user to use for authentication to the service integration bus
-jmspassword	Specifies the password for the user specified for jmsuser
-sibdbpassword	Password for service integration bus database authentication
-sibdbserver	The database server host name or IP address for the Service Integration Bus database. For example, -sibdbserver "localhost"
-sibdbport	The Service Integration Bus database server port number. Depending on the database you are using, you can specify a different port number instead of the default port number
-sibdbname	The name of the Service Integration Bus database
-sibdbschema	The database schema name for the service integration bus database
-sibdbconnectionlocation	Specifies the database connection location. (Only required for DB2ZOS.)
-dbuser	User name to authenticate with the WSRR database
-dbpassword	Password for WSRR database authentication
-dbtype	The type of DBMS being used for WSRR and activity logging databases. Can be one of: • DB2 for a DB2 Universal Database or a DB2 Express database • DB2ZOS for a DB2 for z/OS database • SQLSERVER for a Microsoft SQL Server database using a Microsoft driver • ORACLE10G for an Oracle 10g database • ORACLE11G for an Oracle 11g database
-dbdriverpath	The path to the database driver, For example, -dbdriverpath <WAS_HOME>/jdbcdrivers/DB2
-dbserver	The name of the server hosting the WSRR database
-dbname	WSRR database name
-dbport	WSRR database server port
-dbschema	The WSRR database schema name.
-dbfunctionpath	Used to qualify the WSRR DB2 for z/OS procedures. (Only required for DB2ZOS.)
-dbconnectionlocation	Specifies the WSRR database connection location. (Only required for DB2ZOS.)

Parameter	Description
-transporthost	For example, -transporthost localhost
-fromaddress	Specifies who email notifications should be sent to. For example, -fromaddress admin@wsrr
-apppath	Specifies the directory that contains the ServiceRegistry.ear file. For example, -apppath /devel/ServiceRegistry/installableApps
-adminuser	Specifies any users to be mapped to the Administrator J2EE role in WSRR. For example, -adminuser ALL_AUTHENTICATED
-admingroup	Specifies any groups to be mapped to the Administrator J2EE role in WSRR. For example, -admingroup NONE
-useruser	Specifies any users to be mapped to the User J2EE role in WSRR. For example, -useruser ALL_AUTHENTICATED
-usergroup	Specifies any groups to be mapped to the User J2EE role in WSRR. Fore example,-usergroup NONE
-runasuser	Specifies which user the enterprise application should be run as. Must be a member of the WebSphere Application Server Administrator role. For example, -runasuser WASADMIN
-runaspassword	Specifies the password for the user specified for -runasuser
-prefix	Specify a prefix value to deploy a different instance of WSRR within the cell. (You can deploy WSRR only once to a federated server or cluster, but you can deploy WSRR instances to different servers or clusters in the same cell.) By default, the -prefix value is an empty string. The prefix value is prepended to enterprise application name and other resources. Note that you must specify the same prefix value when you run the both scripts for this deployment.

11.2 Deploy WSRR Service Integration Bus

Use a command with the following syntax to run the deploywsrrbus.py script:

```
wsadmin.sh –username <WAS_ADMIN> –password <WAS_PWD> -lang jython
-f deploywsrrbus.py
      -wsrrhome <WSRR install directory>
      -cluster <cluster name>
      -sibdbuser <user name>
      -sibdbpassword <password>
```

```
-dbtype <database type>
-dbdriverpath <path>
-sibdbserver <server>
-sibdbport <port>
-sibdbname <database name>
-dbfunctionpath <path>
-sibdbschema <schema name>
-jmsuser <user name>
-jmspassword <password>
-prefix <prefix string>
-sibdbconnectionlocation <database connection location>
```

(Enter the command all on one line.)

For example, this command is run from the `<WAS_HOME>/WSRR/install/jython` directory on the deployment manager node:

```
<DMGR_PROFILE_ROOT>/bin/wsadmin.sh –username <WAS_ADMIN> –password
<WAS_PWD> -lang jython -f deploywsrrbus.py -wsrrhome
/devel/ServiceRegistry -cluster WSRRCluster -sibdbuser wsrr -
sibdbpassword dbpword1 -dbtype DB2 -dbdriverpath
/devel/ServiceRegistry/jdbcdrivers/DB2 -sibdbserver db2host -
sibdbport 50000 -sibdbname WSRRDB -sibdbschema WSRR -jmsuser
<WAS_ADMIN> -jmspassword <WAS_PWD>
```

11.3 Deploy WSRR Applications

Use a command with the following syntax to run the `deploywsrrapp.py` script:

```
wsadmin.sh –username <WAS_ADMIN> –password <WAS_PWD> –lang jython
-f deploywsrrapp.py
      -wsrrhome <WSRR install directory>
      -cluster <cluster name>
      -dbuser <user name>
      -dbpassword <password>
      -actdbuser <user name>
      -actdbpassword <password>
      -dbtype <database type>
      -dbdriverpath <path>
      -dbserver <database server name>
      -actdbserver <database server name>
      -dbport <port>
      -actdbport <port>
      -dbname <database name>
      -actdbname <database name>
      -dbschema <schema name>
      -dbfunctionpath <path>
      -dbconnectionlocation <database connection location>
      -actdbschema <schema name>
      -actdbconnectionlocation <database connection location>
      -transporthost <host name>
      -fromaddress <email address>
      -apppath <path>
      -adminuser <user name>
      -admingroup <group name>
      -useruser <user name>
      -usergroup <group name>
      -runasuser <user name>
      -runaspassword <password>
      -prefix <prefix string>
```

(Enter the command all on one line.)

For example, this command is run from the `<WAS_HOME>/WSRR/install/jython` directory on the deployment manager node:

```
<DMGR_PROFILE_ROOT>/bin/wsadmin.sh –username <WAS_ADMIN> –password
<WAS_PWD> -lang jython -f deploywsrrapp.py -wsrrhome
/devel/ServiceRegistry -cluster WSRRCluster -dbuser wsrr -
dbpassword dbpword1 -actdbuser wsrr -actdbpassword dbpword1 -
dbtype DB2 -dbdriverpath /devel/ServiceRegistry/jdbcdrivers/DB2 -
dbserver db2host -actdbserver db2host -dbport 50000 -actdbport
50000 -dbname WSRRDB  -actdbname WSRRDB -dbschema WSRR -
actdbschema WSRR -transporthost localhost -fromaddress
user1@example.com -apppath /devel/ServiceRegistry/installableApps
-adminuser ALL_AUTHENTICATED -admingroup NONE -useruser
ALL_AUTHENTICATED -usergroup NONE -runasuser <WAS_ADMIN> -
runaspassword <WAS_PWD>
```

12 Setup IBM HTTP Server

The instructions below will assume installation of IBM HTTP Server as non-root. However it should be noted that when running IHS as non-root there is a limitation in what ports it can use. So if you wish to use the default port 80 you will need to install IHS as root.

12.1 Install IBM HTTP Server, WebSphere Web Server Plugins and WebSphere Customization Toolbox

IBM HTTP Server, WebSphere Web Server Plugins and the WebSphere Customization Toolbox are included on the WebSphere Application Server Supplemental content DVDs or eImages supplied with WSRR.

Extract the 4 archives to the same directory to create a local Installation Manager repository from which you can install IBM HTTP Server. The following lines will create a new local directory called `WAS_Supplemental` and extract the 4 zip files into it.

```
mkdir WAS_Supplemental
cp CZM91ML.zip CZM94ML.zip CZM95ML.zip CZXR9ML.zip WAS_Supplemental
unzip CZM91ML.zip
unzip CZM94ML.zip
unzip CZM95ML.zip
unzip CZXR9ML.zip
rm CZM91ML.zip CZM94ML.zip CZM95ML.zip CZXR9ML.zip
```

You also need to obtain the install image for IBM Installation Manager 1.7.1 or greater. This can be copied off the WSRR media or downloaded from the IBM Installation Manager download site:
http://www.ibm.com/support/entry/portal/Recommended_fix/Software/Rational/IBM_Installation_Manager
The response file below will assume that the Installation Manager install image has been downloaded and extracted to an "`IM`" directory in the same parent directory as `WAS_Supplemental`.

To install silently, you create a response file that contains the information for your system, and then run an installation command that calls that response file. An example template response file is supplied in Appendix C: Example HTTP Server Installation Manager Response File on page 57 for you to edit.

12.1.1 Creating a response file

The response file provides input to the Installation Manager (IM) when it is running a silent installation. You edit the response file to control the silent installation.

The examples below will assume the response file is named `template_response.xml` and is in the same directory as the `WAS_Supplemental` repository created above (the parent directory for `WAS_Supplemental`).

The response file passes parameters to the IM silent installation command. The response file completes the following tasks:

- Defines the repositories that IM uses to locate install or upgrade files for IM, and IBM HTTP Server.
- Upgrades IM and IBM HTTP Server, if required.

You must modify certain fields in the response file before you run the script file that calls it.

a) Specify the repository locations. A repository is a location that stores data for installing, modifying, rolling back, updating, or uninstalling packages. You modify the response file to specify the location of the following repositories, or you can leave the fields unchanged to accept the default values.

   IM Repository Location
   Specify the location of the repository that holds installation or update data for IM itself. By default this is set to the IM directory at the top level of your WSRR product DVD or download directory. For example:

   ```
   <repository location='../IM/' temporary='true'/>
   ```

   IBM HTTP Server Repository Location
   Specify the location of the repository that holds installation data for the WAS Supplemental content. For example:

   ```
   <repository location="../WAS_Supplemental/" />
   ```

b) Specify the location of Installation Manager (IM). You specify the location where IM is or should be installed. The response file contains a placeholder path which you must edit. You can specify the standard installation location for IM:

   ```
   <profile kind='self'
   installLocation='/home/<INSTUSER>/IBM/InstallationManager/ecl
   ipse' id='IBM Installation Manager'>
     <data key='eclipseLocation'
   value='/home/<INSTUSER>/IBM/InstallationManager/eclipse'/>
   </profile>
   ```

   You should also edit the Eclipse cache location to match the path that you specified for the IM installation.

   ```
   <preference value="/home/<INSTUSER>/IBM/InstallationManager
   /eclipseCache"
   name="com.ibm.cic.common.core.preferences.eclipseCache" />
   ```

   Note: You should not edit the `<install>` section which instructs the silent install script to install IM. If IM is already installed, this step is skipped.

c) Specify the location of IBM HTTP Server. Replace `/devel/HTTPServer` in the example below with your target IBM HTTP Server installation directory.
   Example:

```
<profile installLocation='/devel/HTTPServer' id='IBM HTTP
Server'>
  <data key='eclipseLocation' value='/devel/HTTPServer'/>
  <data key="cic.selector.nl" value="en" />
</profile>
```

   The profile name is set to "IBM HTTP Server" by default. If you have an existing installation that uses that profile id, then you must edit the response file to specify an alternative name.

d) Specify the location of IBM WebSphere Plugins. Replace `/devel/WebSphere/Plugins` in the example below with your target IBM WebSphere Plugins installation directory.
   Example:

```
<profile installLocation='/devel/WebSphere/Plugins' id='IBM
WebSphere Plugins'>
  <data key='eclipseLocation'
value='/devel/WebSphere/Plugins'/>
  <data key="cic.selector.nl" value="en" />
</profile>
```

   The profile name is set to "IBM WebSphere Plugins" by default. If you have an existing installation that uses that profile id, then you must edit the response file to specify an alternative name.

e) Specify the location of IBM WebSphere Toolbox. Replace `/devel/WebSphere/Toolbox` in the example below with your target IBM WebSphere Toolbox installation directory.
   Example:

```
<profile installLocation='/devel/WebSphere/Toolbox' id='IBM
WebSphere Toolbox'>
  <data key='eclipseLocation'
value='/devel/WebSphere/Toolbox'/>
  <data key="cic.selector.nl" value="en" />
</profile>
```

   The profile name is set to "IBM WebSphere Toolbox" by default. If you have an existing installation that uses that profile id, then you must edit the response file to specify an alternative name.

12.1.2 Run the silent installation

After you have created your response file, you run a command that calls the file.

Since `<INSTUSER>` is a non-root user, use the following command to start a silent installation:

```
IM/userinstc -input template_response.xml -log
<LOG_FILE_PATH>/iminstall.log –acceptLicense
```

You must also ensure that the non-root user has write access to the installation directories specified in the response file, `template_response.xml`. They also require write access to the IM subdirectory of the install media and its contents.
The installation command reads the response file, installs packages as directed and writes a log file to the directory you specified. You should specify a temporary directory to write the log file to.

The following log entry constitutes a successful installation log:

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
</result>
```

Anything between the `<result>` tags points to an error during installation, and needs further investigation.

See ‘Installing IBM HTTP Server’ in the WebSphere Application Server Information Center for guidance:
http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/topic/com.ibm.websphere.ihs.doc/ihs/welc6miginstallihsdist.html

12.2 Install WAS

To use IBM HTTP Server in managed mode, managed by the deployment manager, then you also need to install WAS on the IHS node. The easiest way to accomplish that is to use the WSRR installation media but alter the response file slightly from previously to only install WAS and not WSRR.

In order to install silently, you create a response file that contains the information for your system, and then run an installation command that calls that response file. WSRR supplies a template response file for you to edit. An example response file for this scenario is supplied in Appendix D: Example WAS Installation Manager Response File on page 60.

12.2.1 Creating a response file

The response file provides input to the Installation Manager (IM) when it is running a silent installation. You edit the response file to control the silent installation.

The response file is located in the `responsefiles` directory at the top level of your WSRR product DVD or download directory. All the default paths specified in the response file are relative to the `responsefiles` directory. The file is named

`template_response.xml`. The response file is called from the silent installation script and should not be moved or renamed (although you can make a copy of the original response file before you modify it so that you can revert to the default version if required).

The response file passes parameters to the IM silent installation command. In this case, the response file completes the following tasks:

- Defines the repositories that IM uses to locate install or upgrade files for IM, WebSphere Application Server, and WSRR.
- Upgrades IM and WebSphere Application Server, if required.

You must modify certain fields in the response file before you run the script file that calls it.

a) Specify the repository locations. A repository is a location that stores data for installing, modifying, rolling back, updating, or uninstalling packages. You modify the response file to specify the location of the following repositories, or you can leave the fields unchanged to accept the default values.

   IM Repository Location
   Specify the location of the repository that holds installation or update data for IM itself. By default this is set to the IM directory at the top level of your WSRR product DVD or download directory. For example:

   ```
   <repository location='../IM/' temporary='true'/>
   ```

   WSRR Repository Location
   Specify the location of the repository that holds installation data for WSRR. By default this field is set to the repository directory at the top level of your WSRR product DVD or download directory. For example:

   ```
   <repository location="../repository/" />
   ```

   You can also add a repository entry for the location that holds fix packs for your version of WSRR. If you add this repository in addition to the installation repository for WSRR, then the silent installation will install the latest version, including available fix packs. For example:

   ```
   <repository location="../8501_fp" />
   ```

b) Specify the location of Installation Manager (IM). You specify the location where IM is installed. The response file contains a placeholder path which you must edit. You can specify the standard installation location for IM:

   ```
   <profile kind='self'
   installLocation='/home/<INSTUSER>/IBM/InstallationManager/eclip
   se' id='IBM Installation Manager'>
     <data key='eclipseLocation'
   value='/home/<INSTUSER>/IBM/InstallationManager/eclipse'/>
   </profile>
   ```

You should also edit the Eclipse cache location to match the path that you specified for the IM installation.

```
<preference value="/home/<INSTUSER>/IBM/InstallationManager
/eclipseCache"
name="com.ibm.cic.common.core.preferences.eclipseCache" />
```

Note: You should not edit the `<install>` section which instructs the silent install script to install IM. If IM is already installed, this step is skipped.

c) Specify the location of WebSphere Application Server (this directory is also used to install WSRR). Replace `<WAS_HOME>` in the example below.
Example:

```
<profile installLocation='<WAS_HOME>' id='IBM WebSphere
Application Server - ND'>
  <data key='eclipseLocation' value='<WAS_HOME>'/>
  <data key="cic.selector.nl" value="en" />
</profile>
```

The WebSphere Application Server profile name is set to "IBM WebSphere Application Server - ND" by default. If you have an existing WebSphere Application Server installation that uses that profile id, then you must edit the response file to specify an alternative name.

d) Edit the <install> section to remove the WSRR offering. It should be altered to look like the example below:

```
<install>
   <offering profile="IBM WebSphere Application Server - ND"
   id="com.ibm.websphere.ND.v85" />
</install>
```

12.2.2 Run the WAS silent installation

After you have created your response file, you run a command that calls the file.

Since <INSTUSER> is a non-root user, use the following command to start a silent installation:

```
<DVD_ROOT>/IM/userinstc -input
<DVD_ROOT>/responsefiles/template_response.xml -log
<LOG_FILE_PATH>/iminstall.log –acceptLicense
```

You must also ensure that the non-root user has write access to the installation directories specified in the response file, template_response.xml. They also require write access to the IM subdirectory of the install media and its contents.
The installation command reads the response file, installs packages as directed and writes a log file to the directory you specified. You should specify a temporary directory to write the log file to.

The following log entry constitutes a successful installation log:

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
</result>
```

Anything between the <result> tags points to an error during installation, and needs further investigation.

12.3 Create a managed profile

You create the profile by running the manageprofiles command with a response file that defines the profile. Template response files can be found in the following directory
<WAS_HOME>/WSRR/install/profile-responsefiles
A sample response file for a managed node is also shown in Appendix F: Example Custom Profile Manageprofiles Response File on page 63.

Creating a custom profile creates a managed node. This is federated to a deployment manager as part of profile creation (this requires the deployment manager to be running).

12.3.1 Create a managed node response file

Edit the `create-managed.txt` file to specify values for the following parameters (or accept the default values):

cellName
The name of the cell. This parameter is not required and can be left at the default value or removed entirely.

profileName
The name of the profile. The default profile name is based on the profile type and a trailing number, for example: `Custom03` The profile name must not contain spaces or characters that are not valid such as: *, ?, ", <, >, ,, /, \, and |. The profile name that you choose must not be in use.

profilePath
The fully qualified path to the profile. `<NODE3_PROFILE_ROOT>` For example: `/devel/AppServer/profiles/Custom03`

templatePath
The fully qualified path to the managed.wsrr template file in the installation root directory. For example: `/devel/AppServer/profileTemplates/managed.wsrr`

nodeName
The node name for the managed server that you are creating, for example, `Custom01Node`.

hostName
The host name of the server that you are creating the profile on.

dmgrHost
The name of the host of the deployment manager.

dmgrPort
The SOAP port number of the deployment manager e.g. the default port is `8879`

dmgrAdminUserName
The administrative user ID for the deployment manager(not required if you set `enableAdminSecurity` to false).

dmgrAdminPassword
The password for the deployment manager(not required if you set `enableAdminSecurity` to false).

If you are unsure of the variable values such as `dmgrHost` or `dmgrPort` for the deployment manager then they can be found by looking in `<DMGR_PROFILE_ROOT>/logs/AboutThisProfile.txt`.

12.3.2 Run manageprofiles

Run the manageprofiles command to create the custom profile:

```
<WAS_HOME>/bin/manageprofiles.sh -response
<WAS_HOME>/WSRR/install/profile-responsefiles/create-managed.txt
```

The command displays status as it runs. Wait for it to finish. Normal syntax checking on the response file applies as the file is parsed like any other response file. Individual values in the response file are treated as command-line parameters.

You can see that your profile creation completed successfully if you receive the message "INSTCONFSUCCESS: Profile creation succeeded.", and you can check the following log file:
<WAS_HOME>/logs/manageprofiles/Custom01_create.log

12.3.3 Start managed node

You can start the managed node from the command line.

Change directory to <NODE3_PROFILE_ROOT>/bin, for example:
/devel/AppServer/profiles/Custom03/bin
Enter the command:
./startNode.sh

After starting the managed node status messages are displayed.

12.4 Configure the Web Server Plug-in

After installing the web server plug-in, you need to configure it. The Web Server Plug-in Configuration Tool in the WebSphere Customization Toolbox is used for configuring web server plug-ins. The Web Server Plug-in Configuration Tool creates one or more configurations for the web server plug-ins that can direct requests from a web client through the web server, and then interact with applications running on an application server. The Web Server Plug-in Configuration Tool edits the configuration file or files for a web server by creating directives that point to the location of the binary plug-in module and the plug-in configuration file.

a) Create a response file for use with the WebSphere Plugin Configuration Tool by creating a file (for example called /devel/WebSphere/Plugins/responsefile.txt) with content similar to this example:

```
configType=local_distributed
enableAdminServerSupport=true
enableUserAndPass=true
enableWinService=false
ihsAdminCreateUserAndGroup=false
ihsAdminPassword=passw0rd
ihsAdminPort=8008
ihsAdminUnixUserGroup=ihsgrp
ihsAdminUnixUserID=ihsadmin
ihsAdminUserID=wasadmin
mapWebServerToApplications=true
profileName=Custom01
wasExistingLocation=/devel/AppServer
webServerConfigFile1=/devel/HTTPD/conf/httpd.conf
webServerDefinition=wsrrwebserver
webServerHostName=ihshost.example.com
webServerInstallArch=64
webServerPortNumber=80
webServerSelected=ihs
webServerType=IHS
```

Replace the values as appropriate.
The ihsAdminUnixUserGroup (`ihsgrp`) and ihsAdminUnixUserID (`ihsadmin`) must be created manually before the script is run. This has to be done as root.

b) This response file can then be fed into the WebSphere Plugin Configuration Tool:

```
/devel/WebSphere/Toolbox/WCT/wctcmd.sh -tool pct -defLocPathname
/devel/WebSphere/Plugins -defLocName wsrrwebserver -response
/devel/WebSphere/Plugins/responsefile.txt
```

Where `/devel/WebSphere/Toolbox` is the WebSphere Customization Toolbox installation location, `/devel/WebSphere/Plugins` is the WebSphere Plugins installation location, `wsrrwebserver` is a name to give this specific webserver definition and `/devel/WebSphere/Plugins/responsefile.txt` is the responsefile location.

During configuration, the following tasks are completed:

- A default temporary plug-in configuration file is created and placed into the location specified.
- The web server configuration file is updated with the plug-in configuration, including the location of the plug-in configuration file.
- A script is generated to define the web server to WebSphere Application Server. The script is located in:
  `<PLUGIN_HOME>/bin/configure<web_server_name>`
  For example:
  `/devel/WebSphere/Plugins/bin/configure_wsrrwebserver.sh`

c) The managed node should be running, and then execute the script:
`/devel/WebSphere/Plugins/bin/configure_wsrrwebserver.sh`

Note: Always open a new command window in which to execute the `configure_<web_server_name>` script. There is a potential conflict between a shell environment variable, the WAS_USER_SCRIPT variable, and the real default profile. The script always works against the default profile. However, if the WAS_USER_SCRIPT environment variable is set, a conflict arises as the script attempts to work on the profile identified by the variable.

d) When the web server is defined to WebSphere Application Server, the plug-in configuration file is generated automatically. For the IBM HTTP Server, the new plug-in file will be propagated to the web server automatically. For other web server types, you need to propagate the new plug-in configuration file to the web server.

12.5 Configure IHS to allow encoded slashes in URLs

Set the HTTP Server to allow encoded slashes in a URL. The following steps describe how to do this for IBM HTTP Server.

a) Edit your HTTP server configuration file (e.g. `/devel/HTTPD/conf/httpd.conf`).

b) Add the `AllowEncodedSlashes` directive to the global environment and any virtual hosts defined in the configuration. Set its value to "`On`". An example of the directive in the global environment section of the `httpd.conf` file is:

```
#
# KeepAlive: Whether or not to allow persistent connections
(more than
# one request per connection). Set to "Off" to deactivate.
#
KeepAlive On

# allow encoded slashes
AllowEncodedSlashes On
```

c) Restart the HTTP server.
To restart IBM HTTP Server use the `apachectl` command:

```
/devel/HTTPD/bin/apachectl stop
/devel/HTTPD/bin/apachectl start
```

13 Start Cluster

This should be done from the Deployment Manager node.

Follow the instructions in
http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/topic/com.ibm.websphere.nd.multiplatform.doc/ae/txml_startcluster.html

14 Configure IHS for WSRR

In order to use the webserver, the web modules of the WSRR and WSRRCRE applications need mapping to it.

A jython script has been provided to do this for you (see Appendix G: Sample IHS Configuration Script on page 64).

The script will exchange security keys between the WebSphere Application Server cell and the web server. It will also map the web modules from the WSRR and WSRRCRE applications to the web server.

Use the `configureIHS.py` script to configure IHS. Call the following command from the `<DMGR_PROFILE_ROOT>/bin` directory:

```
cd <DMGR_PROFILE_ROOT>/bin
./wsadmin.sh –username <WAS_ADMIN> –password <WAS_PWD> -lang
jython -f <WAS_HOME>/WSRR/install/jython/configureIHS.py –cluster
'WSRRCluster' –webNodeName 'Custom03Node' –webServerName
'wsrrwebserver'
```

Where `WSRRCluster` is the cluster name, `WSRRCell` is the name of your cell, `Custom03Node` is the name of the WebSphere Application Server node the webserver is running on and `wsrrwebserver` is the name of the webserver.

The `nodeName` for the webserver node will have been defined in the manageprofiles response file used to create the managed profile for the webserver (`<NODE3_PROFILE_NAME>`). If you are unsure of the `nodeName` of the webserver node then it can be found by looking in `<NODE3_PROFILE_ROOT>/logs/AboutThisProfile.txt`.

15 Restart Cluster

You must restart the cluster after configuring the web server.

This should be done from the Deployment Manager node.

Follow the instructions in
http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/topic/com.ibm.websphere.nd.multiplatform.doc/ae/txml_startcluster.html

16 Load & Activate WSRR Configuration Profile

A WSRR configuration profile contains a complete set of WSRR configuration files. Configuration profiles are used for backup, restore and management of entire sets of WSRR configuration.

A WSRR configuration profile contains: web UI views to display registry content in a way that is suitable for its intended use, roles and perspectives to implement these views, classification systems to support your user-defined models, business domains and technical domains of relevance, lifecycles appropriate to the service lifecycle and its governance, and configuration of validators, modifiers, and notifiers to implement governance policies. It should not be confused with the word profile as used in conjunction with an IBM Installation Manager installation directory or a WebSphere Application Server node.

This should be done from the Deployment Manager node.

The cluster must be running for you to load and activate the configuration profile.

a) To load a configuration profile, run the following script on the deployment manager node from the `<WAS_HOME>/WSRR/admin/scripts_cell` directory:

   ```
   <DMGR_PROFILE_ROOT>/bin/wsadmin.sh –username <WAS_ADMIN> –
   password <WAS_PWD> -f loadConfigurationProfile.jacl -cluster
   WSRRCluster -filepath <WAS_HOME>/WSRR/config -filename
   GovernanceEnablementProfile_v85.zip -profilename GEP
   ```

   Where `WSRRCluster` is the name of the cluster.

b) To activate the configuration profile, run the following script on the deployment manager node from the `<WAS_HOME>/WSRR/admin/scripts_cell` directory:

   ```
   <DMGR_PROFILE_ROOT>/bin/wsadmin.sh –username <WAS_ADMIN> –
   password <WAS_PWD> -f activateConfigurationProfile.jacl -
   cluster WSRRCluster –profilename GEP
   ```

   Where `WSRRCluster` is the name of the cluster and `GEP` is the profile name that you specified when you loaded the configuration profile.

17 Installation Verification

In order to verify that everything is configured correctly point a web browser at the following URL:

Error! Hyperlink reference not valid.

Where `httpserver` is the name of your IBM HTTP Server node and `port` is the secure port value.

18 Stop Cluster

If you need to stop the cluster, this should be done from the Deployment Manager node.

Follow the instructions in
http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/topic/com.ibm.websphere.nd.multiplatform.doc/ae/txml_stopcluster.html

Additional Resources

The following resources might provide additional information if required.

WebSphere Service Registry and Repository

WebSphere Service Registry and Repository v8.5 Information Center
http://pic.dhe.ibm.com/infocenter/sr/v8r5/index.jsp

WSRR Information Portal
https://www.ibm.com/developerworks/mydeveloperworks/wikis/home?lang=en#/wiki/WebSphere%20Service%20Registry%20and%20Repository

WebSphere Application Server

WebSphere Application Server v8.5 Information Center
http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/index.jsp

WAS v8.5: Concepts, Planning, and Design Guide Redbook
http://www.redbooks.ibm.com/abstracts/sg248022.html

WAS v8.5: Administration and Configuration Redbook
http://www.redbooks.ibm.com/abstracts/sg248056.html

Installation Manager

IBM Installation Manager v1.7.1 Information Center
http://pic.dhe.ibm.com/infocenter/install/v1r7/index.jsp

DB2

IBM DB2 v10.5 Information Center
http://publib.boulder.ibm.com/infocenter/db2luw/v10r5/index.jsp

Appendix A: Command reference

Command	*Description*
`addNode.sh`	Federate this managed node into a deployment manager controlled cell
`db2`	IBM DB2 command line interface
`manageprofiles.sh`	Command to work with IBM WebSphere Application Server profiles
`startManager.sh`	Start a deployment manager
`startNode.sh`	Start a managed node
`tar`	GNU Tar command
`userinstc`	IBM Installation Manager non-root command line install command
`wsadmin.sh`	IBM WebSphere Application Server administration interface

Appendix B: Example WSRR Installation Manager Response File

This file is also included with this document as `WSRR-IM-response.txt`.

```
<?xml version="1.0" encoding="UTF-8"?>
<agent-input>

<!-- ##########################################################################
All repositories are listed here.
A repository can be either a local location or a live repository.

If you have a local repository, replace the appropriate value below.
You do not need to remove the repository from this section if you decide not to
install
that product.
##########################################################################-->
<server>
      <!-- ##################### IM Repository Location ######################-->
      <repository location='../IM/' temporary='true'/>
      <!-- ################## WSRR Repository Location ###################-->
      <repository location="../repository/" />
</server>
<!-- ### Set this value to false if you do not have direct internet access ###
-->
<preference value="false"
name="com.ibm.cic.common.core.preferences.searchForUpdates" />

<!-- ##########################################################################
This profile node defines where IBM Installation Manager (IM) is/or will be
installed.

If you want to modify where IM is installed modify both the
installLocation and eclipseLocation values to specify the correct directory
##########################################################################-->
<profile kind='self'
installLocation='/home/INSTUSER/IBM/InstallationManager/eclipse' id='IBM
Installation Manager'>
      <data key='eclipseLocation'
value='/home/INSTUSER/IBM/InstallationManager/eclipse'/>
</profile>
<!-- ##########################################################################
Modify to change the location of the eclipseCache (only if no cache has been
created yet).
##########################################################################-->
<preference value="/home/INSTUSER/IBM/InstallationManager/eclipseCache"
name="com.ibm.cic.common.core.preferences.eclipseCache" />

<!-- ##########################################################################
This installation node directs the IM installer to install IM.
You do not need to edit this line. If IM is already installed, this instruction
is skipped; if IM is not installed, this instruction installs it.
##########################################################################-->
<install>
      <offering features='agent_core,agent_jre' id='com.ibm.cic.agent'/>
</install>

<!-- ##########################################################################
```

```
This profile node defines where IBM WebSphere Application Server Network
Deployment & WSRR are to be installed

Modify the installLocation and eclipseLocation values to specify the correct
directory where the products are to be installed.

Both products must be installed to the same directory and both parameters must
point to the same directory.

If the specified profile ID exists, you must also change the profile ID.

################################################################################-->
<profile installLocation='/devel/ServiceRegistry' id='IBM WebSphere Application
Server - ND'>
      <data key='eclipseLocation' value='/devel/ServiceRegistry'/>
      <data key="cic.selector.nl" value="en" />
</profile>

<!-- ################################################################################
This installation node directs the IM installer to install IM-based offering.

The ID must match a valid offering ID of a repository that is specified in the
first section of this file.

See the online documentation for more information about modifying
this node.
################################################################################-->
<install>
      <offering profile="IBM WebSphere Application Server - ND"
id="com.ibm.websphere.ND.v85" />
      <offering profile="IBM WebSphere Application Server - ND"
id="com.ibm.wsrr.server" />
</install>

<!-- ################################################################################
Do not modify the properties in this section.
################################################################################-->
<preference value="30"
name="com.ibm.cic.common.core.preferences.connectTimeout" />
<preference value="30" name="com.ibm.cic.common.core.preferences.readTimeout"
/>
<preference value="0"
name="com.ibm.cic.common.core.preferences.downloadAutoRetryCount" />
<preference value="true" name="offering.service.repositories.areUsed" />
<preference value="false"
name="com.ibm.cic.common.core.preferences.ssl.nonsecureMode" />
<preference value="false"
name="com.ibm.cic.common.core.preferences.http.disablePreemptiveAuthentication"
/>
<preference value="true"
name="com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts" />
<preference value="false" name="PassportAdvantageIsEnabled" />
<preference value="true"
name="com.ibm.cic.common.core.preferences.import.enabled" />
</agent-input>
```

Appendix C: Example HTTP Server Installation Manager Response File

This response file will install IBM Installation Manager, IBM HTTP Server, WebSphere Web Server Plugins and the WebSphere Customization Toolbox.

This file is also included with this document as `IHS-IM-response.txt`.

```
<?xml version="1.0" encoding="UTF-8"?>
<agent-input>

<!-- ##########################################################################
All repositories are listed here.
A repository can be either a local location or a live repository.

If you have a local repository, replace the appropriate value below.
You do not need to remove the repository from this section if you decide not to
install
that product.
##########################################################################-->
<server>
      <!-- ##################### IM Repository Location ####################-->
      <repository location='../IM/' temporary='true'/>
      <!-- ############### HTTP Server Repository Location #################-->
      <repository location="../repository/" />
</server>
<!-- ### Set this value to false if you do not have direct internet access ###
-->
<preference value="false"
name="com.ibm.cic.common.core.preferences.searchForUpdates" />

<!-- ##########################################################################
This profile node defines where IBM Installation Manager (IM) is/or will be
installed.

If you want to modify where IM is installed modify both the
installLocation and eclipseLocation values to specify the correct directory
##########################################################################-->
<profile kind='self'
installLocation='/home/INSTUSER/IBM/InstallationManager/eclipse' id='IBM
Installation Manager'>
      <data key='eclipseLocation'
value='/home/INSTUSER/IBM/InstallationManager/eclipse'/>
</profile>
<!-- ##########################################################################
Modify to change the location of the eclipseCache (only if no cache has been
created yet).
##########################################################################-->
<preference value="/home/INSTUSER/IBM/InstallationManager/eclipseCache"
name="com.ibm.cic.common.core.preferences.eclipseCache" />

<!-- ##########################################################################
This installation node directs the IM installer to install IM.
You do not need to edit this line. If IM is already installed, this instruction
is skipped; if IM is not installed, this instruction installs it.
##########################################################################-->
<install>
      <offering features='agent_core,agent_jre' id='com.ibm.cic.agent'/>
```

```
</install>

<!-- ############################################################################
This profile node defines where IBM HTTP Server is to be installed

Modify the installLocation and eclipseLocation values to specify the correct
directory where the products are to be installed.

If the specified profile ID exists, you must also change the profile ID.

############################################################################-->
<profile installLocation='/devel/HTTPD' id='IBM HTTP Server'>
      <data key='eclipseLocation' value='/devel/HTTPD'/>
      <data key="cic.selector.nl" value="en" />
      <data key="user.ihs.allowNonRootSilentInstall" value="true" />
      <data key="user.import.profile" value="false" />
      <data key="user.ihs.http.server.service.name" value="none" /> <!-- Always
none if user.ihs.installHttpService = false Otherwise Unique Windows service
name -->
      <data key="user.ihs.httpPort" value="80" />
      <data key="user.ihs.installHttpService" value="false" />
</profile>

<profile installLocation='/devel/WebSphere/Plugins' id='IBM WebSphere Plugins'>
      <data key='eclipseLocation' value='/devel/WebSphere/Plugins'/>
      <data key="cic.selector.nl" value="en" />
</profile>

<profile installLocation='/devel/WebSphere/Toolbox' id='IBM WebSphere Toolbox'>
      <data key='eclipseLocation' value='/devel/WebSphere/Toolbox'/>
      <data key="cic.selector.nl" value="en" />
</profile>


<!-- ############################################################################
This installation node directs the IM installer to install IM-based offerings.

The ID must match a valid offering ID of a repository that is specified in the
first section of this file.

See the online documentation for more information about modifying
this node.
############################################################################-->
<install>
      <offering profile="IBM HTTP Server" id="com.ibm.websphere.IHS.v85" />
</install>

<install>
      <offering profile="IBM WebSphere Plugins" id="com.ibm.websphere.PLG.v85"
/>
</install>

<install>
      <offering profile="IBM WebSphere Toolbox" id="com.ibm.websphere.WCT.v85"
/>
</install>

<!-- ############################################################################
Do not modify the properties in this section.
############################################################################-->
```

```
<preference value="30"
name="com.ibm.cic.common.core.preferences.connectTimeout" />
<preference value="30" name="com.ibm.cic.common.core.preferences.readTimeout"
/>
<preference value="0"
name="com.ibm.cic.common.core.preferences.downloadAutoRetryCount" />
<preference value="true" name="offering.service.repositories.areUsed" />
<preference value="false"
name="com.ibm.cic.common.core.preferences.ssl.nonsecureMode" />
<preference value="false"
name="com.ibm.cic.common.core.preferences.http.disablePreemptiveAuthentication"
/>
<preference value="true"
name="com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts" />
<preference value="false" name="PassportAdvantageIsEnabled" />
<preference value="true"
name="com.ibm.cic.common.core.preferences.import.enabled" />
</agent-input>
```

Appendix D: Example WAS Installation Manager Response File

This file is also included with this document as `WAS-IM-response.txt`.

```
<?xml version="1.0" encoding="UTF-8"?>
<agent-input>


<!-- ##########################################################################
All repositories are listed here.
A repository can be either a local location or a live repository.

If you have a local repository, replace the appropriate value below.
You do not need to remove the repository from this section if you decide not to
install
that product.
##########################################################################-->
<server>
      <!-- ##################### IM Repository Location ######################-->
      <repository location='../IM/' temporary='true'/>
      <!-- ################## WSRR Repository Location ###################-->
      <repository location="../repository/" />
</server>
<!-- ### Set this value to false if you do not have direct internet access ###
-->
<preference value="false"
name="com.ibm.cic.common.core.preferences.searchForUpdates" />

<!-- ##########################################################################
This profile node defines where IBM Installation Manager (IM) is/or will be
installed.

If you want to modify where IM is installed modify both the
installLocation and eclipseLocation values to specify the correct directory
##########################################################################-->
<profile kind='self'
installLocation='/home/INSTUSER/IBM/InstallationManager/eclipse' id='IBM
Installation Manager'>
      <data key='eclipseLocation'
value='/home/INSTUSER/IBM/InstallationManager/eclipse'/>
</profile>
<!-- ##########################################################################
Modify to change the location of the eclipseCache (only if no cache has been
created yet).
##########################################################################-->
<preference value="/home/INSTUSER/IBM/InstallationManager/eclipseCache"
name="com.ibm.cic.common.core.preferences.eclipseCache" />

<!-- ##########################################################################
This installation node directs the IM installer to install IM.
You do not need to edit this line. If IM is already installed, this instruction
is skipped; if IM is not installed, this instruction installs it.
##########################################################################-->
<install>
      <offering features='agent_core,agent_jre' id='com.ibm.cic.agent'/>
</install>

<!-- ##########################################################################
```

```
This profile node defines where IBM WebSphere Application Server Network
Deployment is to be installed

Modify the installLocation and eclipseLocation values to specify the correct
directory where the products are to be installed.

If the specified profile ID exists, you must also change the profile ID.

##############################################################################-->
<profile installLocation='/devel/AppServer' id='IBM WebSphere Application
Server - ND'>
      <data key='eclipseLocation' value='/devel/AppServer'/>
      <data key="cic.selector.nl" value="en" />
</profile>

<!-- ############################################################################
This installation node directs the IM installer to install IM-based offering.

The ID must match a valid offering ID of a repository that is specified in the
first section of this file.

See the online documentation for more information about modifying
this node.
##############################################################################-->
<install>
      <offering profile="IBM WebSphere Application Server - ND"
id="com.ibm.websphere.ND.v85" />
</install>

<!-- ############################################################################
Do not modify the properties in this section.
##############################################################################-->
<preference value="30"
name="com.ibm.cic.common.core.preferences.connectTimeout" />
<preference value="30" name="com.ibm.cic.common.core.preferences.readTimeout"
/>
<preference value="0"
name="com.ibm.cic.common.core.preferences.downloadAutoRetryCount" />
<preference value="true" name="offering.service.repositories.areUsed" />
<preference value="false"
name="com.ibm.cic.common.core.preferences.ssl.nonsecureMode" />
<preference value="false"
name="com.ibm.cic.common.core.preferences.http.disablePreemptiveAuthentication"
/>
<preference value="true"
name="com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts" />
<preference value="false" name="PassportAdvantageIsEnabled" />
<preference value="true"
name="com.ibm.cic.common.core.preferences.import.enabled" />
</agent-input>
```

Appendix E: Example Deployment Manager Manageprofiles Response File

This file is also included with this document as `create-dmgr.txt`.

```
# WSRR manageprofiles sample response file
# Run this file using:
# manageprofiles -response <path to this file>

# For Windows please use double backslashes as path separators (\\)
# Please ensure there are no trailing space characters on the end of property
values.

# This file is for creating a new WSRR deployment manager.
# WSRR can then be deployed using the WAS AdminConsole.

create

# details of the profile
profileName=Dmgr01
profilePath=/devel/ServiceRegistry/profiles/Dmgr01
templatePath=/devel/ServiceRegistry/profileTemplates/dmgr.wsrr

# WAS security
enableAdminSecurity=true
adminUserName=wasadmin
adminPassword=wasadmin
```

Appendix F: Example Custom Profile Manageprofiles Response File

This file is also included with this document as `create-managed.txt`.

```
# WSRR manageprofiles sample response file
# Run this file using:
# manageprofiles -response <path to this file>

# For Windows please use double backslashes as path separators (\\)
# Please ensure there are no trailing space characters on the end of property
values.

# This file is for creating a new WSRR managed profile.

create

# details of the profile
profileName=Custom01
profilePath=/devel/ServiceRegistry/profiles/Custom01
templatePath=/devel/ServiceRegistry/profileTemplates/managed.wsrr

nodeName=Custom01Node
cellName=WSRRCell
hostName=node1host.example.com
# Identifies the SOAP port of the deployment manager.
# Specify this parameter and the dmgrHost parameter to federate a custom
profile as it is created.
# The deployment manager must be running and accessible.
# If you have enabled security or changed the default Java Management
Extensions (JMX) connector type,
# you cannot federate with the manageprofiles command. Use the addNode command
instead.
dmgrHost=dmgrhost.example.com
dmgrPort=8879
dmgrAdminUserName=wasadmin
dmgrAdminPassword=password
# Specifies the starting port number for generating and assigning all ports for
the profile.
startingPort=22000
```

Appendix G: Sample IHS Configuration Script

After installing WSRR v8.5 this script can also be found in `<WAS_HOME>/WSRR/install/jython/configureIHS.py`.

This file is also included with this document as `configureIHS.py`.

```
# begin_generated_IBM_copyright_prolog
#
# Licensed Materials - Property of IBM
# 5724-N72 5655-WBS
# (c) Copyright IBM Corp. 2008, 2014 All Rights Reserved
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with
# IBM Corp.
#
# end_generated_IBM_copyright_prolog

import sys
import os

# Simple routine sets calling routine variables from command line parameters
# Parameters:
#   argv -- sequence of parameters (from command line)
#     options -- array of [option, variable name] pairs
#              e.g., [ ["-u", "user"], ["-p", "password"] ]
#   varDict -- dictionary in which to set variable associated with option
def processArgs(argv, options, varDict) :
      programName = argv[0]
      for opt in options :
            varDict[opt[1]] = None  # initialize variables specified in options
      _processArgs(programName, argv[1:], options, varDict)
      for opt in options :
            if varDict[opt[1]] == None:
                  if opt[2] == "true":
                        usage(programName, "missing option " + opt[0], options)
                  else:
                        varDict[opt[1]] = "null"

# Auxilliary routine processes next two command line parameters,
#     then calls itself to process the rest of the command line
def _processArgs(programName, argv, options, varDict) :
      if len(argv) == 0 :
            return

      option = argv[0]
      if option == "-?" or option == "-h" or option == "--help" :
            usage(programName, None, options)

      if len(argv) <= 1 :
            usage(programName, "illegal argument count", options)
      value = argv[1]

      for opt in options :
            if option == opt[0] :
                  varDict[opt[1]] = value # set variable in dictionary as
specified in options
                  break
      else :
```

```
            usage(programName, "unknown option " + option, options)

      _processArgs(programName, argv[2:], options, varDict)

# Usage message, constructed from command line options
def usage(programName, message, options) :
      if message != None :
            print "Error exit: " + message
      usage = "usage: jython " + programName
      for opt in options :
            if opt[2] == "true":
                  usage += " " + opt[0] + " <" + opt[1] + ">"
            else:
                  usage += " [" + opt[0] + " <" + opt[1] + ">]"

      print usage
      sys.exit(1)


True=1==1
False=1==0

class Module:
    def __init__(self,name):
        self.name = name
        self.uri = ""
        self.oldMapping = ""

    def getMapping(self,webServer):
        print "Setting mapping for module "+self.name+" to
"+webServer+"+"+self.oldMapping
        return webServer+"+"+self.oldMapping


def checkIfAppExists(appName):
    application=AdminConfig.getid("/Deployment/"+appName+"/")
    return len(application) != 0

def mapToServers(webServer,appName):
    modulesToServer=AdminApp.view(appName,["-
MapModulesToServers"]).replace("\r\n","\n")
    modulesToServer=modulesToServer.split("\n")
    mappings=[]
    currentModule=None
    for line in modulesToServer:
        if line.startswith("Module:"):
            if(currentModule != None):
                print "Never found a URI/Server for module
"+currentModule.name+" - discarding it"
            currentModule = Module(line[len("Module:  "):])
        elif line.startswith("URI:"):
            if(currentModule == None):
                print "Found a URI without a module "+line+" - ignoring it"
            else:
                currentModule.uri = line[len("URI:  "):]
        elif line.startswith("Server:"):
            if(currentModule == None):
                print "Found a Server without a module "+line+" - ignoring it"
            else:
                currentModule.oldMapping = line[len("Server: "):].strip()
```

```
                print "Found module "+currentModule.name+" uri
"+currentModule.uri+" mapped to "+currentModule.oldMapping
                mappings.append(currentModule)
                currentModule = None
        elif line.strip() == "":
            pass
        else:
            print "Ignoring line: "+line

    mappings = map(lambda x,w=webServer: [x.name,x.uri,x.getMapping(w)],
mappings)

    print "Applying mappings"
    AdminApp.edit(appName,["-MapModulesToServers",mappings])

    print "Saving changes"
    AdminConfig.save()

    print "Sync nodes"
    nodeSyncObjects = AdminControl.queryNames("type=NodeSync,*")
    if len(nodeSyncObjects) > 0:
        for nodeSync in nodeSyncObjects.split(lineSeparator):
            try:
                print "Syncing "+str(nodeSync)
                syncResult = AdminControl.invoke(nodeSync, "sync", "")
                print "- returned "+str(syncResult)
            except:
                print "- threw exception"
    return True

def _splitlines(s):
    rv = [s]
    if '\r' in s:
        rv = s.split('\r\n')
    elif '\n' in s:
        rv = s.split('\n')
    if rv[-1] == '':
        rv = rv[:-1]
    return rv

def getDmgrNode():
    """Return config id of the Dmgr node"""
    node_ids = _splitlines(AdminConfig.list( 'Node' ))
    for node_id in node_ids:
        nodename = getNodeName(node_id)
        if nodeIsDmgr(nodename):
            return node_id
    return None

def getDmgrNodeName():
    """Return node name of the Dmgr node"""
    return getNodeName(getDmgrNode())

def getNodeName(node_id):
    """Get the name of the node with the given config object ID"""
    return AdminConfig.showAttribute(node_id, "name")

def nodeIsIHS( nodename ):
    """Returns true if the node is IHS."""
    # Note: This method queries whether variable WAS_INSTALL_ROOT is defined.
    # This is a weak technique for identifying an IHS node.
```

```
    # Hopefully a more robust mechanism can be found in the future.
    return None == getWasInstallRoot(nodename)

def getNodeId( nodename ):
    """Given a node name, get its config ID"""
    return AdminConfig.getid( '/Cell:%s/Node:%s/' % ( AdminControl.getCell(),
nodename ) )

def nodeIsDmgr( nodename ):
    """Return true if the node is the deployment manager"""
    return nodeHasServerOfType( nodename, 'DEPLOYMENT_MANAGER' )

def nodeHasServerOfType( nodename, servertype ):
    node_id = getNodeId(nodename)
    serverEntries = _splitlines(AdminConfig.list( 'ServerEntry', node_id ))
    for serverEntry in serverEntries:
        sType = AdminConfig.showAttribute( serverEntry, "serverType" )
        if sType == servertype:
            return 1
    return 0

def getNodeVariable(nodename, varname):
    """Return the value of a variable for the node -- or None if no such
variable or not set"""
    vmaps = _splitlines(AdminConfig.list('VariableMap', getNodeId(nodename)))
    if 0 < len(vmaps):  # Tolerate nodes with no such maps, for example, IHS
nodes.
        map_id = vmaps[-1] # get last one
        entries = AdminConfig.showAttribute(map_id, 'entries')
        # this is a string '[(entry) (entry)]'
        entries = entries[1:-1].split(' ')
        for e in entries:
            name = AdminConfig.showAttribute(e,'symbolicName')
            value = AdminConfig.showAttribute(e,'value')
            if name == varname:
                return value
    return None

def getWasInstallRoot(nodename):
    """Return the absolute path of the given node's WebSphere installation"""
    return getNodeVariable(nodename, "WAS_INSTALL_ROOT")

def getWasProfileRoot(nodename):
    """Return the absolute path of the given node's profile directory"""
    return getNodeVariable(nodename, "USER_INSTALL_ROOT")


def listAppServerNodes():
    """Returns a list of nodes excluding dmgr and IHS nodes"""
    m = "listAppServerNodes:"
    node_ids = _splitlines(AdminConfig.list( 'Node' ))
    result = []
    for node_id in node_ids:
        nodename = AdminConfig.showAttribute(node_id, "name")
        if not nodeIsDmgr(nodename) and not nodeIsIHS(nodename):
            result.append(nodename)
    if 0 == len(result):
        sop(m,"Warning. No non-manager/non-IHS nodes are defined!!!")
    return result
```

```
#-------------------------------------------------------------------------------
---------
#
# Main
#
#-------------------------------------------------------------------------------
---------

options = [
      [ "-cluster", "clusterName", "true" ],
      [ "-webNodeName", "webNodeName", "true" ],
      [ "-webServerName", "webServerName", "true" ],
      [ "-prefix", "prefix", "false" ]
]

## process command line arguments
sys.argv.insert(0,"configureIHS.py")
processArgs(sys.argv, options, globals())
cellName = AdminControl.getCell()
#get list of cluster members
clusterID = AdminConfig.getid("/ServerCluster:"+clusterName+"/" )
temp = AdminConfig.showAttribute(clusterID, "members" )
temp = temp.replace('[','')
temp = temp.replace(']','')
memberList = temp.strip().split(" ")

webServerId =
AdminConfig.getid('/Node:'+webNodeName+'/Server:'+webServerName+'/')
webNodeId = AdminConfig.getid('/Node:'+webNodeName+'/')
webServerHost=AdminConfig.showAttribute(AdminConfig.list("ServerIndex",
webNodeId),"hostName")
version=AdminTask.getNodeBaseProductVersion('[-nodeName '+webNodeName+']')
mapping = AdminConfig.getObjectName(webServerId)
dmgrNodeName = getDmgrNodeName()
configDir = getWasProfileRoot(dmgrNodeName)+'/config'

if prefix == None
      prefix = ""

print " ----------------------"
print " -- mapping modules."
print " ----------------------"
apps = ['ServiceRegistry', 'WSRRReportViewer', 'WSRRCRE']
for appName in apps:
      appName = prefix+appName
    if checkIfAppExists(appName):
        if mapToServers(mapping,appName):
            print "Modules mapped successfully for "+appName
        else:
            print "Map modules failed for "+appName
    else:
        print "Application "+appName+" not installed so mapping unchanged"

print " ----------------------"
print " -- module mapping completed."
print " ----------------------"

if os.path.isfile('/tmp/key'):
    os.remove('/tmp/key')

print " ----------------------"
```

```
print " -- Extracting certificate from cell"
print " ----------------------"
AdminTask.extractSignerCertificate('[ -keyStoreName CellDefaultTrustStore -
keyStoreScope (cell):' +cellName+' -certificateFilePath /tmp/key -base64Encoded
true -certificateAlias root ]')

print " ----------------------"
print " -- Importing Certificate from cell to webserver"
print " ----------------------"
AdminTask.addSignerCertificate('[-keyStoreName CMSKeyStore -keyStoreScope
(cell):'+cellName+':(node):'+webNodeName+':(server):'+webServerName+' -
certificateFilePath /tmp/key -base64Encoded true -certificateAlias CELL'
+cellName+' ]')
os.remove('/tmp/key')

for node in listAppServerNodes():
    print " ----------------------"
    print " -- Extracting certificate from %s " % (node)
    print " ----------------------"
    AdminTask.extractSignerCertificate('[ -keyStoreName NodeDefaultTrustStore -
keyStoreScope (cell):'+cellName+':(node):'+node+' -certificateFilePath /tmp/key
-base64Encoded true -certificateAlias default ]')
    print " ----------------------"
    print " -- Importing Certificate from %s to webserver" % (node)
    print " ----------------------"
    AdminTask.addSignerCertificate('[-keyStoreName CMSKeyStore -keyStoreScope
(cell):'+cellName+':(node):'+webNodeName+':(server):'+webServerName+' -
certificateFilePath /tmp/key -base64Encoded true -certificateAlias NODE'
+node+' ]')
    AdminConfig.save()
    os.remove('/tmp/key')
#endfor

AdminConfig.save()

print " ----------------------"
print " -- Pushing the keystore to the webserver"
print " ----------------------"
AdminControl.invoke("WebSphere:name=PluginCfgGenerator,process=dmgr,platform=co
mmon,node=%s,version=%s,type=PluginCfgGenerator,mbeanIdentifier=PluginCfgGenera
tor,cell=%s,spec=1.0" % (dmgrNodeName,version,cellName), "propagateKeyring",
"[%s %s %s %s]" % (configDir, cellName , webNodeName, webServerName),
"[java.lang.String java.lang.String java.lang.String java.lang.String]")
AdminConfig.save()

print " ----------------------"
print " -- Syncing nodes"
print " ----------------------"
nodeSyncObjects = AdminControl.queryNames("type=NodeSync,*")
if len(nodeSyncObjects) > 0:
    for nodeSync in nodeSyncObjects.split(lineSeparator):
        try:
            print "Syncing "+str(nodeSync)
            syncResult = AdminControl.invoke(nodeSync, "sync", "")
            print "- returned "+str(syncResult)
        except:
            print "- threw exception"

print " ----------------------"
print " -- Restarting webserver"
print " ----------------------"
```

```
AdminControl.invoke("WebSphere:name=WebServer,process=dmgr,platform=common,node
=%s,version=%s,type=WebServer,mbeanIdentifier=WebServer,cell=%s,spec=1.0" %
(dmgrNodeName,version,cellName), 'stop', "[ %s %s %s ]" % (cellName ,
webNodeName, webServerName), "[java.lang.String java.lang.String
java.lang.String]")
AdminControl.invoke("WebSphere:name=WebServer,process=dmgr,platform=common,node
=%s,version=%s,type=WebServer,mbeanIdentifier=WebServer,cell=%s,spec=1.0" %
(dmgrNodeName,version,cellName), 'ping', "[ %s %s %s ]" % (cellName ,
webNodeName, webServerName), "[java.lang.String java.lang.String
java.lang.String]")
AdminControl.invoke("WebSphere:name=WebServer,process=dmgr,platform=common,node
=%s,version=%s,type=WebServer,mbeanIdentifier=WebServer,cell=%s,spec=1.0" %
(dmgrNodeName,version,cellName), 'start',"[ %s %s %s ]" % (cellName ,
webNodeName, webServerName), "[java.lang.String java.lang.String
java.lang.String]")
AdminControl.invoke("WebSphere:name=WebServer,process=dmgr,platform=common,node
=%s,version=%s,type=WebServer,mbeanIdentifier=WebServer,cell=%s,spec=1.0" %
(dmgrNodeName,version,cellName), 'ping', "[ %s %s %s ]" % (cellName ,
webNodeName, webServerName), "[java.lang.String java.lang.String
java.lang.String]")
print " ------------------------"
print " -- Complete."
print " ------------------------"
```