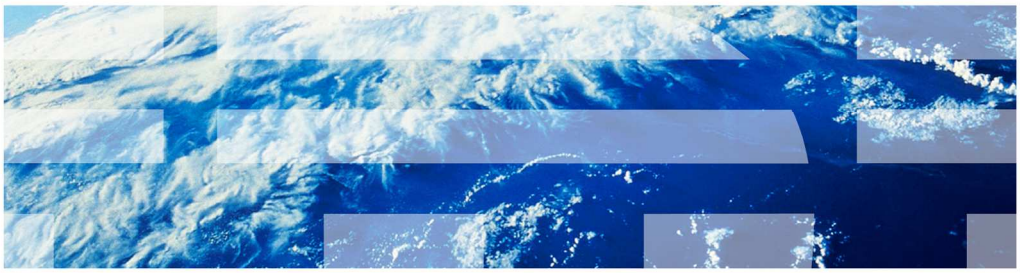
IBM WebSphere Application Server V8.5

Administrative audit for repository checkpoint

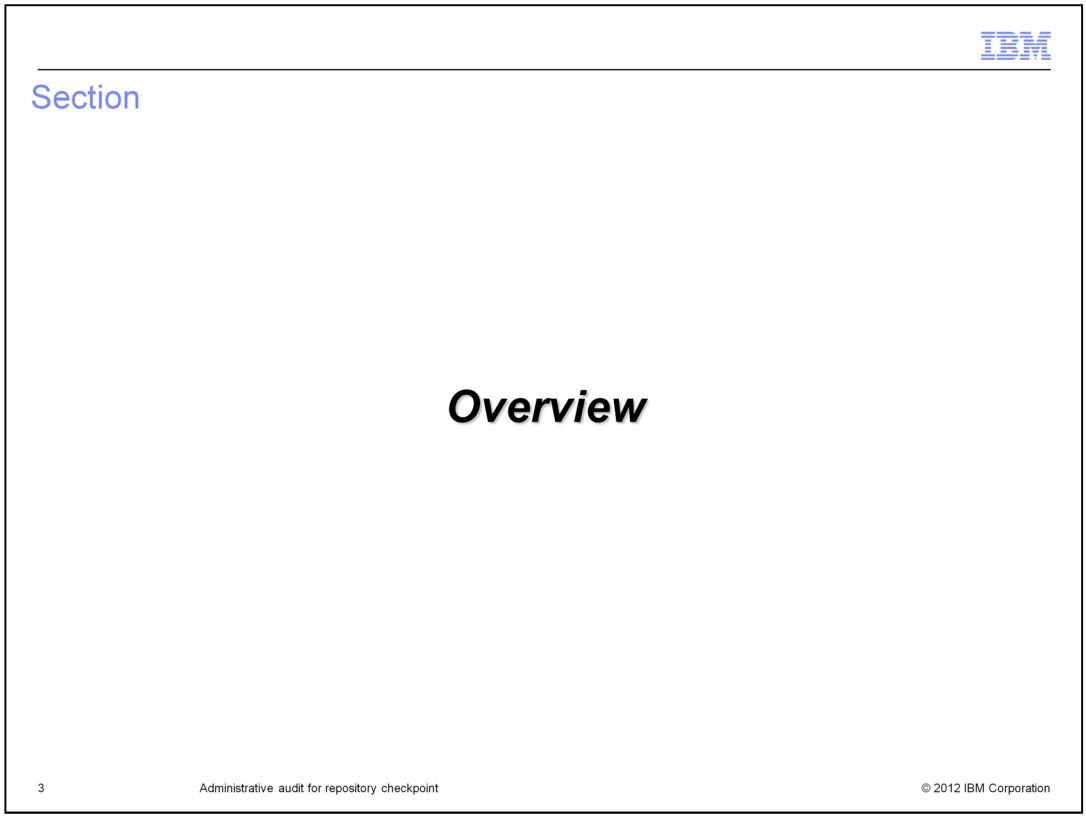


This presentation describes the extended repository service and administrative auditing capability in IBM WebSphere Application Server V8.5.

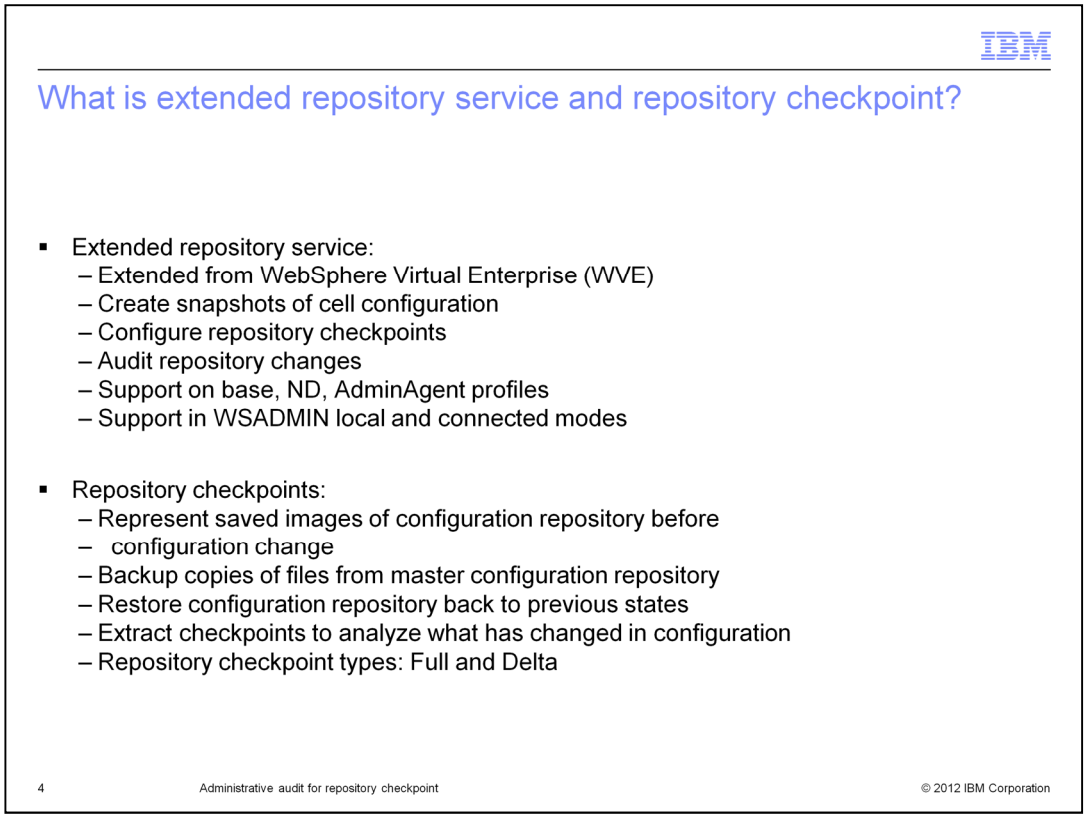
Table of contents

- Overview
- Extended repository service
- Repository checkpoints
- Repository checkpoints commands
- Admin auditing for configuration repository changes
- Summary
- References

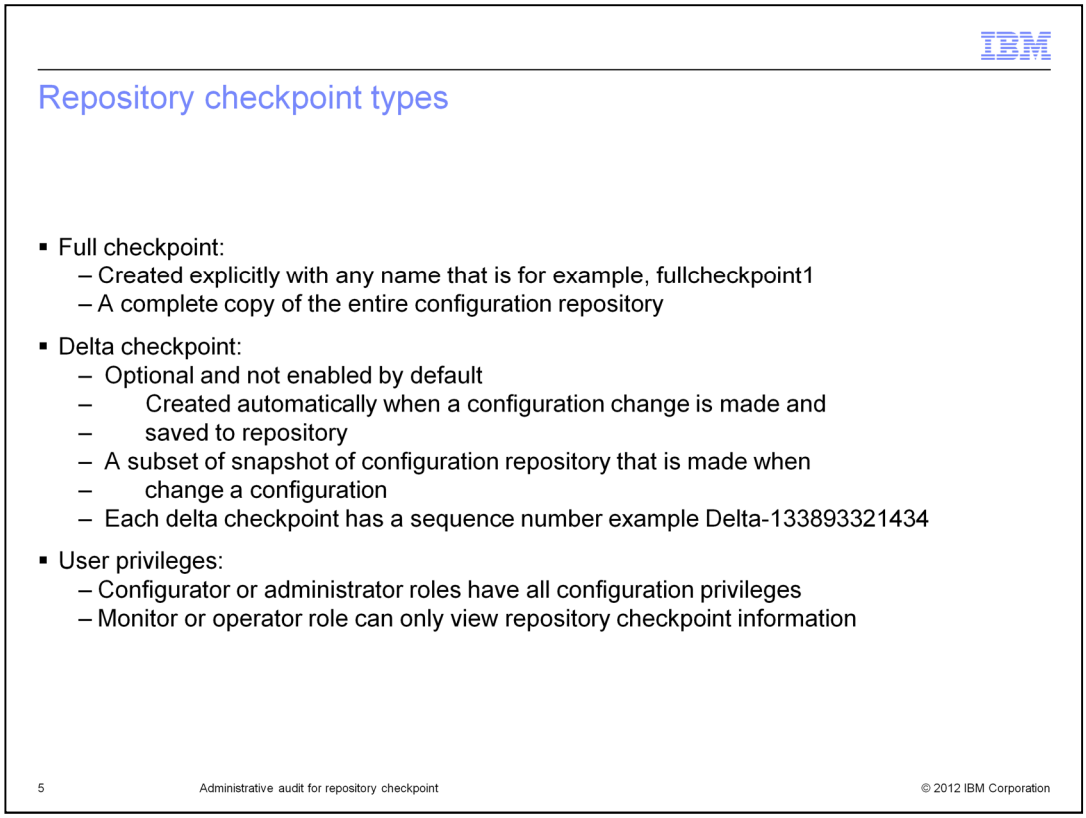
This presentation provides an overview of the extended repository service for repository checkpoints and administrative auditing capability.



This section will contain a brief overview of the extended repository service for the repository checkpoints feature.

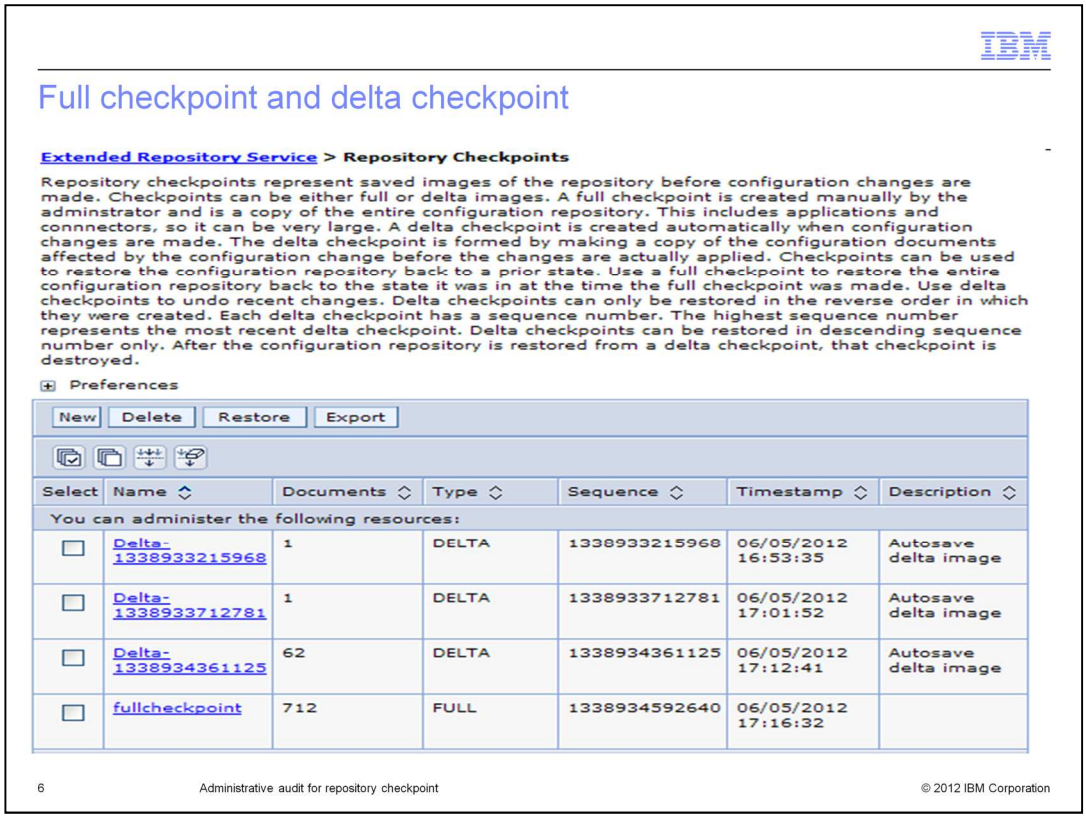


The extended repository service was originally part of WebSphere Virtual Enterprise (WVE), an add-on product to WebSphere Application Server (WAS). In WebSphere Application Server V8.5, the extended repository service was made part of the base and Network Deployment products.
The extended repository service allows you to create snapshots of cell configurations and restore them later. The snapshots are referred to as repository checkpoints. The service allows full or delta checkpoints to be created. The administrator can use the repository checkpoints to backup copies of files from the master configuration repository. Checkpoints can be used to restore the repository back to a good state when a configuration change causes a problem.



There are two types of repository checkpoints. The full checkpoint is created by an explicit user action and it contains the entire configuration repository. The delta checkpoint is created automatically when a configuration change is made and saved to the repository. The delta checkpoint is not enabled by default and it contains only a subset of the configuration files. Delta checkpoints capture a changed set of repository documents as they were just before the change was made.

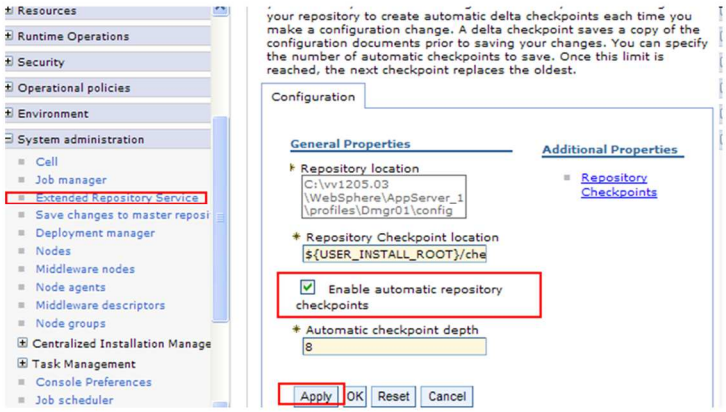
Users with configurator or administrator roles have all configuration privileges to delete, restore or extract checkpoints. Users with monitor or operator roles can only view the repository checkpoint information.



This slide shows how full delta checkpoints look in the administrative console.

Enabling extended repository service for repository checkpoint

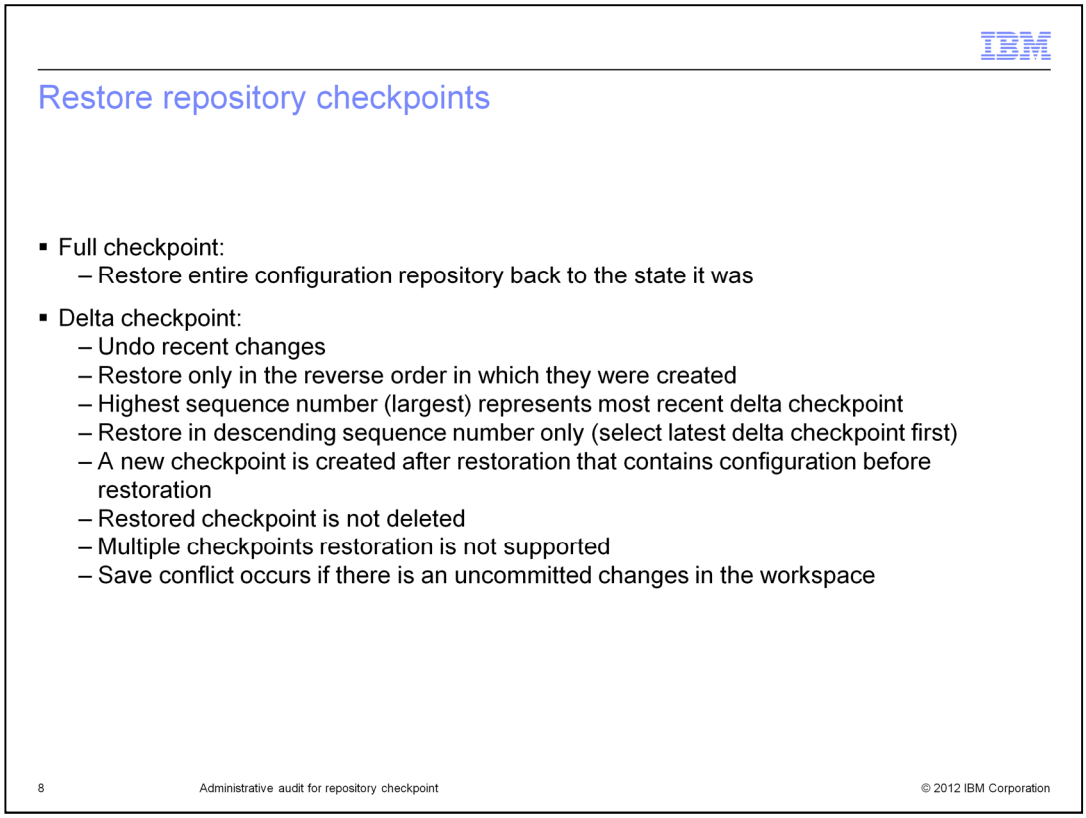
- By default, the automatic repository checkpoint is disabled in both base and network deployment editions
- Enable by way of the administrative console or WSADMIN scripting before using
- Change maximum number of checkpoints to keep (default is 5)



The automatic repository checkpoint service is disabled by default. This setting can be modified by way of the administrative console's extended repository service page.

Click System administration and then Extended repository service on the left navigation menu. On the resulting panel select or deselect "Enable automatic repository checkpoints" to enable or disable checkpoints. For automatic checkpoint depth, specify the maximum number of checkpoints to keep. After the number of checkpoints reaches this checkpoint depth, the product deletes the oldest delta checkpoint when a new delta checkpoint is created. You can use the setAutoCheckpointEnabled and setAutoCheckpointDepth commands to enable and set maximum number of checkpoints to keep. These commands will be discussed in more detail.

Modifications to the automatic repository checkpoint service configuration does not require a restart of the deployment manager or base server to become effective.

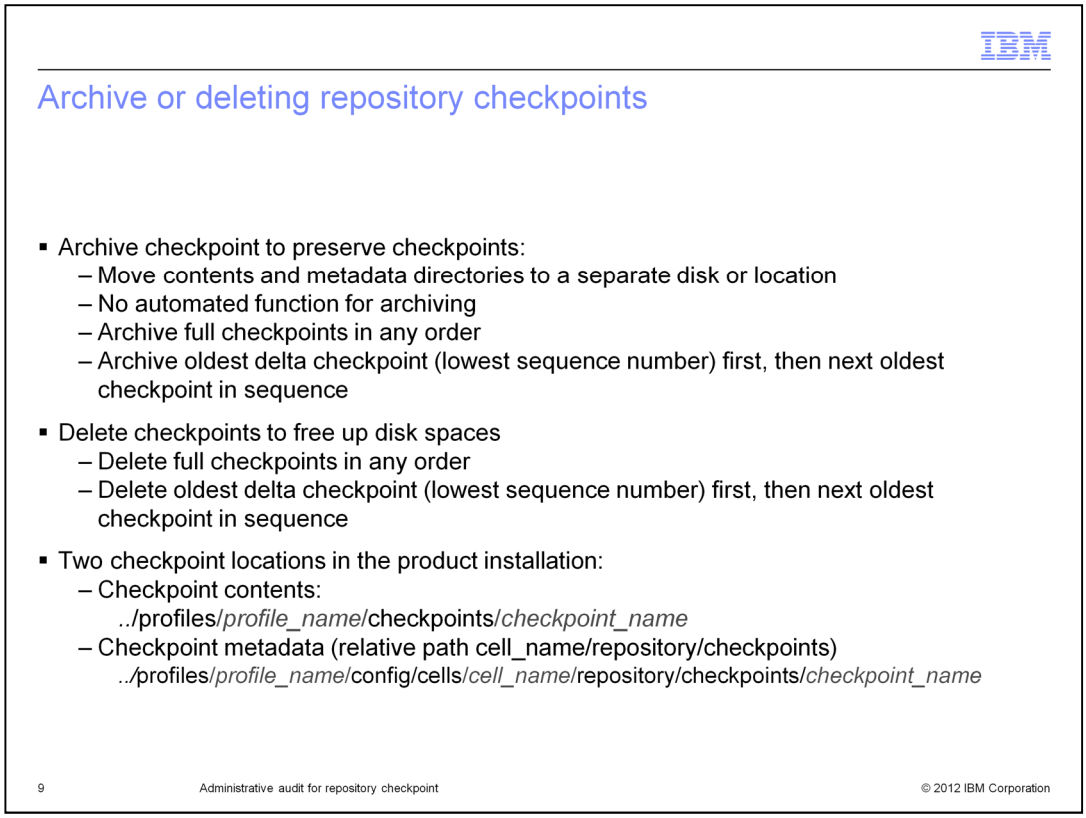


Use the full checkpoint to restore the entire configuration repository back to the state it was in at the time the full checkpoint was created.

Use delta checkpoints to undo recent changes. Restore delta checkpoints only in the reverse order in which they were created. Each delta checkpoint has a sequence number. The highest sequence number represents the most recent delta checkpoint. Therefore, restore delta checkpoints in descending order of the sequence numbers, only.

After the configuration repository is restored from a delta checkpoint, the product creates a new checkpoint that contains the configuration before restoration. The restored checkpoint is not deleted from the checkpoint directory for administrative auditing purpose. When restoring a checkpoint, a save conflict can occur if there are uncommitted changes in the workspace. Multiple checkpoint restoration is not supported.

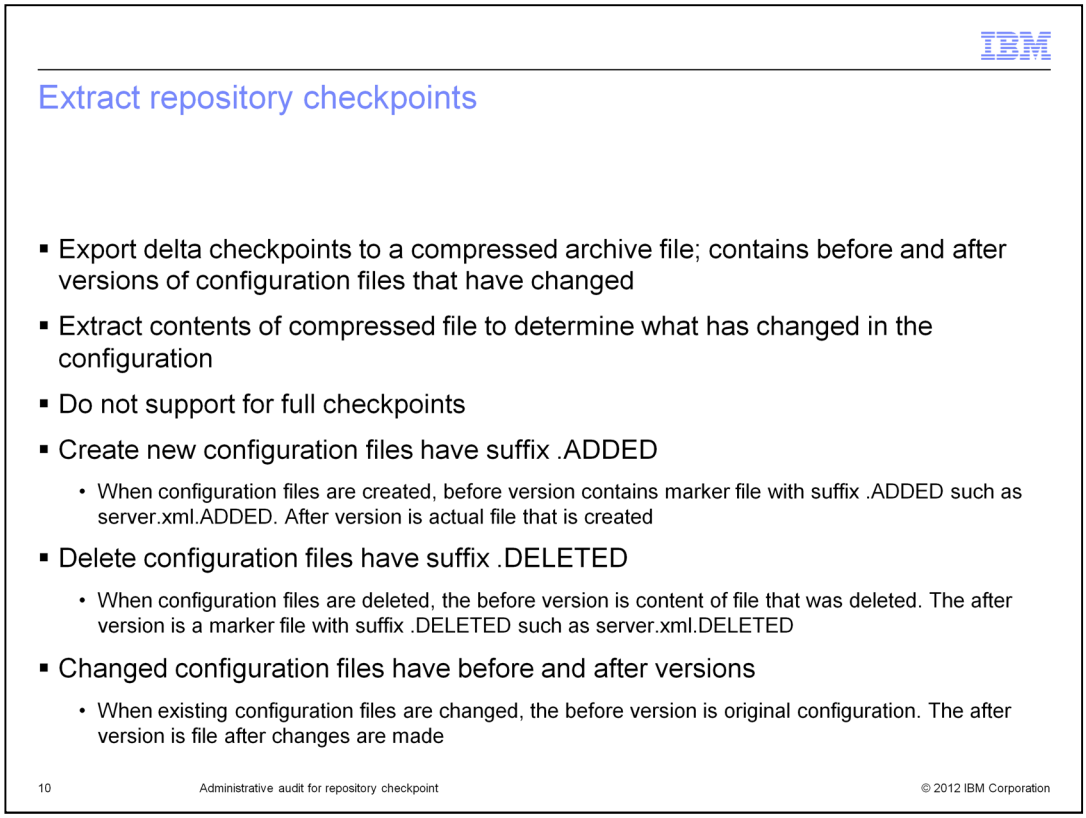
If you want to restore a delta checkpoint that is the oldest saved checkpoint, to avoid problems, you might need to increase the maximum number of delta checkpoints. After the maximum number of delta checkpoints is reached, the product deletes the oldest delta checkpoint each time a new delta checkpoint is created.



To reduce clutter and free disk space, you need to archive or delete old checkpoints periodically. The number of checkpoints that are stored to disk adds up when automatic delta checkpoints are enabled and checkpoint depth is high. The product automatically deletes delta checkpoints when the number of checkpoints reaches the checkpoint depth. If you want to preserve delta checkpoints, you must archive checkpoints before they are automatically deleted. You must delete both the checkpoint metadata and content directories to delete a checkpoint.

The product does not have an automatic archiving function. However, you can easily archive checkpoints as needed by moving checkpoint directories to a separate disk or location. There are two locations in the product installation that hold information for configuration repository checkpoints. One is located in the relative path of cells/cellName/repository/checkpoints of the profile configuration directory, it holds the checkpoint metadata.

The other location is in the profile home/checkpoints directory and it holds the checkpoint contents. These locations contain subdirectories, one for each checkpoint. Subdirectories for full checkpoints have the user-specified checkpoint name. Delta checkpoint subdirectories are named Delta-*sequence number*, where the sequence numbers reflect the time of creation. Older delta checkpoints have smaller sequence numbers and newer delta checkpoints have larger sequence numbers. To store a checkpoint for later restoration you need to archive both the checkpoint metadata and content directories.



The delta checkpoints can be exported to a compressed archive file. The compressed archive file contains the before and after versions of configuration files that have changed. To view the files, you can extract the contents of the compressed file and then examine the extracted files to determine what has changed in the configuration.

When configuration files are created, the before version contains 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 systems integration bus.

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

When existing configuration files are changed such as changing Java virtual machine settings, the before version is the original configuration, while the after version is the file after the changes are made. 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.

If you are using a stand-alone application server then the only possible target is the server itself. A monitored directory is created if the service is enabled. For example if the profile is called AppSrv01, and the server is named server1, the path is app_server_root/profiles/AppSrv01/monitoredDeployableApps/server1.

If you are using a network deployment system it is necessary to create monitored directories. For application servers on a node federated with a deployment manager, you will need to create the monitored directories for servers under the deployment manager profile. Starting at the Application Server root directory the path will be ./profiles/dmgrprofilename/monitoredDeployableApps/servers/server_name.

If multiple servers on different nodes share the same name and you want only one of the servers to be a target you can specify the node and server in the path to the monitored directory. Create a directory for the node using the node name, then ‘servers’ and finally the server_name directory. Example, starting at the application server root directory, ./profiles/dmgrprofilename/monitoredDeployableApps/nodes/node_name/servers/server_name.

For clusters, create a monitored directory under the deployment manager profile with the name of the targeted cluster. Example, starting at the application server root directory the path is: /profiles/dmgrprofilename/monitoredDeployableApps/clusters/cluster_name.

Extract repository checkpoints – Example 1

- **Creating cluster and cluster members:**
  - Creating a cluster causes product to add a cluster.xml file to configuration repository
  - Creating a cluster member causes an update to node serverindex.xml file and creation of new server.xml and other related configuration files
  - For example, creating a cluster called cluster1 with two members on node01 and node02, the exported compressed archive file contains:

```
before/cells/cell01/clusters/cluster1/cluster.xml.ADDED
before/cells/cell01/nodes/node01/serverindex.xml
before/cells/cell01/nodes/node02/serverindex.xml
before/cells/cell01/nodes/node02/servers/cluster1_node2_1/server.xml.ADDED
before/cells/cell01/nodes/node01/servers/cluster1_node1_1/server.xml.ADDED
  ...
after/cells/cell01/clusters/TestCluster1/cluster.xml
  after/cells/cell01/nodes/node01/serverindex.xml
  after/cells/cell01/nodes/node02/server
after/cells/cell01/nodes/node02/servers/cluster1_node2_1/server.xml
after/cells/cell01/nodes/ode01/servers/cluster1_node1_1/server.xml
```


This slide provides an example of how to create a new cluster and two cluster members on each node. 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 the creation of a new server.xml and other related configuration files.

Extract checkpoints – Example 2

- **Uninstalling an application**
  - Change serverindex.xml file and delete application files
  - In the exported compressed file, deleted files are appended with .DELETED suffix
  - For example, files affected by uninstalling the IVT application from a cluster of two nodes are:

```
    before/cells/cell01/nodes/node01/serverindex.xml
    before/cells/cell01/nodes/node02/serverindex.xml
    before/cells/cell01/blas/IVT Application/bver/BASE/bla.xml
    before/cells/cell01/cus/IVT Application/cver/BASE/controlOpDefs.xml
before/cells/cell01/applications/IVT Application.ear/deployments/IVT Application/deployment.xml
    ...
   after/cells/cell01/nodes/node01/serverindex.xml
   after/cells/cell01/nodes/node02/serverindex.xml
   after/cells/cell01/blas/IVT Application/bver/BASE/bla.xml.DELETED
   after/cells/cell01/cus/IVT Application/cver/BASE/controlOpDefs.xml.DELETED
after/cells/cell01/applications/IVT Application.ear/deployments/IVT
Application/deployment.xml.DELETED ...
```


This slide provides an example of how to uninstall an application. Uninstalling an application will cause the product to modify the serverindex.xml file and delete application configuration files in order to remove the application. In the exported compressed file, the deleted files are appended with .DELETED suffix.

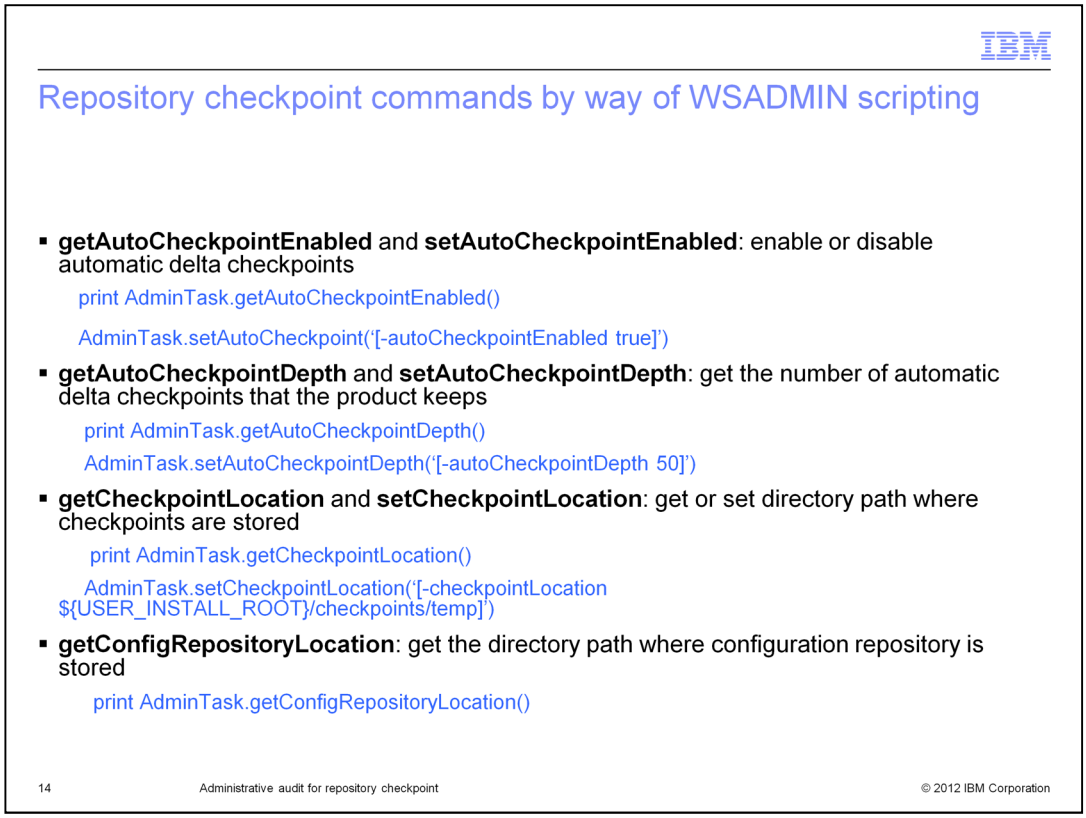
Extract checkpoints – Example 3

- **Changes to extended repository service configuration**
  - When enabling or changing configuration of extended repository service, extracted delta repository shows a change to repository.xml file
  - After version contains updated configuration:

    before/cells/isthmusCell03/repository/repository.xml
    after/cells/isthmusCell03/repository/repository.xml
  - Changes autoCheckpointDepth to “50”, after version contains the updated value in respository.xml

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

When enabling or changing the configuration of the extended repository service, the extracted delta repository shows the change to the repository.xml file.



You can use either Jython or Jacl scripting languages to create, restore, delete, and administer checkpoints by way of the WSADMIN tool. The commands in the RepositoryCheckpointCommands group support the repository checkpoint functions in WSADMIN local and connected modes. The save command, AdminConfig.save, is not needed with checkpoint commands. The product will save the changes automatically.

Repository checkpoint commands

- **createFullCheckpoint:** create a full checkpoint

  AdminTask.createFullCheckpoint('[-checkpointName full1 –checkpointDesc "a test"]')
- **extractRepositoryCheckpoint**: extract delta checkpoint to a compressed file

  AdminTask.extractRepositoryCheckpoint('[-checkpointName Delta-1338934361125 –extractToFile /temp/test1.zip]')
- **restoreCheckpoint**: restore configuration repository back to previous state (restore in reverse order)

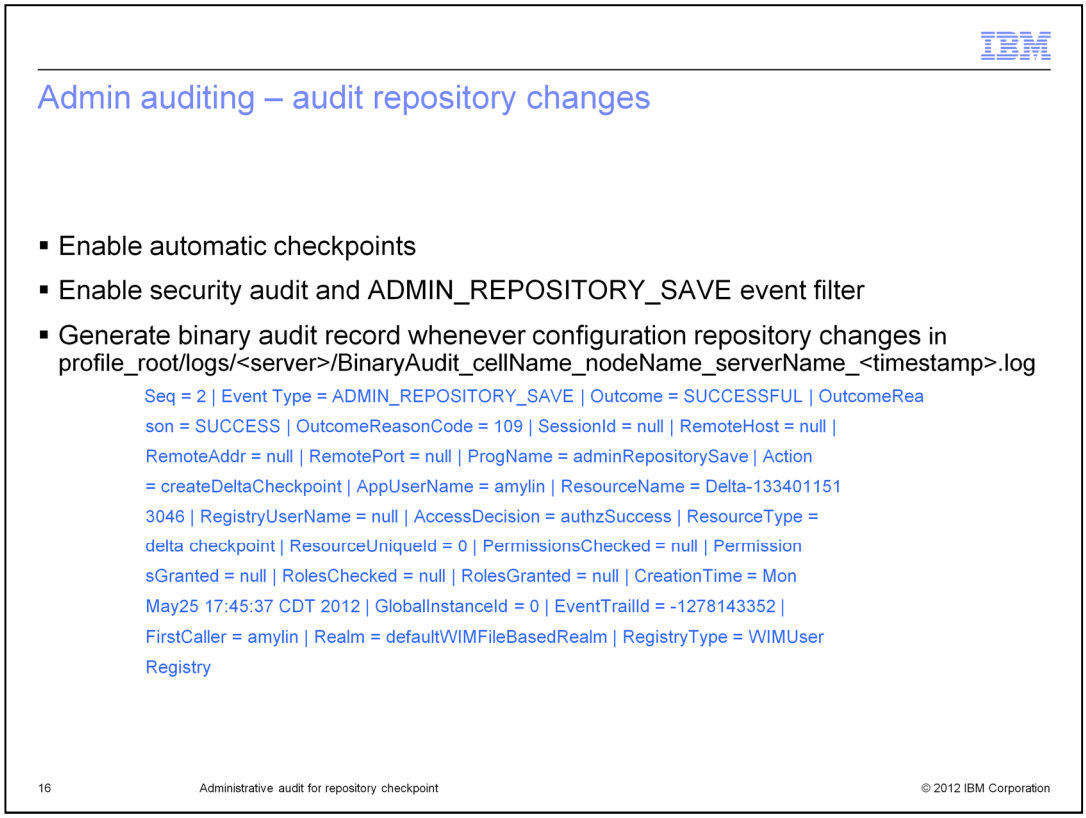
  AdminTask.restoreCheckpoint('[-checkpointName Delta-1338934361125]')
- **deleteCheckpoint**: delete full or oldest delta checkpoint

  AdminTask.deleteCheckpoint('[-checkpointName Delta-1338934361125]')
- **listCheckpoints**: get a list of existing checkpoints

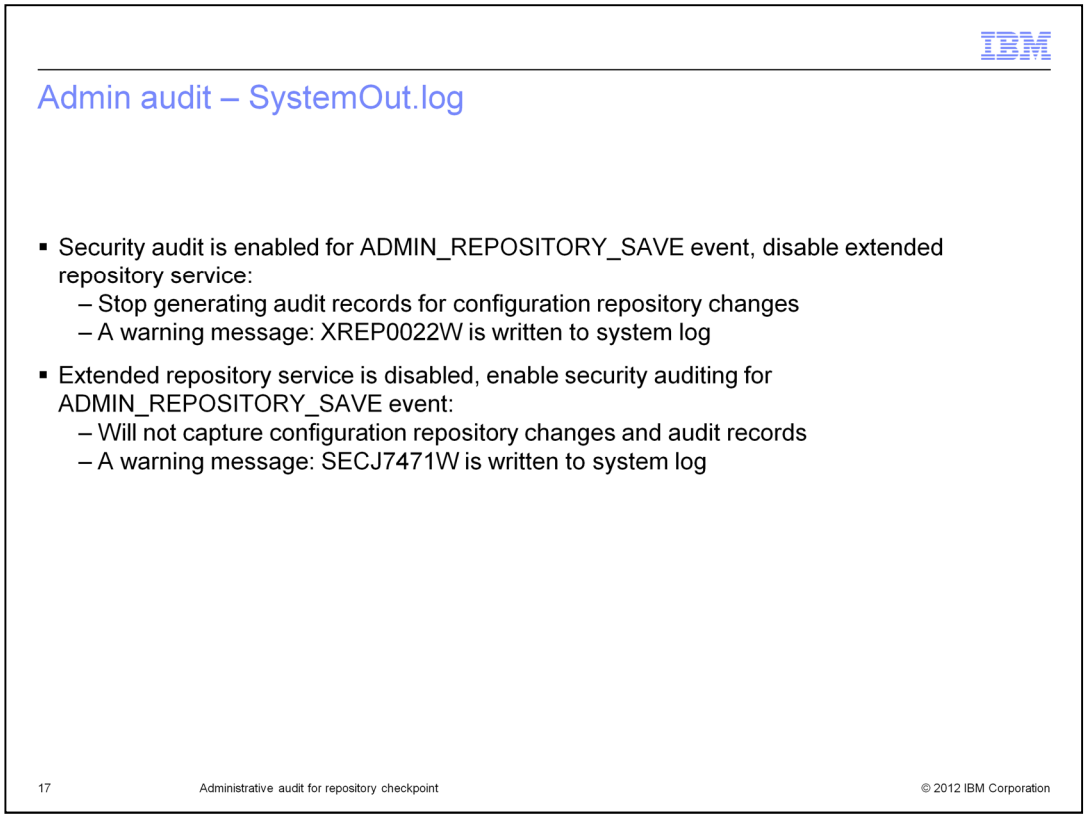
  print AdminTask.listCheckpoints()
- **listCheckpointDocuments**: get a list of documents in a checkpoint repository

  print AdminTask.listCheckpointDocuments('[-checkpointName Delta-1338934361125]')

This slide shows additional commands of the RepositoryCheckpointCommands group.

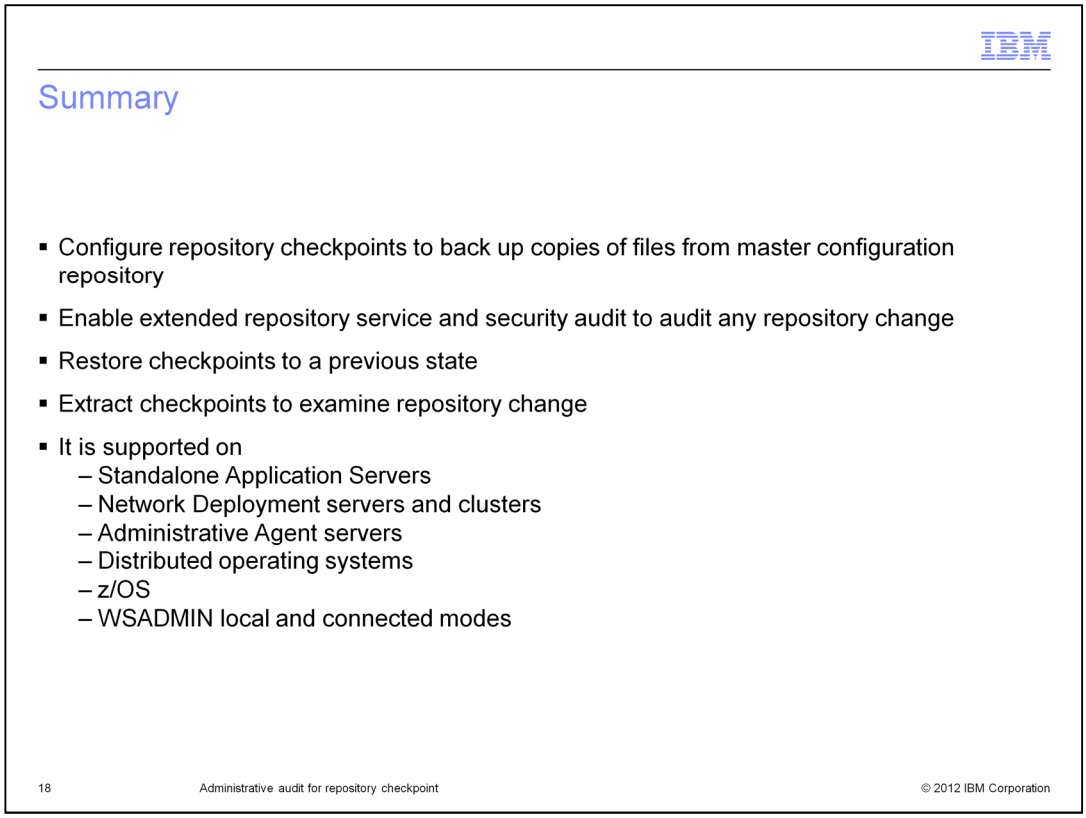


A new security audit event filter, ADMIN_REPOSITORY_SAVE, is available to the security audit function. It supports using delta checkpoints to create an audit trail. When repository checkpoint is enabled, an audit record is created each time a delta checkpoint is created. The name of the checkpoint is recorded along with user ID of the user triggering the configuration change. The Event Type = ADMIN_REPOSITORY_SAVE and Outcome = SUCCESSFUL indicate a successful configuration change in an audit record. ResourceName = Delta-xxxx indicates the name of the checkpoint. Information on how to enable automatic checkpoints, security audit and the ADMIN_REPOSITORY_SAVE event filter is available in the Information Center.



If security auditing is enabled and an audit event filter is created for ADMIN_REPOSITORY_SAVE event in the audit.log, disabling automatic checkpoint causes the product to stop generating audit records for the configuration repository changes in the log file (BinaryAudit_*xxx*.log). Warning message XREP0022W about this situation is written to the system log.

If automatic checkpoint is disabled, enabling the security auditing filter for the ADMIN_REPOSITORY_SAVE event does not capture the changes to the configuration repository and corresponding audit records. A warning message SECJ7471W about this situation is written to the system log.



In summary, the extended repository service allows you to create snapshots of cell configurations and restore them later. You can use the repository checkpoints to backup copies of files from the master configuration repository. The checkpoints can be used to restore the repository back to a previous state when configuration changes cause problems. Enabling extended repository service and security audit enables you to audit any repository change event. Delta checkpoints can be exported to a compressed archive file. The compressed archive file contains the before and after versions of configuration files that have changed. You can extract the contents of the compressed file and examine the extracted files to determine what has changed in the configuration.

References

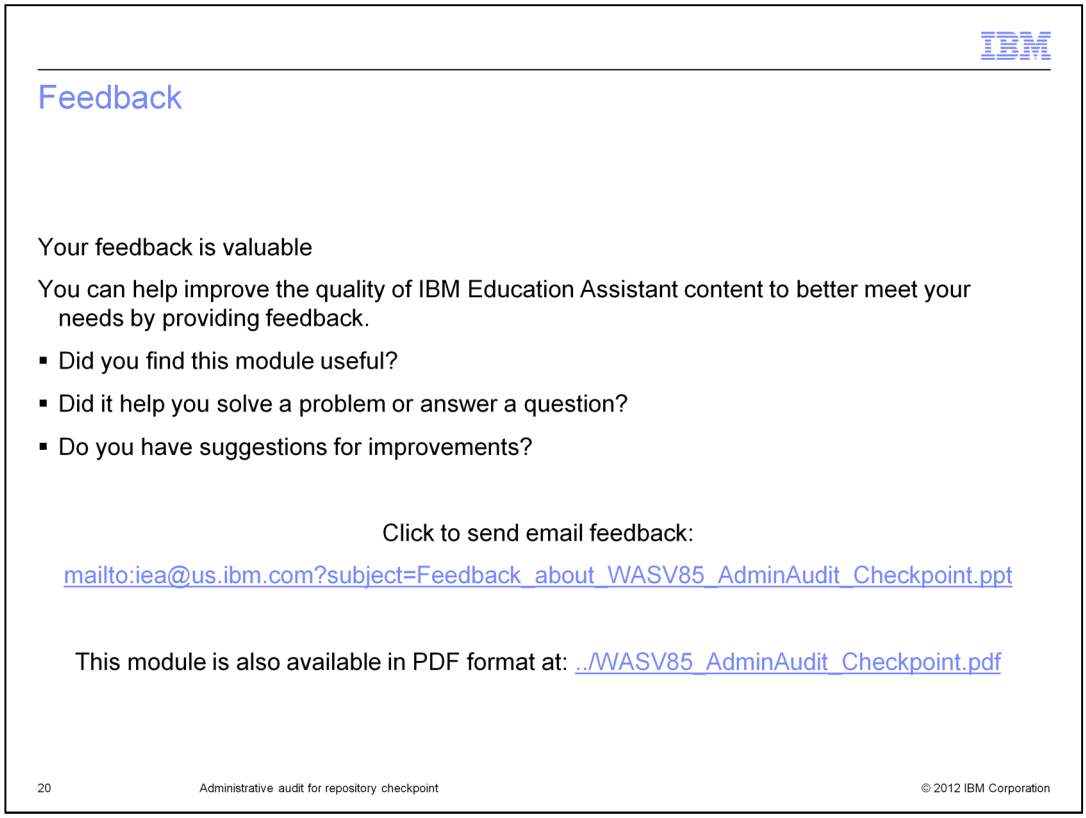
- Repository Checkpoint and restore function

http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/topic/com.ibm.websphere.nd.multiplatform.doc/ae/cwve_xdsochckpt.html

- RepositoryCheckpointCommands command group for the AdminTask object using WSADMIN scripting

http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/topic/com.ibm.websphere.nd.multiplatform.doc/ae/rxml_checkpoint_repository.html

See the WebSphere Application Server version 8.5 Information Center for additional information.



You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, disclaimer, and copyright information