IBM UrbanCode Deploy

# Middleware Configuration for WebSphere Application Server plug-in

*Version 6.0 This information is under development and can change at any time; the released version will be available on ibm.com.*

**Note**

Before using this information and the product it supports, read the information in “Notices,” on page 23.

This edition applies to version 6.0 of Middleware Configuration for Websphere Application Server plug-in for IBM UrbanCode Deploy and to all subsequent releases and modifications until otherwise indicated in new editions.

# Contents

# Middleware Configuration for WebSphere plug-in

The Middleware Configuration for WebSphere® plug-in includes steps that deploy configuration templates of IBM® WebSphere Application Server that you generate from an existing cell.

## Requirements

This plug-in requires the following software environments:
- UrbanCode Deploy version 6.0.1.4 or later
- WebSphere Application Server 6.1 or later

## Overview

This plug-in allows you to automate the process of deploying WebSphere configurations and applications to different environments (for example, development, test, and production environments. The following steps provide an overview of the deployment process you follow.
1. Set up UrbanCode Deploy to support the process. Load plug-ins and install an agent on the host where the WebSphere installation resides.
2. Create a component using the configuration template.
3. Capture topology information from an existing WebSphere installation and model it in Resources. A resource tree represents scopes in the installation.
4. Map the resources (scopes) to manage to the configuration template component.
5. Associate the component with an application.
6. Associate other components needed for deployment, for example components containing EAR or WAR files (application binaries)
7. Associate the resources with an application environment.
8. Generate the configuration template
9. Deploy the configuration template to create new instances.

The process for using the plug-in is designed to support one application per scope (cell, cluster, server, or node). The process is not designed to support use cases where multiple applications are deployed within the same cell scope or the same cluster scope.

## Optional features

Additional sections describe the following features.

**Specifying multiple profiles**

Specifying multiple profiles enables you collect information for multiple cells, either for the top-level group or for the agent. You use a special property, `websphere.profilePath`, to specify the paths.

You **must** use this property if your WebSphere Application Server installation is in a non-default location.

See "Specifying multiple profiles with the websphere.profilePath property" on page 8.

**Using custom properties**

You can use custom properties on the component to differentiate resources across environments. The custom properties are used to generate *tokens* when you generate the templates. Tokens are used like variables to substitute values in configuration information that varies depending on environment. For example, you may use a different database or a different JDBC resource in each of the deployment environments. See “Using custom properties when deploying WebSphere configurations and applications” on page 9

**Comparing WebSphere configurations**

Comparing configurations allows you to see how a deployed cell needs to be adjusted to fit a norm defined in a configuration template.

See “Comparing WebSphere configurations” on page 15.

**Layered templates**

Use layered templates to organize configuration objects that are managed in multiple applications and managed by multiple teams.

See “Enabling distributed management using layered templates” on page 16.

## Limitations

**OnDemand Router servers and clusters not supported**

This plug-in cannot manage OnDemand Router servers and clusters, due to a known issue with WebSphere Application Server.

If you attempt to work with this type of server, you get the following message.

**Compare does not work with layered templates**

Workaround: Generate a template from the layered-template implementation, then use that template as a snapshot to compare.

```
[exec] WASX7017E: Exception received while running file "<MCWAS-plugin-root>/product/actions/configu
  [exec] java.lang.ArrayIndexOutOfBoundsException: java.lang.ArrayIndexOutOfBoundsException: 0
```

## Release History

Features and fixes have been provided for each release.

**Version 6**

Version 6 is supported with version 6.0.1.4 of UrbanCode Deploy.

- Layered templates: a method to support distributed management of configurations (requires version 6.0.1.4 of UrbanCode Deploy)
- Defect fixes

**Version 5**

Various defects were fixed for Version 5 of the plug-in, which is supported in UrbanCode Deploy 6.0.1.

**Version 4**

Version 4 of the plug-in is supported in UrbanCode Deploy 6.0.1.

- Support for multiple profiles on the same host. Support was added for multiple profiles for top-level resource group properties and multiple profiles for agent properties.
- Use of XPath to create tokens. The previous method of using name-value pairs in a promote.properties file is still supported.
- Updated method for importing information from a configuration: The method is called **QuickCapture** and is enabled by default. It can be disabled for debugging purposes. An example usage would be that generating the configuration template does not work after upgrading to this version of the Middleware Configuration for WebSphere plug-in
- Base template specification: If the **SkipConfigCapture** flag is enabled when you import the a template, an existing WebSphere Application Server configuration is used as a base template. Normally this flag is enabled in order to tokenize an existing template.

**Version 3**

The updates for Version 3 were internal and did not require documentation changes. Version 3 is supported with version 6.0.1 of UrbanCode Deploy.

**Version 2**

- Requirements: version 2 is supported with version 6.0.1 of UrbanCode Deploy. It is not supported in prior versions of Urbancode Deploy.
- Additional scopes: Support for the node and server scopes was added to the existing support for cell and cluster scopes.
- Automated generation of configuration templates: The automated process requires less connection setup and fewer steps than the previous manual, script-based process in version 1.0.
- Compare configurations: You can now compare a component (configuration template) to a live cell. This is useful for detecting when a live cell's configuration "drifts," or departs from the desired configuration. See “Comparing WebSphere configurations” on page 15.

## Deploying WebSphere configurations and applications

To deploy WebSphere configurations and applications, you use plug-ins to read information about a live deployment into a model in Resources. You then define components, applications, environments, and processes to automate deployments that create new instantiations of the modeled WebSphere environment.

### Before you begin

The following setup and preparation is required.

- Select or set up a WebSphere cell to use. The configuration template you create will use its configuration.
- Install an UrbanCode Deploy agent on the appropriate WebSphere host. Note that you need the agent name during the procedures.
  - For ND deployments, install the agent where DMGR is running.
  - For server deployments, install the agent where Base is running.
- Determine what tokens you need. Tokens are used to represent information that is different among deployment environments. For example, you might specify a different database server or JDBC resource in each of a development, test, and production environment. For each token, you need a name and an initial value.

You are prompted to enter the names and values at the end of the process. See "Using custom properties when deploying WebSphere configurations and applications" on page 9

## 1. Set up plug-ins in UrbanCode Deploy

### About this task

In this task you prepare the required UrbanCode Deploy resources.

### Procedure

1. Install the Middleware Configuration for Websphere plug-in.
   a. In UrbanCode Deploy, go to **Settings** > **Automation Plugins**.
   b. Click **Load Plugin**.
   c. In the dialog, click **Choose File**, then choose the `MCWASPlugin-<version>.zip`
   d. Click **Submit**.
2. Install the Application Deployment for WebSphere plug-in.
   a. In UrbanCode Deploy, go to **Settings** > **Automation Plugins**.
   b. Click **Load Plugin**.
   c. In the dialog, click **Choose File**, then choose the `ApplicationDeploymentForWebSphere-<version>.zip` file.
   d. Click **Submit**.

   During installation a component template is automatically created for Middleware Configuration for WebSphere. It is placed in the following file: `<plugin_home>/imports/componenttemplates/Middleware Configuration for WebSphere.json`.

## 2. Create a component for the configuration template

### About this task

The component is at the center of the deployment process. It is associated with a configuration template and with resources so that it holds a configuration definition to be deployed.

### Procedure

Create a component if one does not exist.

1. Click the **Components** tab.
2. Click **Create New Component**. Provide the following information:
   - **Name**: Enter a name for the component.
   - **Template**: select **Middleware Configuration for WebSphere**.
3. Click **Save**.

## 3. Model a WebSphere cell in Resources

### About this task

Using autodiscovery, you locate a WebSphere cell through the agent that is installed on the host where it is running. Using auto-configure, you populate the resource tree with data about the WebSphere cell. For non-ND environments, you model a server resource. The resource tree is the centralized source of configuration data. Multiple processes in UrbanCode Deploy can include components that contain resources selected from the resource tree.

### Procedure

1. In Urbancode Deploy, go to **Resources**.
2. Click **Create Top-Level Group** to create a group.
3. Optional: if you need to specify multiple profiles for the group or WebSphere Application Server is installed in a non-default location, add `websphere.profilePath` as a property on the group and specify the appropriate paths. See "Specifying multiple profiles with the websphere.profilePath property" on page 8 for details.
4. Add one or more agents, as required.
   a. Hover over the row for the resource group, click **Actions**, and select **Add agent**.
   b. Select the agent to add. Use the agent that is installed on the host for the WebSphere environment you are going to model.
   c. Wait 10 to 30 seconds, then click **Refresh**. A twisty is now next to the agent. When you expand it, there is a sub-resource cell, **WebSphereCell**.
   d. Hover over the row, then click **Edit**.
   e. Enter values for the following properties.

      If you want to use `soap.properties` to provide the values, you must leave the WebSphere User and WebSphere Password properties blank.
      - WebSphere Profile Path
      - WebSphere User
      - WebSphere Password

      Leave the **Cell Name** property blank.
   f. Click **Save**.
5. Set Auto configure options for **WebSphereCell**.
   a. Hover over the row for **WebSphereCell**, click **Actions**, then click **Auto Configure**.
   b. Click **No auto configure for resource**.
   c. Check **Websphere Topology Discovery** box.
   d. Click **OK**.
   e. Click **Save**.
   f. Wait 30-60 seconds, then click **Refresh**. A twisty is now next to **WebSphereCell**. Expand it and make sure the resource tree matches your WebSphere Application Server topology.
   g. On the **WebSphereCell** entry, click **Edit**. Check that **Cell Name** was filled in and is correct.

## 4. Map the desired resource to the configuration template component

### About this task

During mapping you choose a scope from the resource tree to be used for deployments. Note that a captured scope configuration can be applied only to a like scope. For example, if you capture configuration information for a cluster scope, it can only be used to deploy a cluster configuration.

### Procedure

1. Click **Resources** to view the resource tree for the desired WebSphere cell.
2. Add a component to the scope or scopes you want to use: cell, cluster, node, or server.
   a. Select the scope you want to manage. Hover over a row to select a single scope to show a menu, then click **Add component** in the menu.
   b. Select the component to use. You created it above.
   c. Click **Save**.

## 5. Associate the component with an application

### About this task

Create the application if it does not exist. The application is the container for deployment data.

### Procedure

1. In the **Dashboard**, click **Applications**.
2. Create the application if it does not exist.
   a. Click **Create New Application**.
   b. Enter an application name, then click **Save**.
3. Add the component to the application.
   a. In the **Dashboard**, click **Applications**.
   b. Click the application to use.
   c. Click the **Components** tab, then add the component that you created for the configuration template.

## 6. Associate other needed components with the application

### About this task

To deploy applications, you need to have components configured for the WAR or EAR files. Add them to the application.

### Procedure

1. In the **Dashboard**, click **Applications**.
2. Click the application to use.
3. Click the **Components** tab, then add the desired component.
4. Repeat for all necessary components.

## 7. Associate the resources with an application environment

### About this task

The resources define the configuration data to be deployed with the application.

### Procedure

1. In the **Dashboard**, click the **Applications** tab.
2. Click the application you created.
3. Create an application environment.
   a. Click **Create New Environment**.
   b. Enter an environment name, then click **Save**.

4. Add the resource group to the environment.
   a. In the application, click **Environments**.
   b. Click the environment name.
   c. Click **Add Base Resource**. Choose the group for your exemplar cell and environment.
5. Optional: view the environment properties for the component.
   a. In the **Dashboard**, click **Components**.
   b. Click the component.
   c. Click **Configuration**.
   d. Click **Environment Property Definitions**.

## 8. Generate a configuration template

### About this task

Run an application process to generate the configuration template.

### Procedure

1. In the application you created, click the **Processes** tab.
2. Define the process as follows.
   a. In the process design page, locate each component that you have added to the application.
   b. Click the component to show a **Generate Template** process step.
   c. Drag the **Generate Template** step to the process editor, then give the process a name.
   d. Click **Save**.
3. Run the application process to generate the template.
4. Enter information for the following fields.
   - **New Component Version**: Specify a new version to use. The configuration template files are associated with that version.
   - **SkipConfigCapture**: If this flag is enabled, an existing WebSphere Application Server configuration is used as a base template. Normally this flag is enabled in order to tokenize an existing template.
   - **UseQuickCapture**: Enabled by default. Leave it enabled unless there are problems generating the template. You may uncheck it to debug problems with generating the template.
5. Click **Submit**.

   Wait for the process to finish running before continuing.

## 9. Deploy the configuration template

### About this task

Create an application process of type **Deployment** for configuration and deployment, then run the process to test it.

### Procedure

1. In the **Dashboard**, click the **Applications** tab.
2. Click the application you created.
3. Click the **Processes** tab.
4. Define the process.

a. Drag and drop the **Install Component** step onto the process editor. Choose the **Configure WebSphere Application Server** component process.
b. Select the component you created
c. Choose the **Configure WebSphere Application Server (Template)** component process

5. Click **Save**.
6. Run the process to test the newly created component template.

## Specifying multiple profiles with the websphere.profilePath property

Specifying multiple profiles enables you collect information for multiple cells, either for the top-level group or for the agent.

You specify multiple profiles during deployment. You specify them in a `websphere.profilePath` property in one of two places:

- *Top-level group*: Set the `websphere.profilePath` here to collect profiles from multiple cells deployed across multiple hosts. All hosts must use the same installation directory for the cell.
- *Agent*: Set the `websphere.profilePath` here to collect profiles for multiple profiles on the same host.

### Values for the websphere.profilePath property

Specify one or more paths as the property value. Separate multiple paths with a comma. The paths can be one of the following:

- path to a profile. The path does not have to specify `/bin/wsadmin.sh`, but autodiscovery works with paths that do.
- path to a container directory of profiles. In this case the autodiscovery code loops over each first-level directory. It registers base and nd profiles. It skips node and server profiles.

When you specify multiple profiles, the following values are read during autodiscovery:

- SOAP port (read from `portdef.properties`)
- profile path
- install path

Example value with two container directories and one profile path:

```
"/opt/IBM/WebSphere/Profiles/,/opt/WAS/Profiles,/opt/IBM/profiles/dmgr"
```

**Restriction:** If you specify multiple container directories, there should be no duplicates in the profile names they contain. If autodiscover finds the same profile name, it overwrites the previously found profile name.

Example: `/opt/IBM/WebSphere/Profiles` and `/opt/WAS/Profiles` both contain a `dmgr` profile. A resource is created only for the second `dmgr` resource, because the first profile is overwritten.

If you encounter this situation, you can work around it by creating a separate top-level group and segregate the profile directories.

## Using custom properties when deploying WebSphere configurations and applications

Use custom properties on a component to differentiate configurations between environments.

### Before you begin

- Understand the process for deploying WebSphere configurations and applications. See “Deploying WebSphere configurations and applications” on page 3.

Set up and deploy WebSphere configurations and applications without custom properties. Examine the deployment to determine where you need to differentiate deployment environments.

### About this task

Typically multiple environments are used in a staged software development process. The environments are segregated, for example into Development, Test, and Production areas. Code is promoted from one area to the next after exit criteria are met.

Configuration data for a deployed WebSphere instance may need to vary according to the environment where it is deployed. For example, a different database may be used for backing a JDBC resource in each environment. Defining custom properties allows you to customize deployment into each environment.

You define custom properties on the component you create for the deployment. When you generate the template, the custom properties are scanned and tokens are created in the configuration template. The tokens are substituted with values from the environment during deployment.

Property values can be either an explicit value or an XPath expression. You typically use XPath if the value could be confused with another value or part of a value if you used a simple value. For example, specifying 80 (HTTP port) could be confused with another value (9080) and result in an incorrect token. Note that only some XPath functionality is available for this purpose.

### Procedure

1. Click **Components**.
2. Click on the component, then the **Configuration** tab, then **Enviroment Property Definitions**.
3. Add custom tokens prefixed with **websphere**. You can assign values directly (using key-value pairs) or you can use an XPath expression. You can also add default values to be substituted for the tokens during the configuration process for the environment.

   **Note:** These properties should be for values in your configuration that you know need to be different between your different websphere cells.

   a. Add properties using key-value pairs. These properties are ones that can use an explicit, unambiguous value. See “Using XPath to specify property values” on page 10 for a full example in the context of IBM UrbanCode Deploy.

For example, the database port may be different for your development and production environment. In this case you would create a property with the following name:

```
websphere.dbport.value=50000
```

The string `50000` is extracted from the configuration when it is read. It is replaced with the token `@dbport@`. The token is later replaced during deployment with the value for the environment where you are applying the configuration data.

b. Add properties that require an XPath expression for a value.

   For example, the following XPath expression locates and updates a value in the `jdbc.xml` data file. It looks for a DataSource with the name widgetDB and updates the J2EEResourceProperty attribute to 50000 where the name attribute is portNumber.

   ```
   websphere.dbport.xpath=//DataSource[@name='widgetDB']/J2EEResourcePropertySet/J2EEResourceProp
       [@name='portNumber']/@value
   ```

   **Why use XPath?**

   The token creation process uses a simple string substitution method. In some cases tokens can be created incorrectly because one property is represented as a property that is itself a substring of another property.

   A common case for this is in defining multiple HTTP ports within a configuration. The following custom property could be encountered during configuration.

   ```
   port='80'
   ```

   The property is converted into the token `@port@`.

   However, if another assignment to the port is encountered during configuration, the token is formed improperly. The following custom property could be encountered.

   ```
   port='9080'
   ```

   When it is scanned, the previous token creation causes an incorrect token to be created: `90@port@`.

## Using XPath to specify property values

Use XPath expressions to specify custom property values.

Use XPath expressions to locate values that are otherwise difficult to specify with name-value pairs.

**XPath expression**

An XPath expression locates and assigns the specified value to all instances of the value in the data file that have the location specified by the XPath expression.

XPath expressions work for values that

- are not unique within the data file, for example, the values true and false might be assigned to several variables
- cannot be replaced everywhere in the data file with the same value

For information about XPath expressions, see the XML Path Language (XPath) Version 1.0 produced by the World Wide Web Consortium (WC3).

**XPath expression syntax**

The syntax of a generic XPath expression is as follows:

```
value_name.value
value_name.xpath=/XmlNode/Child[@attrName="value"]/XmlNode/@matchAttrName
```

The table provides descriptions of the XPath expression elements and are provided as a guide to create a basic XPath expression.

See the WC3 documentation on XPath language: World Wide Web Consortium (WC3).

| XPath expression element | Description |
|---|---|
| *value_name*.value | Assigns a `value_name` to the `value` that you want to locate in a data file. |
| *value_name*.xpath | Assigns the same `value_name` to the XPath expression. |
| .xpath | Query to locate that attribute whose `value` is to be replaced by `value_name` |
| /XmlNode/ | Selects an element node named XmlNode in the XML data file. |
| /XmlNode/Child | Selects an element node named Child with a parent named XmlNode. |
| /XmlNode[@attrName="value"] | Selects an element node of type XmlNode with an attribute of attrName that has a specific value. |

**Rules for XPath expressions used in promote.properties files**

- Do not select an element node with your XPath query.
- The XPath expression must point to an attribute that you want to update, not to an element node in the XML file.
- XPath queries should always end with /@attrName to target an attribute and not the node element.
- When you configure datasources, do not use absolute paths. Specify paths that are relative to container nodes.

**Example of XPath expressions in promote.properties files**

The following is an example of an XPath expression that locates and updates a value in the `jdbc.xml` data file.

The example XPath expression looks for a DataSource with the name widgetDB and updates the J2EEResourceProperty attribute to 50000 where the name attribute is portNumber.

```
dbport.value=50000
dbport.xpath=//DataSource[@name='widgetDB']/J2EEResourcePropertySet/J2EEResourceProperty
[@name='portNumber']/@value
```

The XML data file in the following example shows an excerpt from the `jdbc.xml` file with the updated port number value.

```
<jdbc>
<RAFW_JDBCDataSources>
<DataSource
  name="widgetDB"
        >
<J2EEResourcePropertySet
  WASKey="propertySet"
 >
 <J2EEResourceProperty
    WASKey="resourceProperties"
    name="portNumber"
    required="false"
   type="java.lang.Integer"
   value="50000"
   >
 </J2EEResourceProperty>
 </J2EEResourcePropertySet>
</DataSource>
</RAFW_JDBCDataSources>
</jdbc>
```

### Example of relative paths to datasources in XPath expressions

When you run the action `was_common_configure_jdbc_datasources` in promote mode, the starting point for the XPath expression must be the relative path to the container node. You must therefore omit the absolute path declarations `/jdbc/RAFW_JDBCDataSources` from the expression and start with `/DataSource`. For example,

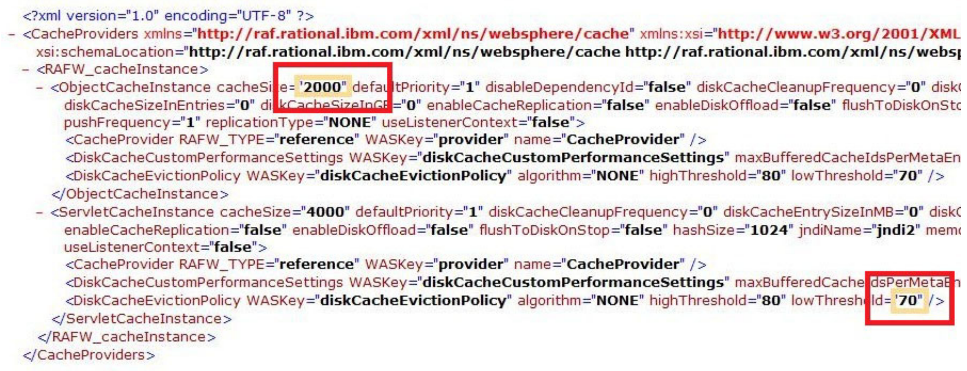
```
<DataSource
  name="widgetDB"
          >
<J2EEResourcePropertySet
  WASKey="propertySet"
 >
 <J2EEResourceProperty
    WASKey="resourceProperties"
    name="portNumber"
    required="false"
   type="java.lang.Integer"
   value="50000"
   >
 </J2EEResourceProperty>
 </J2EEResourcePropertySet>
</DataSource>
```

## Example of using an XPath expression

The following example is set in an IBM UrbanCode Deploy environment.

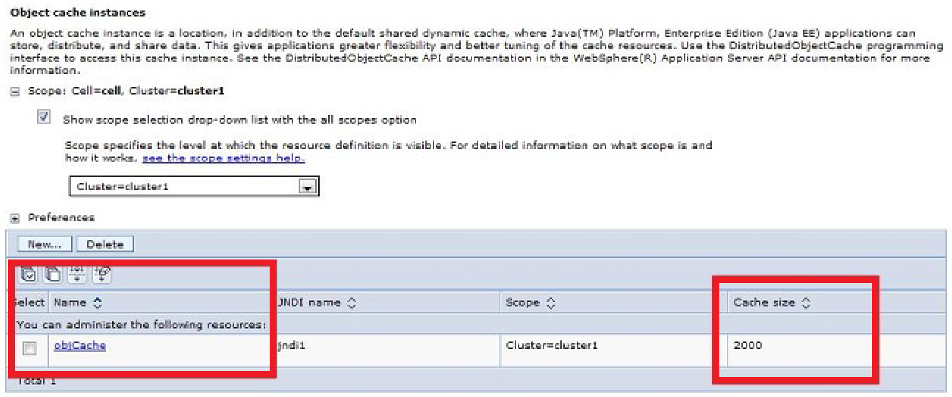
The example uses a WebSphere Application Server configuration file, `cache.xml`.

1. Original `cache.xml` file:



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <CacheProviders xmlns="http://raf.rational.ibm.com/xml/ns/websphere/cache" xmlns:xsi="http://www.w3.org/2001/XML
    xsi:schemaLocation="http://raf.rational.ibm.com/xml/ns/websphere/cache http://raf.rational.ibm.com/xml/ns/websp
  - <RAFW_cacheInstance>
    - <ObjectCacheInstance cacheSize="2000" defaultPriority="1" disableDependencyId="false" diskCacheCleanupFrequency="0" disk
        diskCacheSizeInEntries="0" diskCacheSizeInGB="0" enableCacheReplication="false" enableDiskOffload="false" flushToDiskOnSto
        pushFrequency="1" replicationType="NONE" useListenerContext="false">
        <CacheProvider RAFW_TYPE="reference" WASKey="provider" name="CacheProvider" />
        <DiskCacheCustomPerformanceSettings WASKey="diskCacheCustomPerformanceSettings" maxBufferedCacheIdsPerMetaEn
        <DiskCacheEvictionPolicy WASKey="diskCacheEvictionPolicy" algorithm="NONE" highThreshold="80" lowThreshold="70" />
      </ObjectCacheInstance>
    - <ServletCacheInstance cacheSize="4000" defaultPriority="1" diskCacheCleanupFrequency="0" diskCacheEntrySizeInMB="0" disk
        enableCacheReplication="false" enableDiskOffload="false" flushToDiskOnStop="false" hashSize="1024" jndiName="jndi2" mem
        useListenerContext="false">
        <CacheProvider RAFW_TYPE="reference" WASKey="provider" name="CacheProvider" />
        <DiskCacheCustomPerformanceSettings WASKey="diskCacheCustomPerformanceSettings" maxBufferedCacheIdsPerMetaEn
        <DiskCacheEvictionPolicy WASKey="diskCacheEvictionPolicy" algorithm="NONE" highThreshold="80" lowThreshold="70" />
      </ServletCacheInstance>
    </RAFW_cacheInstance>
  </CacheProviders>
```

2. Cache resources as seen in the WebSphere Application Server console:



**Object cache instances**

An object cache instance is a location, in addition to the default shared dynamic cache, where Java(TM) Platform, Enterprise Edition (Java EE) applications can store, distribute, and share data. This gives applications greater flexibility and better tuning of the cache resources. Use the DistributedObjectCache programming interface to access this cache instance. See the DistributedObjectCache API documentation in the WebSphere(R) Application Server API documentation for more information.

⊟ Scope: Cell=cell, Cluster=cluster1

[✓] Show scope selection drop-down list with the all scopes option

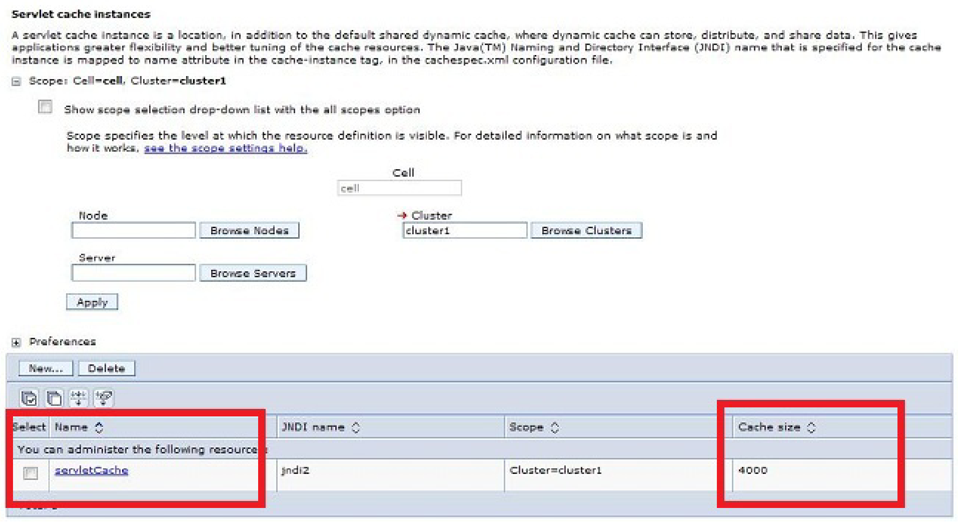
Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it works, see the scope settings help.

Cluster=cluster1

⊞ Preferences

New... Delete

| Select | Name | JNDI name | Scope | Cache size |
|---|---|---|---|---|
| You can administer the following resources: | | | | |
| ☐ | objCache | jndi1 | Cluster=cluster1 | 2000 |
| Total 1 | | | | |



**Servlet cache instances**

A servlet cache instance is a location, in addition to the default shared dynamic cache, where dynamic cache can store, distribute, and share data. This gives applications greater flexibility and better tuning of the cache resources. The Java(TM) Naming and Directory Interface (JNDI) name that is specified for the cache instance is mapped to name attribute in the cache-instance tag, in the cachespec.xml configuration file.

Scope: Cell=cell, Cluster=cluster1

Show scope selection drop-down list with the all scopes option

Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it works, see the scope settings help.

Cell: cell

Node:

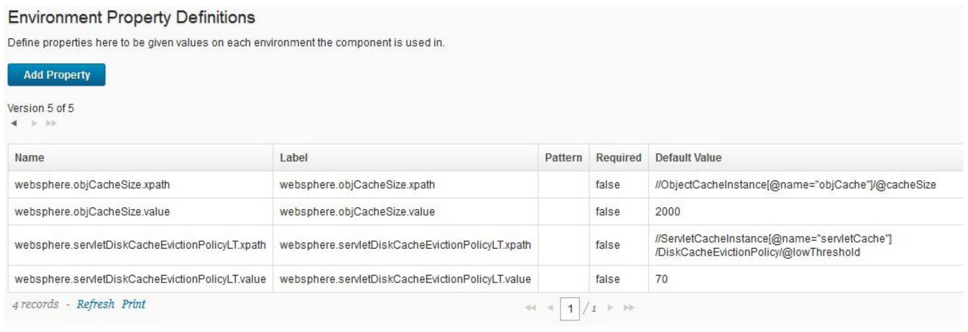
Cluster: cluster1

Server:

Preferences

| Select | Name | JNDI name | Scope | Cache size |
|---|---|---|---|---|
| You can administer the following resources: | | | | |
| ☐ | servletCache | jndi2 | Cluster=cluster1 | 4000 |

3. In IBM UrbanCode Deploy, the resources are expressed as environment property definitions on the component.



**Environment Property Definitions**

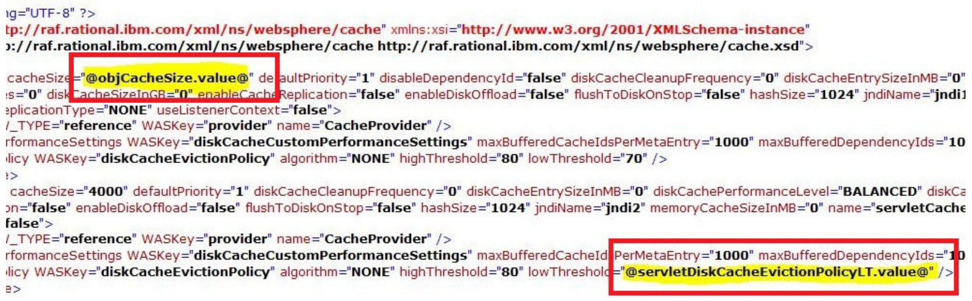
Define properties here to be given values on each environment the component is used in.

Version 5 of 5

| Name | Label | Pattern | Required | Default Value |
|---|---|---|---|---|
| websphere.objCacheSize.xpath | websphere.objCacheSize.xpath | | false | //ObjectCacheInstance[@name="objCache"]/@cacheSize |
| websphere.objCacheSize.value | websphere.objCacheSize.value | | false | 2000 |
| websphere.servletDiskCacheEvictionPolicyLT.xpath | websphere.servletDiskCacheEvictionPolicyLT.xpath | | false | //ServletCacheInstance[@name="servletCache"]/DiskCacheEvictionPolicy/@lowThreshold |
| websphere.servletDiskCacheEvictionPolicyLT.value | websphere.servletDiskCacheEvictionPolicyLT.value | | false | 70 |

4 records - Refresh Print

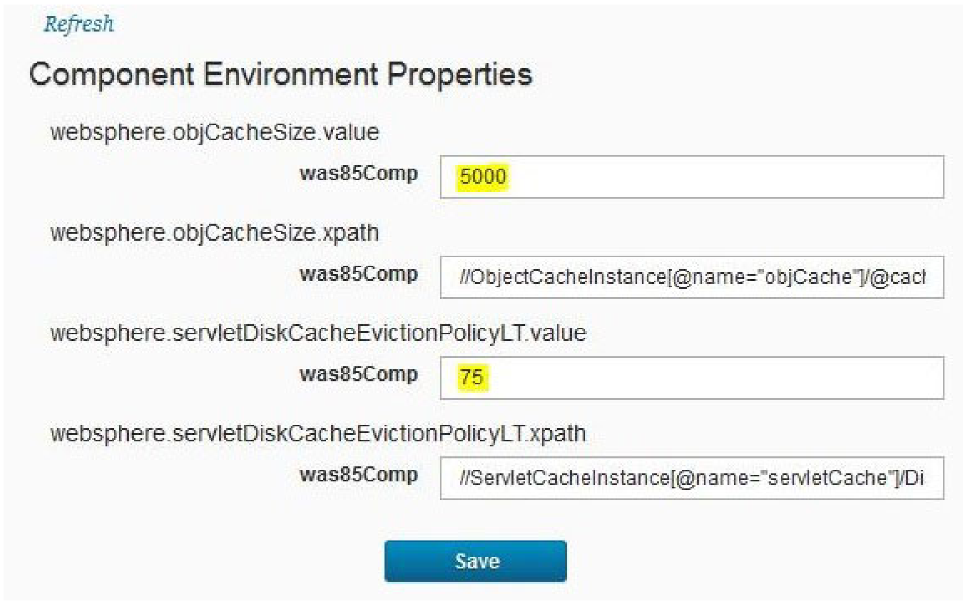
4. When you generate the template, tokens are inserted into the file to represent the locations for the cache resources.



```
ng="UTF-8" ?>
tp://raf.rational.ibm.com/xml/ns/websphere/cache" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
o://raf.rational.ibm.com/xml/ns/websphere/cache http://raf.rational.ibm.com/xml/ns/websphere/cache.xsd">

 cacheSize="@objCacheSize.value@" defaultPriority="1" disableDependencyId="false" diskCacheCleanupFrequency="0" diskCacheEntrySizeInMB="0"
s="0" diskCacheSizeInGB="0" enableCacheReplication="false" enableDiskOffload="false" flushToDiskOnStop="false" hashSize="1024" jndiName="jndi1
eplicationType="NONE" useListenerContext="false">
/_TYPE="reference" WASKey="provider" name="CacheProvider" />
rformanceSettings WASKey="diskCacheCustomPerformanceSettings" maxBufferedCacheIdsPerMetaEntry="1000" maxBufferedDependencyIds="10
licy WASKey="diskCacheEvictionPolicy" algorithm="NONE" highThreshold="80" lowThreshold="70" />
e>
 cacheSize="4000" defaultPriority="1" diskCacheCleanupFrequency="0" diskCacheEntrySizeInMB="0" diskCachePerformanceLevel="BALANCED" diskCa
on="false" enableDiskOffload="false" flushToDiskOnStop="false" hashSize="1024" jndiName="jndi2" memoryCacheSizeInMB="0" name="servletCache
false">
/_TYPE="reference" WASKey="provider" name="CacheProvider" />
rformanceSettings WASKey="diskCacheCustomPerformanceSettings" maxBufferedCacheIdsPerMetaEntry="1000" maxBufferedDependencyIds="10
licy WASKey="diskCacheEvictionPolicy" algorithm="NONE" highThreshold="80" lowThreshold="@servletDiskCacheEvictionPolicyLT.value@" />
e>
```

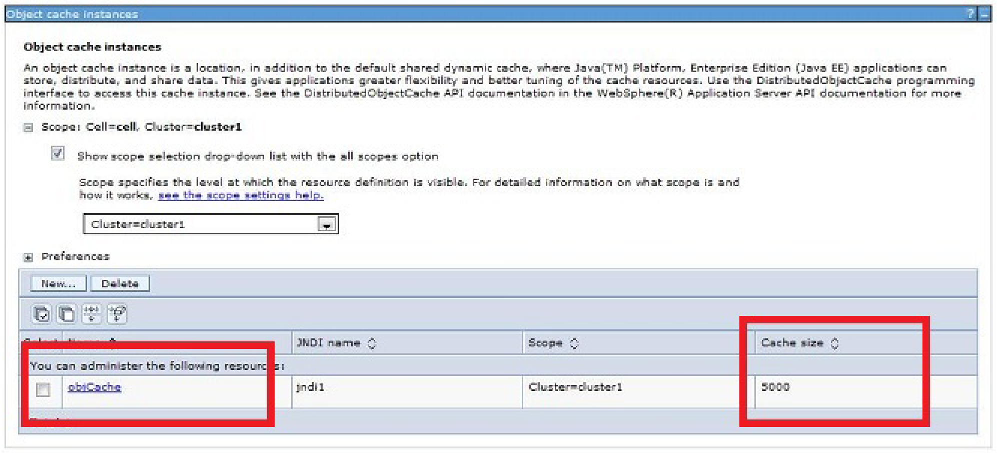
5. Run the Configure step to apply the template to a live cell. You can specify values for the properties. In the following dialog, 5000 is specified for *objCache* and 75 is specified for *servletCache*.


Refresh

Component Environment Properties

| | |
|---|---|
| websphere.objCacheSize.value | |
| was85Comp | 5000 |
| websphere.objCacheSize.xpath | |
| was85Comp | //ObjectCacheInstance[@name="objCache"]/@cacl |
| websphere.servletDiskCacheEvictionPolicyLT.value | |
| was85Comp | 75 |
| websphere.servletDiskCacheEvictionPolicyLT.xpath | |
| was85Comp | //ServletCacheInstance[@name="servletCache"]/Di |

The changes produced by running the configuration process are shown in the WebSphere Application Server console, as the following example shows for *objCache*.


Object cache instances

**Object cache instances**

An object cache instance is a location, in addition to the default shared dynamic cache, where Java(TM) Platform, Enterprise Edition (Java EE) applications can store, distribute, and share data. This gives applications greater flexibility and better tuning of the cache resources. Use the DistributedObjectCache programming interface to access this cache instance. See the DistributedObjectCache API documentation in the WebSphere(R) Application Server API documentation for more information.

Scope: Cell=**cell**, Cluster=**cluster1**

[✓] Show scope selection drop-down list with the all scopes option

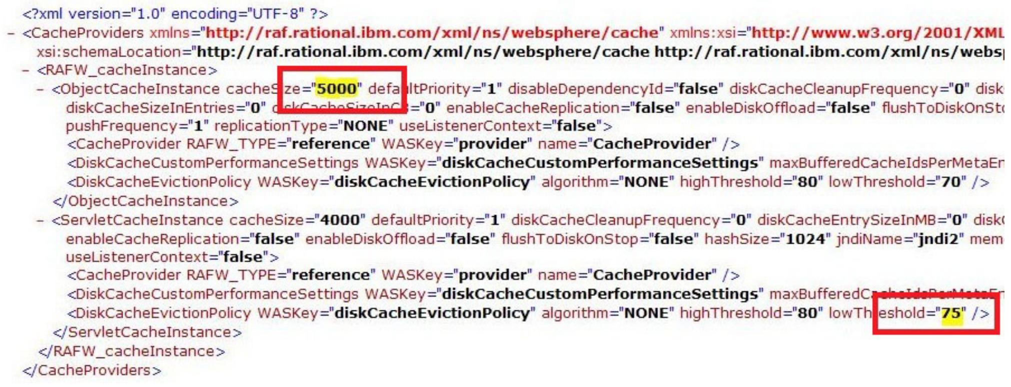
Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it works, see the scope settings help.

Cluster=cluster1

Preferences

| Select | Name | JNDI name | Scope | Cache size |
|---|---|---|---|---|
| You can administer the following resources: | | | | |
| ☐ | objCache | jndi1 | Cluster=cluster1 | 5000 |

You can also import the configuration data from the cell to see the changed contents of `cache.xml`. Note the highlighted changes.



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <CacheProviders xmlns="http://raf.rational.ibm.com/xml/ns/websphere/cache" xmlns:xsi="http://www.w3.org/2001/XML
    xsi:schemaLocation="http://raf.rational.ibm.com/xml/ns/websphere/cache http://raf.rational.ibm.com/xml/ns/websp
  - <RAFW_cacheInstance>
    - <ObjectCacheInstance cacheSize="5000" defaultPriority="1" disableDependencyId="false" diskCacheCleanupFrequency="0" disk
        diskCacheSizeInEntries="0" diskCacheSizeInGB="0" enableCacheReplication="false" enableDiskOffload="false" flushToDiskOnSto
        pushFrequency="1" replicationType="NONE" useListenerContext="false">
        <CacheProvider RAFW_TYPE="reference" WASKey="provider" name="CacheProvider" />
        <DiskCacheCustomPerformanceSettings WASKey="diskCacheCustomPerformanceSettings" maxBufferedCacheIdsPerMetaEn
        <DiskCacheEvictionPolicy WASKey="diskCacheEvictionPolicy" algorithm="NONE" highThreshold="80" lowThreshold="70" />
      </ObjectCacheInstance>
    - <ServletCacheInstance cacheSize="4000" defaultPriority="1" diskCacheCleanupFrequency="0" diskCacheEntrySizeInMB="0" disk
        enableCacheReplication="false" enableDiskOffload="false" flushToDiskOnStop="false" hashSize="1024" jndiName="jndi2" mem
        useListenerContext="false">
        <CacheProvider RAFW_TYPE="reference" WASKey="provider" name="CacheProvider" />
        <DiskCacheCustomPerformanceSettings WASKey="diskCacheCustomPerformanceSettings" maxBufferedC
        <DiskCacheEvictionPolicy WASKey="diskCacheEvictionPolicy" algorithm="NONE" highThreshold="80" lowThreshold="75" />
      </ServletCacheInstance>
    </RAFW_cacheInstance>
  </CacheProviders>
```

## Comparing WebSphere configurations

Comparing configurations allows you to see how a deployed cell needs to be adjusted to fit a norm defined in a configuration template.

### About this task

You can compare a WebSphere component (configuration template) to a live cell using the IBM Middleware Configuration for WebSphere plug-in. There are two steps to follow:

1. Create an application process to apply a configuration to a cell
2. Create an application process to run the comparison

## Creating an application process to apply a configuration to a cell

### Procedure

1. Create an application process.
2. View the application.
3. Click the **Processes** tab for the application.
4. Click **Create New Process**, then save.
5. Edit the process.
6. Drag and drop the **Install Component...** step onto the process editor.
   a. Enter a name for the step.
   b. Select the template component to use (from the template generation process).
   c. Choose the **Configure WebSphere Application Server** component process.
7. Click **Save**.
8. Save the process.

   **Note:** You can create custom processes to configure subsets of the configuration data.

## Creating an application process to run the comparison

### Procedure

1. Create an application process.
2. View the application.

3. Click the **Processes** tab for the application.
4. Click **Create New Process**, then save.
5. Edit the process.
6. Drag and drop the **Run Process for Each Version...** step onto the process editor.
   a. Enter a name for the step.
   b. Select the template component to use (from the template generation process).
   c. Choose the **Compare WebSphere Configuration (template)** component process.
7. Click **Save**.
8. Save the process.

   **Note:** You can create custom processes to compare subsets of the configuration data.

## Enabling distributed management using layered templates

Use layered templates to organize configuration objects that are managed in multiple applications and managed by multiple teams.

### Overview

A *layered template* is a full or partial set of WebSphere configuration objects, such as a data source or cluster variable. It allows you to organize configuration objects across multiple applications and multiple teams. The capability reflects the reality of WebSphere configurations having multiple owners for various parts and allows configuration to be segregated for individual applications.

The configurations are set up and managed in a distributed manner. An UrbanCode Deploy process can run a single transaction that collects the distributed configuration information, merges it into a single template, and applies it. The single transaction configures all of the objects defined. If there is a failure, the transaction is rolled back. The rollback feature prevents problems with taking the WebSphere cell down or creating mismatched data.

### Features

- Multiple objects of the same parent type can be grouped together in the same file, but the file does not necessarily have to contain a full set of configuration. While all objects can be defined in one file, sometimes it is easier to organize across multiple.
- Multiple configuration types can be grouped in a component, allowing for intuitive grouping of configuration and application objects.
- The objects are defined in a file.
- Multiple layers of the same type can be spread across multiple components. The file naming scheme allows the plug-in to find and merge all the configuration objects into a final template, which is then used to configure a scope in the WebSphere topology. For example, one application binary might need a data source definition, while another needs a completely separate data source. These object definitions could in different components. Each component can then be mapped to the application, and the processes from the plugin will aggregate and configure WebSphere with each layer in a single transaction.

- Base templates can be used to create a standard scope configuration across multiple topologies, while also using layered templates to add any specific configuration.
- All configuration files can be tokenized such that values for each application environment can be replaced independently.

### Customization

The Jython code provided by the Middleware Configuration for WebSphere can be extended to meet the needs of your organization and your processes. Hooks are provided for calling custom jython code during the import and export of WebSphere configuration (in a step). See "Customizing layered templates" on page 20

## Setting up deployments to use layered templates

To deploy WebSphere configurations and applications, you use plug-ins to read information about a live deployment into a model in Resources. You then define components, applications, environments, and processes to automate deployments that create new instantiations of the modeled WebSphere environment.

### Before you begin

The following setup and preparation is required.

- Install the Middleware Configuration for WebSphere plug-in if it is not yet installed.
- Install the Application Deployment for WebSphere plug-in if it is not yet installed.
- Determine what tokens you need. Tokens are used to represent information that is different among deployment environments. For example, you might specify a different database server or JDBC resource in each of a development, test, and production environment. For each token, you need a name and an initial value. You are prompted to enter the names and values at the end of the process. See "Using custom properties when deploying WebSphere configurations and applications" on page 9

### About this task

Setting up deployments to use layered templates that you perform the following tasks:

1. Create a tokenized object definition file
2. Create a component to use the object definition file.
3. Create an application and application environment
4. Create an application process

### 1. Create a tokenized object definition file

#### About this task

In this task you prepare the required UrbanCode Deploy resources.

#### Procedure

1. Create a tokenized object definition file using one of the three following methods.

   **Use a previously generated configuration template**

   a. View the template files in a selected version of a selected component. Go to **Components** > **<ComponentName>** > **Versions** > **<Version>**.
   b. Select the file for the object type you want to layer.
   c. Copy one or more XML dom object for the desired configuration object.
   d. Paste the objects into a new file.
   e. Save the file. Use a file name that matches the template.

**Generating a configuration template**
To generate a new configuration template, follow the instructions in "Deploying WebSphere configurations and applications" on page 3, then use the instructions above for a previously generated configuration template.

**Modify an example file**
See the basic example files provided in the downloaded plug-in.

Most of the files provided are tokenized sufficiently to use without modification. For more complicated configurations, use a previously generated configuration template or generate a new one, using the directions above.

2. Add the tokenized object definition files to a source repository. You can also add the files to a location to be used as a files system repository. See UrbanCode Deploy documentation for more information.

## 2. Create a component

### About this task

In this task you prepare the required UrbanCode Deploy resources.

### Procedure

1. Choose **Components** > **Create New Component**. Set the following properties:
   - Set the component template to **Middleware Configuration for WebSphere**.
   - Set the source configuration to the source repository or location where you copied the object definition file.
2. Add component properties.
   a. In the component, go to **Configuration** > **Environment Property Definitions**.
   b. Add properties that correspond to the tokens in the object definition file. Name the property the same as the token name.
3. Add a tag
   a. Go to **Components**.
   b. Next to the component name, hover on the row, then click the tag icon. The icon is to the right of the name. Choose the MCWAS tag. Create it if it does not exist.

## 3. Create an application and application environment

### About this task

In this task create an application and an application environment.

### Procedure

1. Create an application. Choose **Application** > **Create New Application**
2. Add the component that you created to the application.
3. Create an application environment.
   a. Go to **Components**.
   b. Next to the component name, hover on the row, then click the tag icon. The icon is to the right of the name. Choose the MCWAS tag. Create it if it does not exist.
4. Set properties for the application environment.
   a. Go to **Applicationa** > **<ApplicationName>**, click the **Configuration** subtab, then click **Environment Properties**.
   b. Under **Component Environment Properties**, provide values for each property. The values are substituted for the tokens in the object definition files during deployment.

## 3. Create an application process

### About this task

In this task you create an application process and provide property values for the steps in the process.

### Procedure

1. Create an application process. Go to **Applications** > **<ApplicationName>** > **Processes**, then click **Create New Process**.
2. Drag the **Install Multiple Components** step to the process. Provide values for the properties as follows, then click **Save**.
   - **Name**: Aggregate Layered Templates
   - **Component Tag**: MCWAS (you provided this tag on the component earlier)
   - **Component Process**: Aggregate Layered Templates
3. Drag the **Run Process for Each Version** step to the process. Provide values for the properties as follows, then click **Save**.
   - **Name**: Configure WebSphere
   - **Component**: (any WebSphere configuration layer component)
   - **Component Process**: Configure WebSphere Application Server (layered templates)
   - Data set (optional). If this value is not set, then you are prompted for the value at run time. Choose one of these values:
     - **All**: manages all configuration object types
     - **Server**: manages only server object types (JVM settings)
     - **Non-server**: all non-server object types
4. Drag the **Install Multiple Components** step to the process. Provide values for the properties as follows, then click **Save**.
   - **Name**: Synchronize Component Inventories
   - **Tag**: MCWAS
   - **Process**: Synchronize Component Inventories
5. Connect the steps. Drag lines between each step. The steps should appear in the order you dragged them in the steps above.

## Customizing layered templates

Use Jython files to provide customization of layered templates.

To customize processing in layered templates, provide a custom Jython file along with the tokenized object definition files in the source repository or file system location you use. The file is included in the artifacts in CodeStation.

Any custom Jython file must define the two methods shown below. If either is not included, the transaction fails and no data is applied.

```
def preProcess(dictionary):
   print "Running preprocess code."
   print str(dictionary)
#endDef

def postProcess(dictionary):
   print "Running postprocess code."
   print str(dictionary)
#endDef
```

During execution messages are printed to the output log for both sections.

```
CRWWA4037I Executing all custom scripts in preProcess mode
CRWWA4036I Calling custom jython file <file name>
CRWWA4039I Completed execution of custom scripts
Running preprocess code.
{'WAS_HOME': <and other variable definitions>}
...
Running postprocess code.
{'WAS_HOME': <and other variable definitions>}
CRWWA4038I Executing all custom scripts in postProcess mode
CRWWA4036I Calling custom jython file <file name>
CRWWA4039I Completed execution of custom scripts
Running preprocess code.
Saving Config...
Synchronizing Nodes...
```

The save and synchronization steps occur only after the post-processing functions are run. There may be some interleaving of log output from the MCWAS plug-in, which uses Java™ loggers, and print statements from the Jython code.

### Limitations

- Only individual Jython files can be included and run. The Jython code should not rely on external Java classes in custom JAR files. There is no way for those JAR files to be included in the run-time class path or otherwise be available at run time.

## Process steps in the IBM Middleware Configuration for WebSphere plug-in

### Aggregate Template Snippet

Aggregates individual pieces of layered configuration templates into a composite template.

There are no input properties for this step.

## Compare configuration

Compares the configuration from the mapped component to the configuration of the corresponding resource

*Table 1. Input properties for the* **`Compare configuration`** *step*

| Name | Type | Description | Required |
|---|---|---|---|
| **`Options`** | String | Not supported. | No |

## Compare server configuration

Compares the server configuration from the mapped component to the configuration of the corresponding resource

*Table 2. Input properties for the* **`Compare server configuration`** *step*

| Name | Type | Description | Required |
|---|---|---|---|
| **`Options`** | String | Not supported. | No |

## Generate configuration

Generates the configuration representation using tokenized values

## Generate Template

Generates the configuration template and creates a component version

## Manage configuration

Manages the configuration for the corresponding resource

*Table 3. Input properties for the* **`Manage configuration`** *step*

| Name | Type | Description | Required |
|---|---|---|---|
| **`Options`** | String | Not supported. | No |

## Manage Configuration Layers

Compares the configuration from the mapped component to the configuration of the corresponding resource

*Table 4. Input properties for the* **`Manage Configuration Layers`** *step*

| Name | Type | Description | Required |
|---|---|---|---|
| **`Options`** | String | Format: NAME=VALUE,[NAME=VALUE]. Supply a name value pair to be used by actions. This argument can be supplied multiple times for all of the additional parameters used by actions. | No |

## Manage server configuration

Manages the server configuration for the corresponding resource

*Table 5. Input properties for the* **`Manage server configuration`** *step*

| Name | Type | Description | Required |
|---|---|---|---|
| **`Options`** | String | Not supported. | No |

# Appendix. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

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

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamoto-shi
Kanagawa, 242-8502 Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. 2003, 2014.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

## Trademarks

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

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Intel is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Windows is a registered trademark of Microsoft Corporation in the United States, other countries, or both.

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

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

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