Kevin Bates
Software Engineer
March 4, 2013

Product Implementation Training (PIT)

IBM FileNet Content Manager 5.2.0 Holds

Introduction

- Course Overview
  - What are Holds and how are they applied?
- Target Audience:
  - Application designers, P8 Administrators, Support Personnel
- Prerequisites:
  - P8 Administration, API experience
- Version Release Date March 15, 2013

Course Objectives

After this course you will be able to:

- Describe a Hold
- Perform the necessary steps to setup a set of held objects
- Explain how and when these are applied

Course Roadmap

- ➔ Hold Fundamentals
  - Why Holds?
  - Object Model
  - Instance Diagram
  - Internals
  - API Best Practices
- – Demonstration
- – Course Summary

Hold Fundamentals: Why Holds?

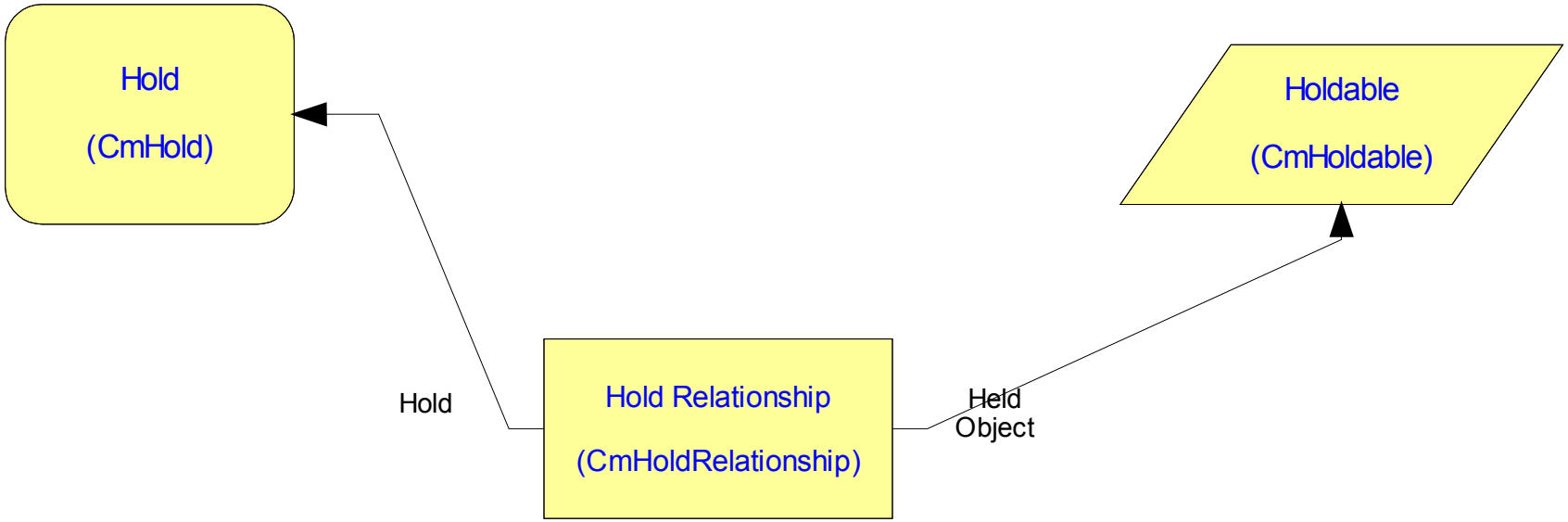
I, as an administrator, need a standard mechanism for applying "holds" preventing certain objects from being deleted.

Use Case: a set of documents will be identified, e.g. by a search, as being relevant to a legal matter and are placed on hold so that they cannot be deleted until the matter has been resolved.

- Deliver **built-in** support for holds
  - Related to retention management conceptually, but a completely independent feature.
  - Provides a standard model for behavior already implemented in IER and eDM products.

Note: Holds are intended to be used by external applications. We expect them to be used by PSS and discovery-type applications, but there are no known client releases on the radar. So, in the short term, Holds will be a CE infrastructure capability that is not exposed in any product yet.

Hold Fundamentals: Object Model



Hold Fundamentals: Object Model

- CmHold
  - A Hold object represents the issue for which documents are being held (e.g., a lawsuit).
  - Holds have a Name, a Description, and an enumeration of HoldRelationships
  - Only Administrators can create/delete Holds
  - A Hold (and its set of HoldRelationships) remain in place until deleted
    - There is no associated RentionPolicy
  - Deleting the Hold cascade deletes all of the HoldRelationships
  - Holds can be sub-classed
    - E.g., LegalHold subclass might add OVP pointing to the case
  - Holds reside in a new table - Hold

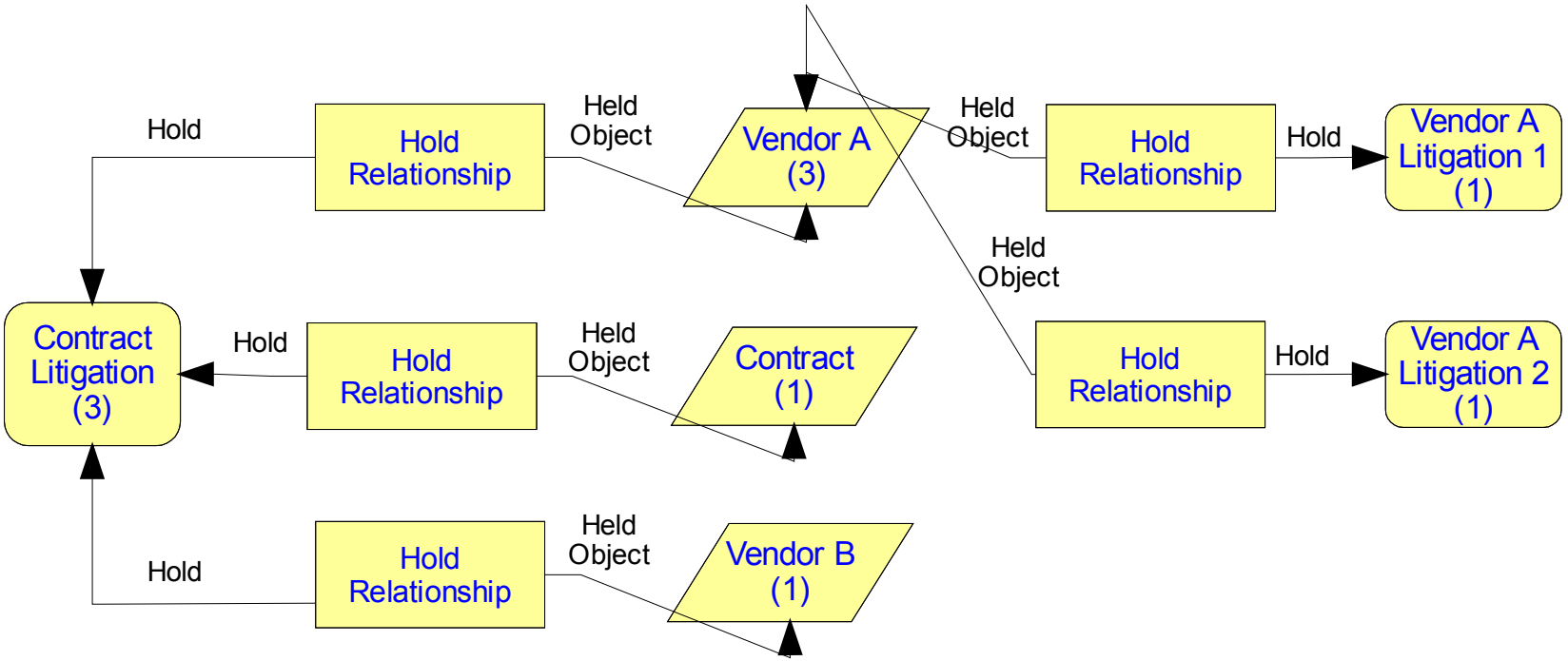
Hold Fundamentals: Object Model

- CmHoldRelationship
  - Relates each held object (CmHoldable) to the Hold object
  - The HeldObject OVP references the held object
    - Has **PreventDelete** constraint so the *held* object cannot be deleted before the *Hold* object is deleted
  - Hold Relationships are not independently securable
    - Security is derived from Hold via full security proxy
    - Only Administrators can create/delete HoldRelationships
  - Cannot participate as target to DeletionEvent or Delete auditing due to optimized behavior
  - HoldRelationships can be sub-classed
  - Resides in a new table – HoldRelationship

Hold Fundamentals: Object Model

- CmHoldable
  - New abstract interface for all classes to which a hold can be applied
    - Essentially all Containable classes plus Annotation (e.g., Document, Folder, Custom Object, Annotation)
  - Not available to Custom Root Classes
  - Defines HoldRelationships enumeration
    - Facilitates navigation from itself to the hold object (i.e., the reasons for which it is being held)
    - Prevent deletion action constraints
  - Once involved in a hold relationship of any kind, it cannot be deleted until that relationship is terminated

Hold Fundamentals: Instance Diagram



Hold Fundamentals: Internals

- By building this behavior into the server, distinct advantages can be realized
  - Cascade delete is extremely fast – direct SQL statement is issued for all Hold Relationships
    - As a result, HRs cannot participate as DeletionEvent targets or audits. Target the Hold object itself.
  - The establishment of redundant Hold Relationships between a Hold and its Held Object are silently ignored
    - Application no longer needs to worry if a given object is held when issuing same query to apply holds to newly added objects
  - When checking duplicates (above) security information is obtained and cached so as to prevent an additional database round trip later when validating the OVP

Hold Fundamentals: API Best Practices

Hold creation is designed around two fundamental ideas

- Query Results Processing and Batching

```
// Select only the Id of the objects to be held using RepositoryRowSet
// Best, of course, if WHERE clause uses a good index
SearchSQL sql = new SearchSQL("SELECT [Id] FROM [Vendor] WHERE VendorName =
                                                    'ACME Dog Food Company'");
SearchScope ss = new SearchScope(os);
// Use continuable query,
// Make page size at least as large as updating batch size
RepositoryRowSet rrs = ss.fetchRows(sql, Integer.valueOf(MY_BATCH_SIZE),
                                                    null, Boolean.TRUE);

// Create UpdatingBatch with NO refresh
UpdatingBatch ub = UpdatingBatch.createUpdatingBatchInstance(domain,
                                RefreshMode.NO_REFRESH);
```

Hold Fundamentals: API Best Practices

```
// Create the Hold and add to the batch - only if there were items returned
CmHold myHold = null;

Iterator iter = rrs.iterator();
if ( iter.hasNext() ) // True if there are query results
{
        myHold = Factory.CmHold.createInstance(os, ClassNames.CM_HOLD,
                                    Id.createId());
        myHold.set_DisplayName("My Hold");
        myHold.set_DescriptiveText("Demonstrate best practices");
        ub.add(myHold, null);
}
```

Hold Fundamentals: API Best Practices

```
// Iterate over the search results adding new Hold Relationship instances
// for each result
while (iter.hasNext())
{
        RepositoryRow rr = (RepositoryRow)iter.next();
        Id vendorId = rr.getProperties().getIdValue("Id"); // Id of to be Held Object

        // Create the HoldRelationship
        CmHoldRelationship holdRel = Factory.CmHoldRelationship.createInstance(os,
                                     ClassNames.CM_HOLD_RELATIONSHIP);
        // Set Hold
        holdRel.set_Hold(myHold);
        // Set HeldObject
        // Important! Use getInstance to build the reference to held object using Id
        holdRel.set_HeldObject(Factory.Document.getInstance(os, "Vendor", vendorId));
        ub.add(holdRel, null);
        holdRelationshipsCreated++;
```


IBM®

Hold Fundamentals: API Best Practices

```
        // Important! Issue batches of some amount
        if ( (holdRelationshipsCreated % MY_BATCH_SIZE) == 0 )
        {
            ub.updateBatch();
            // Create new batch just as before
            ub = UpdatingBatch.createUpdatingBatchInstance(domain,
                        RefreshMode.NO_REFRESH);
        }
} // end of while

// Issue remaining batch if there's anything to do -
// pendingExecute will be false if batch is empty
if ( ub.hasPendingExecute() )
{
        ub.updateBatch();
}
```

Course Roadmap

- Hold Fundamentals
  - Why Holds?
  - Object Model
  - Instance Diagram
  - Internals
  - API Best Practices
- ➔ Demonstration
- Course Summary

Demonstration – ACME Dog Food Company

- ACME Dog Food Company cans a variety of dog food in lots of 10,000 cases per day. Each case is tracked by IBM FileNet Content Manager.
  - For each of the “Life Stages” – Puppy, Youth, Adult and Old Dog – there are four primary flavors – Beef, Chicken, Salmon and Soy – each with varying secondary flavors
- The company just discovered that they were using bad soy across their life stages and now face a lawsuit for the bad soy
- It gets worse. The company also discovered that their entire Old Dog line is making old dogs sick – regardless of the primary flavor – and are facing lawsuits on that front as well!
- The following will demonstrate how two separate holds can be applied (“Bad Soy” and “Sick Old Dog”) such that deletion of the related tracked objects is prevented.

Course Roadmap

- Hold Fundamentals
  - Why Holds?
  - Object Model
  - Instance Diagram
  - Internals
  - API Best Practices
- Demonstration
- ➔ Course Summary

Course Summary

You have completed this course and can:

- Describe a Hold
  - *A Hold is an object that can easily apply a deletion prevention constraint to a set of objects until it is deleted*
  - *It is applied via query results processing and batched creation of Hold Relationships*
- Perform the necessary steps to setup a set of held objects
  - *Query for the objects to which the Hold will be applied*
  - *Walk the results, creating Hold Relationships for each of the referenced objects – batching the creation requests along the way*
  - *Delete the Hold instance once deletion prevention is no longer required*
- Explain how and when these are applied
  - *The primary use case is for litigation where it might be necessary to quickly identify related objects and immediately prevent their deletion until the litigation has been resolved*

Product Help/Documentation/Resources

- Content Engine Java and .NET Developer's Guide
  - Hold Concepts

    http://pic.dhe.ibm.com/infocenter/p8docs/v5r2m0/topic/com.ibm.p8.ce.dev.ce.doc/hold_concepts.htm

  - Working with Holds

    http://pic.dhe.ibm.com/infocenter/p8docs/v5r2m0/topic/com.ibm.p8.ce.dev.ce.doc/hold_procedures.htm

Note: Links will work at eGA (3/15/2013); before then, replace http://pic.dhe.ibm.com/infocenter/p8docs/v5r2m0 with http://cmfogbert.usca.ibm.com:7777/p8ic520 to use an internal InfoCenter