Haibing Qiao
Software Engineer
March 2013

Product Implementation Training (PIT)

IBM Filenet Content Manager 5.2.0 RecoveryBin

Course Overview

- An introduction to the new RecoveryBin feature in Content Engine 5.2.0
- This feature enables an object in the Content Engine to be marked for deletion and put into RecoveryBin, it can then either be completely purged from the system, or be recovered back into its original state
- Course Prerequisites: General knowledge about Content Engine

Course Objectives

After this course you will:

- Understand various concepts related to the RecoveryBin feature
- Know the RecoveryBin use case and scope of the current release
- Understand the security and event handling related to this new feature
- Know the APIs related to the RecoveryBin

IBM®

Course Roadmap

- Terms
- Use cases
- Requirement and scope
- Demo with ACCE
- Concept, design and behavior
- Events
- Security
- API Overview

Terms

- Mark for deletion (soft delete): a CE object appears to be deleted, but still exists in the system and can be recovered.
- Hard delete(normal delete): the object is completely removed from the system. All Content Engine deletes were hard deletes before this feature. Hard delete behavior is the same as before
- Purge: completely remove a soft deleted object from system. An object can not be recovered after it has been purged
- Cascadee: any object which would be cascade deleted in a normal hard delete. For example, Annotations belonging to a Document
- RecoveryItem: the new CE class that represents the soft deleted object.
- RecoveryBin: the container of RecoveryItem. There could be multiple RecoveryBins in the system. Each RecoveryItem has to belong to one and only one RecoveryBin.

The Use Case

To prevent permanent information loss due to object removal, sometimes we need delete to be “recoverable”

- A CE object can be marked for deletion (soft delete) and put into a RecoveryBin
- Limited information of soft deleted object can be obtained by browsing the RecoveryBin
- The object can later be either purged from the system, or recovered to its original state.
- This is conceptually similar to “Trash Bin” in computer file systems

Requirements and scope

- Provide a set of new CE operations of mark for delete, recover and purge
- Fire corresponding events for new CE operations
- There shall be a special permission to allow those hidden soft deleted object to be searched and discovered. A typical use case is legal discovery.
- Object referential integrity shall be preserved
- If an object delete will cascade to subordinate objects, soft delete will also cascade in the same manner.
- If there are constraints like prevent delete or insufficient delete permission for any cascadees, soft delete will fail

Requirements and scope (continued)

- In the 5.2.0 release, the Mark For Delete operation is supported only on the VersionSeries and CustomObject as the top level objects.
- Cascade soft delete to other type of subordinate objects (OVP), such as RCR, Task is allowed.
- The following object types may not be soft deleted:
  - Folder
  - CompoundDocument
  - FederatedDocument

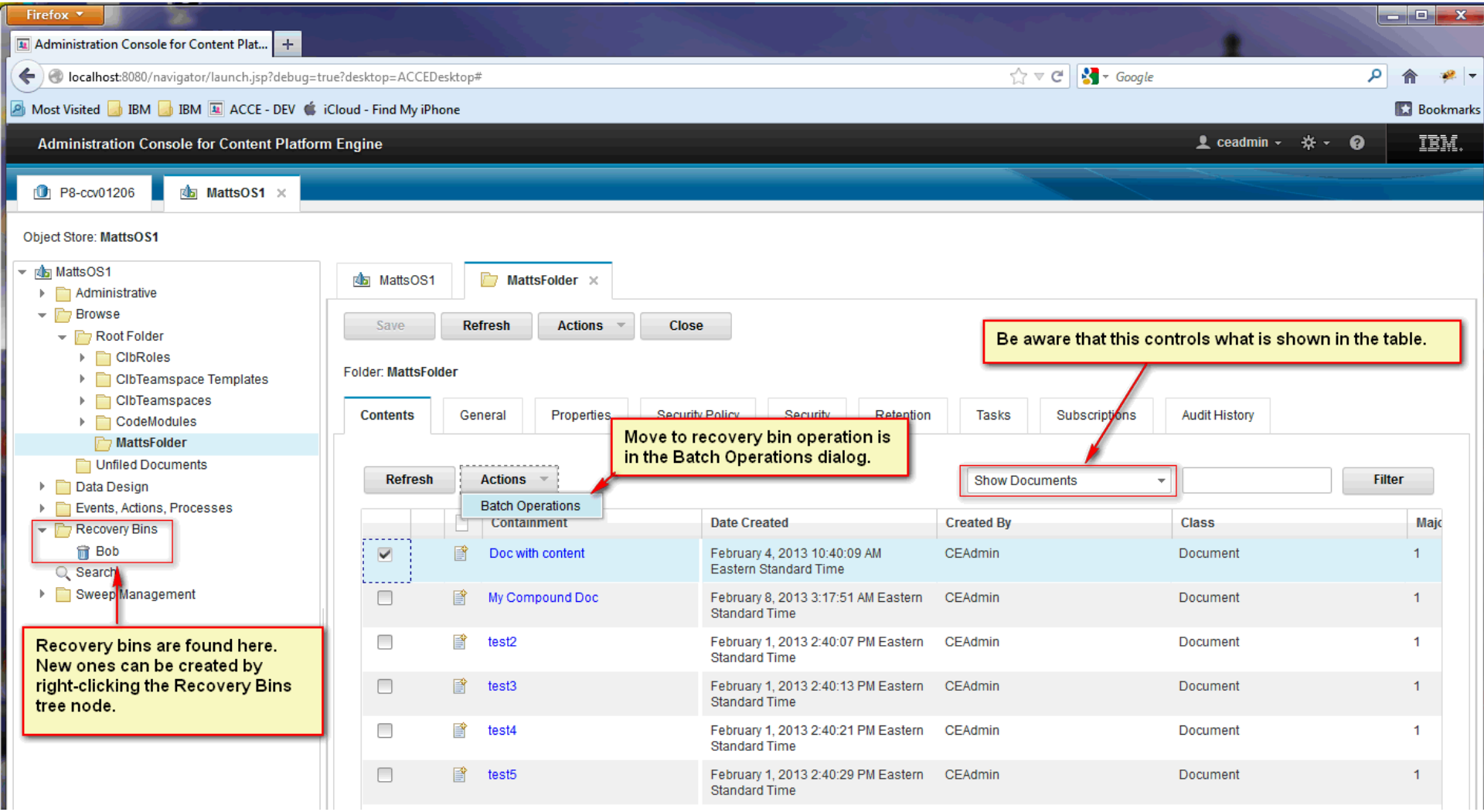
RecoveryItem class

- RecoveryItem class holds the information of a soft deleted object, including cascadees. It is created when an object is marked for deletion. It belongs to one and only one RecoveryBin. It has following properties
  - RecoveryBin: the RecoveryBin this recoveryItem belongs to. It is also a partial proxy.
  - OriginalId: the ID of the root object marked for deletion
  - OriginalClassId: the class ID of the root object marked for deletion
  - OriginalName: the value of the name property of the root object.
  - OriginalCreator: the creator of the root object
  - OriginalLastModifier: the last modifier of the root object
  - OriginalDateLastModified: the date last modified
  - OriginalObject: original root object marked for deletion
  - RecoverableObjectsCount: number of objects deleted include cascadees

RecoveryBin class

- RecoveryBin “contains” zero or many RecoveryItem objects. It is also by default the partial security proxy to the RecoveryItem objects. It has following properties:
  - DisplayName: the name of the RecoveryBin.
  - DescriptiveText string describes this recovery bin.
  - RecoveryItems An enumeration of the RecoveryItems contained with this bin. This property is marked with DeletionAction.PREVENT

Demo using ACCE



Mark for deletion process

- Mark for deletion (soft delete). When an object is marked for deletion, the object is not deleted from the database.
  - The internal flag (*isMarkedForDeletion*) will be set to indicate this object is marked for deletion,
  - The mark for deletion will propagate to all sub objects if those objects are designated to be cascade deleted
  - A MarkForDeletionEvent will be fired on each cascaded object
  - After all mark for deletion are cascaded to all sub objects, an RecoveryItem object is created, with OriginalXXX properties to be set to reflect those of the object marked for deletion
  - For version series, the OriginalXXX properties are taken from the version same as the dynamic RCR

Mark for deletion cascade rules

- The cascade of soft delete will stop at following object and its subordinates, silently:
  - Cascadee is in another object store
  - Cascadee is a document version that has more than one version in the version series
  - Cascadee is an object has already been marked for deletion, the rule is “first writer wins”.
- The cascade of soft delete will stop at these unsupported objects and throw exception:
  - Folder
  - CompoundDocument
  - FederatedDocument

IBM

Events in the soft delete

- A new Event of “MarkForDeletionEvent” will be fired whenever an object is marked for deletion. No other event such as UpdateEvent will be fired in this case
- A new event of “RecoveryEvent” will be fired whenever an object is recovered from the markForDelete state
- “DeletionEvent” will be fired for hard delete and purge. In the case of purge, the object will have “*isMarkedForDeletion*” Boolean property to be true, so event handler can differentiate the purge from the hard delete

Recovery process

- Recovery. Recover an object that has been marked for deletion to its original state
  - The internal flag (*isMarkedForDeletion*) will be set to false, same for all those cascadees that were marked in the soft delete process
  - The corresponding RecoveryItem is removed upon successful recovery of the object
  - A RecoveryEvent will be fired on each of the recovered objects

Recovery rules

- Property checking and verification in the recovery
  - Fundamental referential integrity is preserved with the existing CE logic. When restore, it is not necessary to check the existence of the required object.
  - In the case of metadata change, recovery process will not try to reconcile, for example, will not to try to honor a new required property value
  - If the recovery of a required OVP fails, the recovery will fail

Purge

- Purge. Permanently remove the object marked for deleted
  - Delete the object and cascadees, as if it were a normal delete ignoring the isMarkedForDeletion flag, thus referential integrity is preserved
  - Purge permission is controlled by the security descriptor of the original object and subject to other constraints as the normal delete
  - Upon successful delete/purge, the corresponding RecoveryItem object is removed from the system
- In the normal delete, all cascadees are deleted regardless of its soft delete status. This is necessary to keep the referential integrity
- If a soft deleted object is not deleted through the purge of RecoveryItem, the corresponding RecoveryItem object will be automatically removed

Access soft deleted objects

- Soft deleted object will behave as if there is no such object from the client view. However there are ways to obtain full or partial information of it
  - An new ObjectStore permission *"VIEW_RECOVERABLE_OBJECTS"* is available in the 5.2 release. If an user has this permission, he can access all objects regardless if the object is soft deleted or not.
  - Soft deleted object can also be accessed as the OriginalObject OVP when fetching the RecoveryItem object. It can only be retrieved in the ***single roundtrip*** when fetching the RecoveryItem object, so you will need set the appropriate property filter to do so
  - If a security proxy is soft deleted, it will still act as a security proxy regardless its soft delete status

Version considerations

- For the 5.2.0 release, we only support soft delete of VersionSeries, not individual Document versions.
- If an OVP points to an individual document version, the propagation of the soft delete flag (cascade) will stop, unless that version is the only version in the version series
- The “OriginalXXX” properties of RecoveryItem will be populated from a document version selected using the same rules to select the current version in a Dynamic Referential Containment Relationship (DRCR). This order is as follows:
  - Released version
  - If there is no Released version, use Current version
  - If no Current version, use reservation

Background tasks

- Background task. All outstanding background tasks will continue be executed regardless of the soft delete status. The background task will have access to the soft deleted objects as if they were not soft deleted. These back ground tasks includes:
  - CBR index request
  - Queue item processing
  - Publish
  - Thumbnail generation
  - Content roll forward
  - ...

Security concerns

- Soft delete an object requires both DELETE permission and VIEW permission on the deleted object
- Soft deleted object is accessible if caller has the object store level permission *"VIEW_RECOVERABLE_OBJECTS"*
- Recovery is controlled by DELETE permission on the RecoveryItem
- The restored object will have the same exact security permissions as before. Those permissions are not affected by who did the restore
- Permissions on the RecoveryItem object is by default inherited from the RecoveryBin. User can customize/overwrite this behavior by creating a subclass with a security proxy property pointing to an different object
- Normal security checking are not affected by the soft delete status, if a security proxy object has been soft deleted, it is still used for the security checking regardless its soft delete status

API - create a RecoveryBin

```
CmRecoveryBin myBin =
        Factory.CmRecoveryBin.createInstance(os,
                "CmRecoveryBin");

myBin.set_DisplayName("First Bin");
myBin.save(RefreshMode.REFRESH);
```

API - mark an object for deletion

```
// suppose we have a Document object called “aDoc”

VersionSeries vs = aDoc.get_VersionSeries();
CmRecoveryItem markedForDeleted =
        vs.markForDeletion(myBin, "CmRecoveryItem");
markedForDeleted.save(RefreshMode.REFRESH);

// The RecoveryItem “markedForDeleted' will appear in “myBin”, “aDoc”
   and all its sibling versions will disappear unless the caller has the
   ObjectStore level “VIEW_RECOVERABLE_OBJECTS”
```

API - recover and purge

- Recover:

```
markedForDeleted.recover();
markedForDeleted.save(RefreshMode.REFRESH);
// The object will be restored to its original state
```

- purge

```
markedForDeleted.delete();
markedForDeleted.save(RefreshMode.NO_REFRESH);
// the markedForDeleted and original object will be completely removed from the system
```

Content Engine Java and .NET Developer's Guide

- Internal Information Center (available now):

  http://cmfogbert.usca.ibm.com:7777/p8ic520/index.jsp?nav=%2F10_2_1_32

- External Information Center (available at eGA):

  http://pic.dhe.ibm.com/infocenter/p8docs/v5r2m0/index.jsp?nav=%2F10_2_1_32

Question and discussion