Finding configuration changes in delta checkpoints

Last Updated: 2022-07-25

11.7

If automatic repository checkpoints are enabled, the product creates a delta checkpoint whenever a change is made to the configuration repository. A delta checkpoint compressed `zip` file contains the before and after versions of configuration files that have changed. You can extract the contents of the compressed file and then examine the extracted files to determine what has changed in the configuration.

Before you begin

Enable the product to create delta checkpoints automatically:

1. From the administrative console, click **System administration** > **Extended Repository Service**.
2. Select **Enable automatic repository checkpoints**.
3. For **Automatic checkpoint depth**, specify the number of delta checkpoints to keep.
4. Save the changes.

About this task

You can use a delta checkpoint to undo recent changes to the product configuration.

You can also use a delta checkpoint to determine what changes were made to the configuration. This topic discusses how to interpret the contents of an extracted delta repository to determine changes in the configuration.

Procedure

1. Export a delta checkpoint.
   a. Click **System administration** > **Extended repository service** > **Repository checkpoints**.
   b. On the Repository checkpoints page, select the delta checkpoint and click **Export**.
   c. On the Export repository checkpoints page, select the compressed `zip` file name.
   d. Save the file to a specified location.
2. Extract files from the exported compressed file.
3. Examine the extracted files to determine changes in the configuration.

Example

Review the following information to see how various changes to the product configuration are shown in extracted files:

- New configuration files have the suffix `.ADDED`
- Deleted configuration files have the suffix `.DELETED`
- Changed configuration files have `before` and `after` versions
- Changes to the extended repository service configuration are in `repository.xml` files
- Adding a node results in as many as three `before` and `after` file versions
- Creating clusters and cluster members changes `cluster.xml`, `serverindex.xml`, and `server.xml` files
- Creating data sources changes `resources.xml` and `variables.xml` files
- Modifying Java virtual machine settings changes `server.xml` files
- Creating a Service Integration Bus changes SIB configuration files
- Creating SIBus destinations changes the `sib-destinations.xml` and `sib-engines.xml` files
- Creating a queue connection factory changes the `resources.xml` file
- Creating a JMS queue changes the `resources.xml` file
- Deploying an application changes `serverindex.xml` and possibly other files
- Uninstalling an application changes the `serverindex.xml` file
- Adding role to user mapping changes the `admin-authz.xml` file
- Creating a security domain changes files under `waspolicies` subdirectories
- Adding SSL configurations changes the `security.xml` file

New configuration files have the suffix `.ADDED`

When configuration files are created, the before version is a marker file with the suffix `.ADDED`, such as `server.xml.ADDED`, while the after version is the actual file that is created. New configuration files result from actions such as creating nodes, clusters, application servers, applications, or SIBus artifacts.

Deleted configuration files have the suffix `.DELETED`

When configuration files are deleted, the before version is the content of the file that was deleted, while the after version is a marker file with the suffix `.DELETED`.

and Automation
Course

WebSphere Application Server V8.5.5 Administration
Course

A tour of the hardware in IBM PureApplication System
Articles

Changed configuration files have `before` and `after` versions

When existing configuration files are changed, the before version is the original configuration, while the after version is the file after the changes are made. Changes to existing configuration files result from actions such as creating or modifying resources or changing Java virtual machine settings.
If the changed files are text or XML files, you can use a text comparison tool to compare the difference between the before and after versions. A visual text comparison tool that shows the two files in side by side comparisons is more effective to highlight the differences. If a configuration element shows only changes to the `xmi:id` attribute, you can ignore these changes because they do not modify any behavior.

You cannot use text comparison tools to compare binary files such as keystore and truststore files, application binary files, and shared libraries. For key and truststore files, use ikeyman or other key management tools to look at the contents of these files for any differences in the certificates. For application binary or shared library Java archive (JAR) files, manually compare them using JAR or zip utilities to unpack the files.

Changes to the extended repository service configuration are in `repository.xml` files

When enabling or changing the configuration of the extended repository service, the extracted delta repository shows a change to the `repository.xml` file. For example, the extracted compressed file contains:

```
before/cells/isthmusCell03/repository/repository.xml
after/cells/isthmusCell03/repository/repository.xml
```

The `after` version of the `repository.xml` file contains the updated configuration. In the following example, the `after` version has an updated value for `autoCheckpointsDepth`:

```
repositorycheckpoint:ExtendedRepositoryService xmi:version="2.0"
xmlns:xmi="https://www.omg.org/XMI"
xmlns:repositorycheckpoint="https://www.ibm.com/websphere/appserver/schemas/6.0/repositorycheckpoint.xmi"
xmi:id="ExtendedRepositoryService_1" checkpointRoot="${USER_INSTALL_ROOT}/checkpoints"
  autoCheckpointsEnabled="true" autoCheckpointsDepth="50"/>
```

Adding a node results in as many as three `before` and `after` file versions

When adding a node, you might see up to three delta checkpoints being created. The first repository change is the addNode operation itself. The before image contains mostly marker files of the form *`file_name`*`.ADDED` to show that the files did not previously exist. The after image contains the file that is added. In addition, addNode also changes the configuration for system applications, and security settings in `security.xml`. For example,

```
before/cells/isthmusCell03/nodes/isthmusNode02/node.xml.ADDED
...
before/cells/isthmusCell03/applications/ibmasyncrsp.ear/deployments/ibmasyncrsp/deployment.xml
...
before/cells/isthmusCell03/security.xml
...
after/cells/isthmusCell03/nodes/isthmusNode02/node.xml
after/cells/isthmusCell03/applications/ibmasyncrsp.ear/deployments/ibmasyncrsp/deployment.xml
after/cells/isthmusCell03/security.xml
```

The changes to `security.xml` include additions to SSL configuration and key or trust stores. The addition of new SSL configuration looks like:

```
<repertoire xmi:id="SSLConfig_1326647216593" alias="NodeDefaultSSLSettings"
  managementScope="ManagementScope_1326647216593">
   <setting xmi:id="SecureSocketLayer_1326647216593" clientAuthentication="false" securityLevel="HIGH"
     enabledCiphers="" jsseProvider="IBMJSSE2" sslProtocol="SSL_TLS" keyStore="KeyStore_1326647216593"
     trustStore="KeyStore_2" trustManager="TrustManager_1326647216593"
     keyManager="KeyManager_1326647216593"/>
</repertoire>
...
<managementScopes xmi:id="ManagementScope_1326647216593"
  scopeName="(cell):isthmusCell03:(node):isthmusNode02" scopeType="node"/>
...
```

Node level key and trust stores, and trust managers, resemble:

```
<keyStores xmi:id="KeyStore_1326647216593" name="NodeDefaultKeyStore" password="{xor}CDo9Hgw="
  provider="IBMJCE" location="${CONFIG_ROOT}/cells/isthmusCell03/nodes/isthmusNode02/key.p12"
  type="PKCS12" fileBased="true" hostList="" description="Default key store for isthmusNode02"
  usage="SSLKeys" managementScope="ManagementScope_1326647216593"/>
   <keyStores xmi:id="KeyStore_1326647216594" name="NodeDefaultTrustStore"
     password="{xor}CDo9Hgw=" provider="IBMJCE"
     location="${CONFIG_ROOT}/cells/isthmusCell03/nodes/isthmusNode02/trust.p12" type="PKCS12"
     fileBased="true" hostList="" description="Default trust store for isthmusNode02"
     usage="SSLKeys" managementScope="ManagementScope_1326647216593"/>
...
<trustManagers xmi:id="TrustManager_1326647216594" name="IbmX509" provider="IBMJSSE2"
  algorithm="IbmX509" managementScope="ManagementScope_1326647216593"/>
<trustManagers xmi:id="TrustManager_1326647216593" name="IbmPKIX" provider="IBMJSSE2"
  algorithm="IbmPKIX" trustManagerClass="" managementScope="ManagementScope_1326647216593">
  <additionalTrustManagerAttrs xmi:id="DescriptiveProperty_1326647216593"
    name="com.ibm.security.enableCRLDP" value="false" type="boolean" displayNameKey=""
    nlsRangeKey="" hoverHelpKey="" range="" inclusive="false" firstClass="false"/>
  ...
</trustManagers>
...
<keyManagers xmi:id="KeyManager_1326647216593" name="IbmX509" provider="IBMJSSE2"
  algorithm="IbmX509" keyManagerClass="" managementScope="ManagementScope_1326647216593"/>
  ...
  <sslConfigGroups xmi:id="SSLConfigGroup_1326647216593" name="isthmusNode02" direction="inbound"
```

```
    sslConfig="SSLConfig_1326647216593" managementScope="ManagementScope_1326647216593"/>
  <sslConfigGroups xmi:id="SSLConfigGroup_1326647216594" name="isthmusNode02" direction="outbound"
    sslConfig="SSLConfig_1326647216593" managementScope="ManagementScope_1326647216593"/>
  ...
  <properties xmi:id="Property_1326647216593" name="com.ibm.websphere.security.DeferTAItoSSO"
    value="com.ibm.ws.security.spnego.TrustAssociationInterceptorImpl"
    description="Trust Association Interceptors are invoked after Single Sign On user validation."
    required="false"/>
```

Some system applications are targeted to new servers on the new node. The changes might include new target mappings. For example, the changes to the ibmasyncrsp application include changes to the `isthmusCell03/applications/ibmasyncrsp.ear/deployments/ibmasyncrsp/deployment.xml` file:

```
<targetMappings xmi:id="DeploymentTargetMapping_1326647226406" enable="true"
  target="ServerTarget_1326647226406"/>
  ...
  <targetMappings xmi:id="DeploymentTargetMapping_1326647226407"
    target="ServerTarget_1326647226406"/>
  ...
  <deploymentTargets xmi:type="appdeployment:ServerTarget" xmi:id="ServerTarget_1326647226406"
    name="server1" nodeName="isthmusNode02"/>
```

If you have automatic plug-in generation enabled, the product might regenerate the plug-in file. This results in another delta checkpoint being created, resembling:

```
before/cells/plugin-cfg.xml.ADDED
after/cells/plugin-cfg.xml
```

And finally, the ports of the servers in new node are added to virtual host definitions:

```
before/cells/isthmusCell03/virtualhosts.xml
after/cells/isthmusCell03/virtualhosts.xml
```

The additions to `virtualhosts.xml` include:

```
<aliases xmi:id="HostAlias_1326647278546" hostname="*" port="9130"/>
<aliases xmi:id="HostAlias_1326647278609" hostname="*" port="9508"/>
<aliases xmi:id="HostAlias_1326647278671" hostname="*" port="5113"/>
<aliases xmi:id="HostAlias_1326647278718" hostname="*" port="5112"/>
```

Creating clusters and cluster members changes `cluster.xml`, `serverindex.xml`, and `server.xml` files

Creating a cluster causes the product to add a `cluster.xml` file to the configuration repository. Creating a cluster member causes an update to the node `serverindex.xml` file and creation of new `server.xml` and other related configuration files. For example, creating a cluster called `TestCluster` with members on two different nodes, `TestCluster1_Node1_1` and `TestCluster1_Node2_1`, results in changes to the following files:

```
before/cells/isthmusCell03/clusters/TestCluster1/cluster.xml.ADDED
before/cells/isthmusCell03/nodes/isthmusNode01/serverindex.xml
before/cells/isthmusCell03/nodes/isthmusNode02/serverindex.xml
before/cells/isthmusCell03/nodes/isthmusNode02/servers/TestCluster1_Node2_1/server.xml.ADDED
before/cells/isthmusCell03/nodes/isthmusNode01/servers/TestCluster1_Node1_1/server.xml.ADDED
...
after/cells/isthmusCell03/clusters/TestCluster1/cluster.xml
after/cells/isthmusCell03/nodes/isthmusNode01/serverindex.xml
after/cells/isthmusCell03/nodes/isthmusNode02/server
after/cells/isthmusCell03/nodes/isthmusNode02/servers/TestCluster1_Node2_1/server.xml
after/cells/isthmusCell03/nodes/isthmusNode01/servers/TestCluster1_Node1_1/server.xml
```

Creating data sources changes `resources.xml` and `variables.xml` files

Creating a data source causes the product to change `resources.xml` and `variables.xml` files; for example:

```
before/cells/isthmusCell03/clusters/TestCluster1/resources.xml
before/cells/isthmusCell03/clusters/TestCluster1/variables.xml
after/cells/isthmusCell03/clusters/TestCluster1/resources.xml
after/cells/isthmusCell03/clusters/TestCluster1/variables.xml
```

A new factory is shown in configuration files as follows:

```
<factories xmi:type="resources.jdbc:CMPConnectorFactory" xmi:id="CMPConnectorFactory_1326647771671"
  name="TestCluster1DataSource_CF" authMechanismPreference="BASIC_PASSWORD"
  connectionDefinition="ConnectionDefinition_1054132487569" cmpDatasource="DataSource_1326647771656">
    <propertySet xmi:id="J2EEResourcePropertySet_1326647771671"/>
</factories>
```

A new JDBC provider with a data source is shown in configuration files as follows:

```
<resources.jdbc:JDBCProvider xmi:id="JDBCProvider_1326647771343"
  name="DB2 Universal JDBC Driver Provider (XA)"
  description="Two-phase commit DB2 JCC provider that supports JDBC 3.0. Data sources that use
    this provider support the use of XA to perform 2-phase commit processing. Use of driver
    type 2 on the application server for z/OS is not supported for data sources created under
```

```
    this provider."
  providerType="DB2 Universal JDBC Driver Provider (XA)" isolatedClassLoader="false"
    implementationClassName="com.ibm.db2.jcc.DB2XADataSource" xa="true">
  ...
  <factories xmi:type="resources.jdbc:DataSource" xmi:id="DataSource_1326647771656"
    name="TestCluster1DataSource" jndiName="TestCluster1DataSource"
    description="DB2 Universal Driver Datasource"
    providerType="DB2 Universal JDBC Driver Provider (XA)" authMechanismPreference="BASIC_PASSWORD"
    authDataAlias="" manageCachedHandles="false" logMissingTransactionContext="true"
    xaRecoveryAuthAlias="" diagnoseConnectionUsage="false" relationalResourceAdapter="builtin_rra"
    statementCacheSize="10"
    datasourceHelperClassname="com.ibm.websphere.rsadapter.DB2UniversalDataStoreHelper">
    ...
  </factories>
</resources.jdbc:JDBCProvider>
```

You might see that some configuration elements contain changes to `xml:id` only. You can ignore these changes. For example, the following two elements have changed `xml:id` values:

```
<displayNames xmi:id="DisplayName_1326647771359" value="WS_RdbResourceAdapter"/>
<displayNames xmi:id="DisplayName_1326647771360" value="WebSphere Default Messaging Provider"/>
```

Modifying Java virtual machine settings changes `server.xml` files

The product stores changes to Java virtual machine settings in the `server.xml` file:

```
before/cells/isthmusCell03/nodes/isthmusNode01/servers/TestCluster1_Node1_1/server.xml
after/cells/isthmusCell03/nodes/isthmusNode01/servers/TestCluster1_Node1_1/server.xml
```

The following changes to Java virtual machine settings:

- Enabling verbose garbage collection
- Changing the initial heap size to 512 MB
- Changing the maximum heap size to 768 MB
- Adding a system property, MyVar=MVal

Result in an `after` version of the `server.xml`:

```
<jvmEntries xmi:id="JavaVirtualMachine_1326647543890" verboseModeClass="false"
  verboseModeGarbageCollection="true" verboseModeJNI="false" initialHeapSize="512"
  maximumHeapSize="768" runHProf="false" hprofArguments="" debugMode="false"
  debugArgs="-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=7777"
  genericJvmArguments="-DMyVar=MyVal" executableJarFileName="" disableJIT="false">
```

This new version of the `server.xml` file has the additional XML attributes `executablejarFileName` and `disableJIT`. These attributes do not introduce any behavior change because a managed application server does not need `executableJarFileName` and JIT is disabled by default.

Creating a Service Integration Bus changes SIB configuration files

Creating a bus causes the product to add new files under the `cells/`*`cell_name`*`/buses/`*`bus_name`* directory and change the bus member configurations. For example, the following file change after creating a bus named `TestBus` with bus members under the `TestCluster1` scope:

```
before/cells/isthmusCell03/nodes/isthmusNode01/servers/TestCluster1_Node1_1/sib-service.xml
before/cells/isthmusCell03/nodes/isthmusNode02/servers/TestCluster1_Node2_1/sib-service.xml
before/templates/clusters/TestCluster1/servers/V8MemberTemplate/sib-service.xml
before/cells/isthmusCell03/coregroups/DefaultCoreGroup/coregroup.xml
before/cells/isthmusCell03/buses/TestBus/sib-authorisations.xml.ADDED
before/cells/isthmusCell03/buses/TestBus/sib-bus.xml.ADDED
before/cells/isthmusCell03/buses/TestBus/sib-destinations.xml.ADDED
before/cells/isthmusCell03/clusters/TestCluster1/sib-engines.xml.ADDED
after/cells/isthmusCell03/nodes/isthmusNode02/servers/TestCluster1_Node2_1/sib-service.xml
after/cells/isthmusCell03/nodes/isthmusNode01/servers/TestCluster1_Node1_1/sib-service.xml
after/templates/clusters/TestCluster1/servers/V8MemberTemplate/sib-service.xml
after/cells/isthmusCell03/coregroups/DefaultCoreGroup/coregroup.xml
after/cells/isthmusCell03/buses/TestBus/sib-authorisations.xml
after/cells/isthmusCell03/buses/TestBus/sib-bus.xml
after/cells/isthmusCell03/buses/TestBus/sib-destinations.xml
after/cells/isthmusCell03/clusters/TestCluster1/sib-engines.xml
```

Changes to `sib-service.xml` for the existing cluster members and for the cluster level template enable the SIBService. In the following example, enabling SIBService sets the `enable` property to `true`:

```
sibservice:SIBService xmi:version="2.0" xmlns:xmi="https://www.omg.org/XMI"
  xmlns:sibservice="https://www.ibm.com/websphere/appserver/schemas/6.0/sibservice.xmi"
  xmi:id="SIBService_1" enable="true"/>
```

> **Note:** The `after` version of configure files might contain changes that remove comments from the `before` version of the files.

Additional configurations are added to `coregroup.xml` file, depending on the policies you chose. The following example shows the addition of a policy for high availability:

```
<policies xmi:type="coregroup:OneOfNPolicy" xmi:id="OneOfNPolicy_1326648336750"
  name="TestCluster1.000-TestBus-3423A696EADD6FA7Policy"
  policyFactory="com.ibm.ws.hamanager.coordinator.policy.impl.OneOfNPolicyFactory"
  isAlivePeriodSec="120" quorumEnabled="false" failback="false" preferredOnly="false">
    <MatchCriteria xmi:id="MatchCriteria_1326648336765" name="type" value="WSAF_SIB"/>
    <MatchCriteria xmi:id="MatchCriteria_1326648336781" name="WSAF_SIB_MESSAGING_ENGINE"
       value="TestCluster1.000-TestBus"/>
</policies>
```

Creating SIBus destinations changes the `sib-destinations.xml` and `sib-engines.xml` files

Creating a destination causes the product to change SIB configuration files:

```
before/cells/isthmusCell03/buses/TestBus/sib-destinations.xml
before/cells/isthmusCell03/clusters/TestCluster1/sib-engines.xml
after/cells/isthmusCell03/buses/TestBus/sib-destinations.xml
after/cells/isthmusCell03/clusters/TestCluster1/sib-engines.xml
```

The `sib-destinations.xml` file shows the addition of a SIBQueue:

```
<sibresources:SIBQueue xmi:id="SIBQueue_1326648599140" identifier="TestBusQeue1"
  uuid="0AA3CFB9BB0FFA92BE5BCB57" description="" overrideOfQOSByProducerAllowed="true"
  exceptionDestination="$DEFAULT_EXCEPTION_DESTINATION" sendAllowed="true" receiveAllowed="true">
    <localizationPointRefs xmi:id="SIBLocalizationPointRef_1326648599156" cluster="TestCluster1"
      engineUuid="3423A696EADD6FA7"/>
</sibresources:SIBQueue>
```

The `sib-engines.xml` shows the addition of a SIBQueueLocaliazationPoint:

```
<localizationPoints xmi:type="sibresources:SIBQueueLocalizationPoint"
  xmi:id="SIBQueueLocalizationPoint_1326648599156" identifier="TestBusQeue1@TestCluster1.000-TestBus"
  uuid="A55E76D18D6F4339" targetUuid="0AA3CFB9BB0FFA92BE5BCB57" highMessageThreshold="50000"/>
```

The use of targetUUID correlates with the `uuid` of the SIBQueue.

Creating a queue connection factory changes the `resources.xml` file

The product stores changes to queue connection factories in `resources.xml` files. A queue connection factory that is created at the cluster level changes the cluster level `resources.xml` file:

```
before/cells/isthmusCell03/clusters/TestCluster1/resources.xml
after/cells/isthmusCell03/clusters/TestCluster1/resources.xml
```

The addition to `resources.xml` looks like:

```
<factories xmi:type="resources.j2c:J2CConnectionFactory"
  xmi:id="J2CConnectionFactory_1326648753984" name="TestClusterQCF" jndiName="TestClusterQCF"
  description="" category="" authDataAlias="" manageCachedHandles="false"
  logMissingTransactionContext="false" xaRecoveryAuthAlias=""
  connectionDefinition="ConnectionDefinition_1326644816218">
  ...
</factories>
```

Creating a JMS queue changes the `resources.xml` file

Adding a JMS queue changes the `resources.xml` file:

```
before/cells/isthmusCell03/clusters/TestCluster1/resources.xml
after/cells/isthmusCell03/clusters/TestCluster1/resources.xml
```

Creation of a JMS queue at the cluster level changes the cluster level `resources.xml` file. The addition of the `resources.xml` file looks like:

```
<j2cAdminObjects xmi:id="J2CAdminObject_1326649181984" jndiName="jms/TestClusterQueue"
  name="TestClustereQueue" description="" adminObject="AdminObject_1326644816218">
  ...
</j2cAdminObjects>
```

Deploying an application changes `serverindex.xml` and possibly other files

Application deployment involves changes to `serverindex.xml` file of the target nodes. Changes to business-level application and composition unit configurations, even for Java EE applications, results in changes to file in the application directory under `cells/`*`cell_name`*`/applications/`*`application_name`* subdirectory. For example, deployment of the IVT application to a cluster of two nodes causes changes to the following files:

```
before/cells/isthmusCell03/nodes/isthmusNode01/serverindex.xml
before/cells/isthmusCell03/nodes/isthmusNode02/serverindex.xml
before/cells/isthmusCell03/blas/IVT Application/bver/BASE/bla.xml.ADDED
before/cells/isthmusCell03/cus/IVT Application/cver/BASE/controlOpDefs.xml.ADDED
before/cells/isthmusCell03/applications/IVT Application.ear/deployments/IVT Application/deployment.xml.ADDED
...
after/cells/isthmusCell03/nodes/isthmusNode01/serverindex.xml
after/cells/isthmusCell03/nodes/isthmusNode02/serverindex.xml
after/cells/isthmusCell03/blas/IVT Application/bver/BASE/bla.xml
after/cells/isthmusCell03/cus/IVT Application/cver/BASE/controlOpDefs.xml
```

```
after/cells/isthmusCell03/applications/IVT Application.ear/deployments/IVT Application/deployment.xml
...
```

The addition to the `serverindex.xml` on each node looks like:

```
<deployedApplications>IVT Application.ear/deployments/IVT Application</deployedApplications>
```

Uninstalling an application changes the `serverindex.xml` file

Uninstalling an application causes the product to modify the `serverindex.xml` file to remove the application and to delete application files. In the exported compressed file, the deleted files are appended with `.DELETED` suffix. For example, the files affected by uninstalling the IVT application from a cluster of two nodes are:

```
before/cells/isthmusCell03/nodes/isthmusNode01/serverindex.xml
before/cells/isthmusCell03/nodes/isthmusNode02/serverindex.xml
before/cells/isthmusCell03/blas/IVT Application/bver/BASE/bla.xml
before/cells/isthmusCell03/cus/IVT Application/cver/BASE/controlOpDefs.xml
before/cells/isthmusCell03/applications/IVT Application.ear/deployments/IVT Application/deployment.xml
...
after/cells/isthmusCell03/nodes/isthmusNode01/serverindex.xml
after/cells/isthmusCell03/nodes/isthmusNode02/serverindex.xml
after/cells/isthmusCell03/blas/IVT Application/bver/BASE/bla.xml.DELETED
after/cells/isthmusCell03/cus/IVT Application/cver/BASE/controlOpDefs.xml.DELETED
after/cells/isthmusCell03/applications/IVT Application.ear/deployments/IVT Application/deployment.xml.DELETED
...
```

Adding role to user mapping changes the `admin-authz.xml` file

Administrative authorization changes affect the `admin-authz.xml` file:

```
before/cells/isthmusCell03/admin-authz.xml after/cells/isthmusCell03/admin-authz.xml
```

As an example, when adding `user2` user to the `operator` role, the affected portion of `admin-authz.xml` in the `before` version is:

```
<authorizations xmi:id="RoleAssignmentExt_2" role="SecurityRoleExt_2"/>
```

The `after` version looks like:

```
<authorizations xmi:id="RoleAssignmentExt_2" role="SecurityRoleExt_2">
  <users xmi:id="UserExt_1326649772453" name="user2"
    accessId="user:defaultWIMFileBasedRealm/uid=user2,o=defaultWIMFileBasedRealm"/>
</authorizations>
```

Creating a security domain changes files under `waspolicies` subdirectories

Security domain related files are stored under the `waspolicies` subdirectories. Adding a security domain called, for example, `TestDomain` creates many files under the `waspolices/default/securitydomains/TestDomain` directory:

```
before/waspolicies/default/securitydomains/TestDomain/domain-security-map.xml.ADDED
before/waspolicies/default/securitydomains/TestDomain/domain-security.xml.ADDED
before/waspolicies/default/securitydomains/TestDomain/wim/config/wimconfig.xml.ADDED
...
before/waspolicies/default/securitydomains/TestDomain/domain-security-map.xml
before/waspolicies/default/securitydomains/TestDomain/domain-security.xml
before/waspolicies/default/securitydomains/TestDomain/wim/config/wimconfig.xml
```

Adding SSL configurations changes the `security.xml` file

SSL configurations are stored in `security.xml`. Thus, adding an SSL configuration changes files such as the following:

```
before/cells/isthmusCell03/security.xml
after/cells/isthmusCell03/security.xml
```

A SSLConfig addition to `security.xml` looks like:

```
<repertoire xmi:id="SSLConfig_1326650114281" alias="TestSSLConfig" type="JSSE"
  managementScope="ManagementScope_1">
    <setting xmi:id="SecureSocketLayer_1326650114296" clientAuthentication="false"
      securityLevel="HIGH" jsseProvider="IBMJSSE2" sslProtocol="SSL_TLS" keyStore="KeyStore_1"
      trustStore="KeyStore_1"/>
</repertoire>
```

What to do next

Used the identified file changes to revise the product configuration as needed.

Related concepts

→ Repository checkpoint and restore function