IBM Enterprise Records Performance Best Practices

TABLE OF CONTENTS

1 Introduction

This document provides guidance for implementing and maintaining a records management system. This document is written for users who design file plans, implement and maintain an IBM Enterprise Records system, as well as for database administrators and other users who support the system.

The IBM Enterprise Records system is tightly integrated with the underlying P8 Content Engine (CE) server. As a result, the performance of the IBM Enterprise Records system is highly dependent on the performance of the CE server, including the underlying application server and database components.

The examples provided here are meant to illustrate the differences between the broad approaches for optimal performance and are not to be taken as precise templates for your particular business requirements. There are many subtle details to take into account when creating a file plan configuration to meet precise business requirements.

1.1 Terms

FPOS – File Plan Object Store. The FPOS contains the file plan structure and the RecordInfo Object (RIO) instances.

ROS – Records enabled Object Store. This is the object store that contains electronic content. This content may be declared as records.

Record – The electronic document or physical entity that is to be managed.

RIO – Refers to the RecordInfo Object that is used by IBM Enterprise Records to manage the record. The RIO object is stored in the FPOS.

CE – IBM FileNet P8 Content Engine.

BDS – Bulk Data Services API. This API includes services for record declaration and destruction.

Auto Destroy – This refers to destroying records without requiring any workflow processing. Auto Destroy is performed by running Disposition Sweep with the autodelete option. In this document, the term Auto Destroy is used as shorthand for running the Disposition Sweep process using the autodelete option.

Category – The short form of Record Category.

Schedule – The short form of Disposition Schedule.

1.2 Required Concepts

IBM Enterprise Records best practices require knowledge of FileNet P8 concepts. The following reference information is available:

- General concepts are documented in the IBM Enterprise Records online help.

- General tuning guidance for a FileNet P8 environment is provided in the FileNet P8 Performance Tuning Guide. Use the appropriate link in the References section.

2 File Plan Modeling for Performance

The following section presents five file plan models that can be used as guides for building an optimized records management environment. Again, these are presented as guides. Actual implementations must be adapted with special attention being paid to the business context for that scenario.

- Retention
- Year Record
- Case
- Record Driven
- Delayed Aggregation

Adapt one or more of these models to meet your business practices.

2.1 Retention Model

Use the retention model when the primary goal of the records management system is to retain information for a set period of time. An email archiving application that requires all emails to be kept for a period of, for example, 30 days, is an excellent candidate for this model.

The retention model is in contrast to other file plan models where documents or folders follow a disposition cycle based on events or business context. In the retention model, the required retention period of the document is known at the time when the record is declared, and there are no additional requirements with regard to how or where the records should be filed.

You can use record categories to implement the retention model. This implementation offers both high performance and flexibility.

The following approach can be used to implement the retention model.

- A three-level record category hierarchy:
    - Application

      This is the top level of the hierarchy and is used to group all the categories for a particular application of the retention model. For example, if email originating from the Human Resources department needs to be handled separately from email originating from the Billing department, create separate record category hierarchies for each.

- Retention Period

  This is the second level of the hierarchy and is used to group all the categories that have a similar retention period. For example, if some email needs to be kept for 30 days, some for 60 days, and some for 90 days, create three categories at this level of the hierarchy.

- Retention Group

  This is the third, or lowest, level in the hierarchy. These categories are used to group all records that need to be destroyed on the same day. Use a naming convention that includes a date for the categories at this level of the hierarchy. For example, there might be one category for every day of the year so that all email received on a specific day is filed in the same category, and the category names will be based on the date the email was received.

- Automation for
  - Creating record categories at the retention group level of the hierarchy.
  - Record declaration.

  The automation can be built using
  - The P8 Java API and the IBM Enterprise Records BDS API
  - Content Engine custom events
- A Disposition Schedule for each record category at the retention period level of the record category hierarchy. All the record categories at the retention group level of the hierarchy inherit the schedule. The schedule has the following characteristics:
  - Aggregation occurs at the record category level.
  - Includes at least one phase in which the phase action is set to Auto Destroy and the phase retention period is set to zero.
  - Disposition Event Offset is user-defined, but should not be zero.

    A category is closed when cutoff occurs which prevents additional records being filed into the category. To start the clock on the disposition process, but still allow records to be added to the category, set the disposition event offset to a non-zero value.
  - The Cutoff Base is set to a date property that is common to the originating document, RIO, and the record category.

    The definition of the OOTB record category class must be updated with this property.

    Using a common property on the originating document, RIO and record category enables you to programmatically identify the documents to be declared as records, create an appropriately named record category, and file the records into the correct category.

- An internal event that is triggered when the value of the date property used in the Cutoff Base is no longer null.

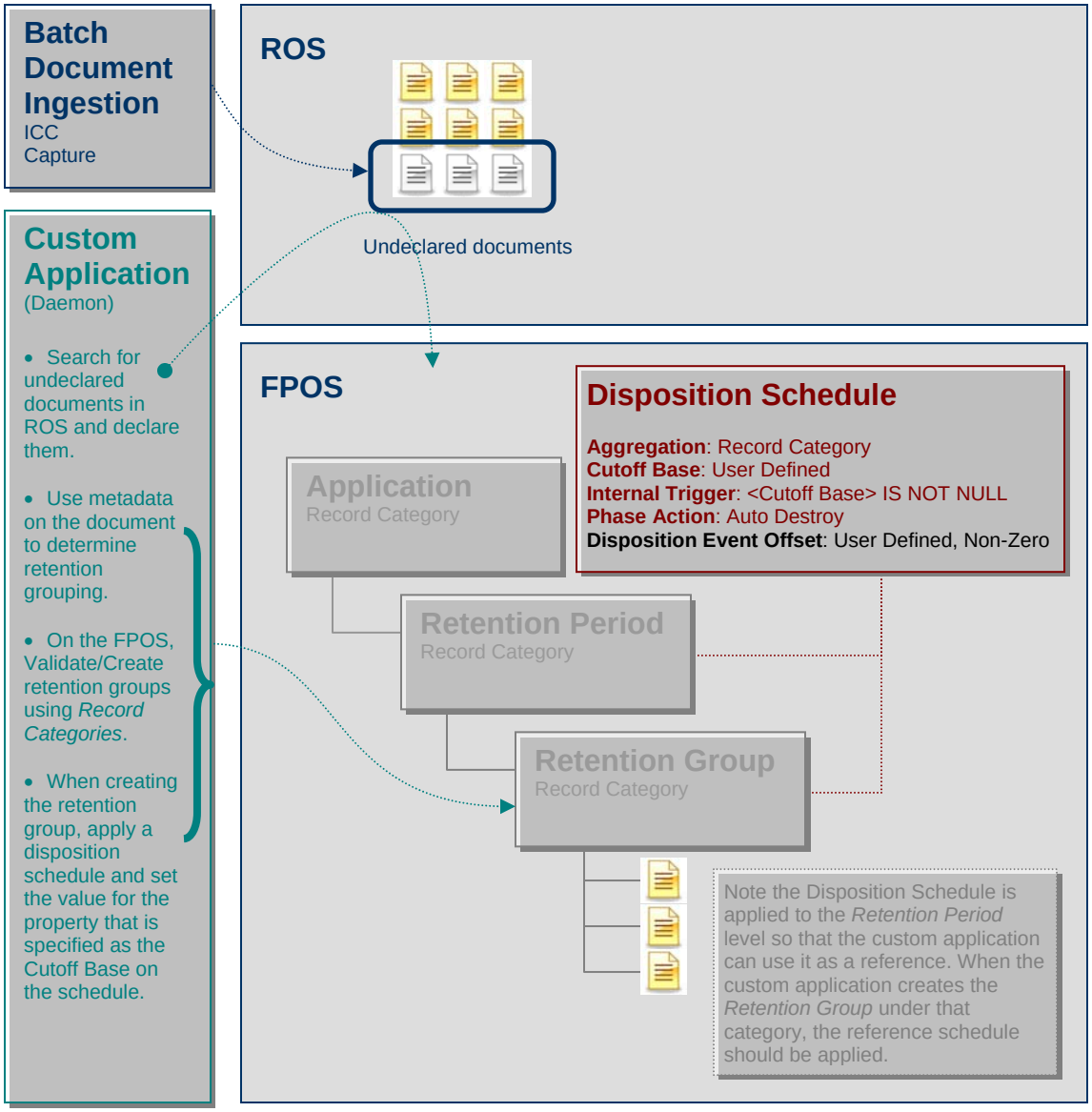
2.1.1 Retention Model Scenario Example

The following example describes how the common property used in the Cutoff Base and internal trigger is used.

- The document classes used for the email document, the RIO, and the record category are updated to include a date property called SentOn.
- The schedule uses the SentOn date property as the Cutoff Base.
- The internal event used by the schedule, is triggered when the SentOn date property has a non-null value.
- A custom application
    1. Identifies all the documents that need to be declared as records because they have the SentOn property set to a specific value.
    2. Checks to see if a category already exists that includes this specific SentOn value in its name. If the category does not exist, the application creates the category and sets the SentOn property value on the category to the same value as that used by the documents.
    3. Declares all the documents identified in step 1 as records and files them into the category created in step 2.

       The internal event associated with the schedule is triggered because the SentOn property is no longer null and this causes the cutoff phase to begin. Since the cutoff action does not occur until after the time defined by the Disposition Event Offset expires, records can continue to be added to the category until cutoff occurs.

The following diagram shows the general configuration for the record category implementation of the Retention Model.



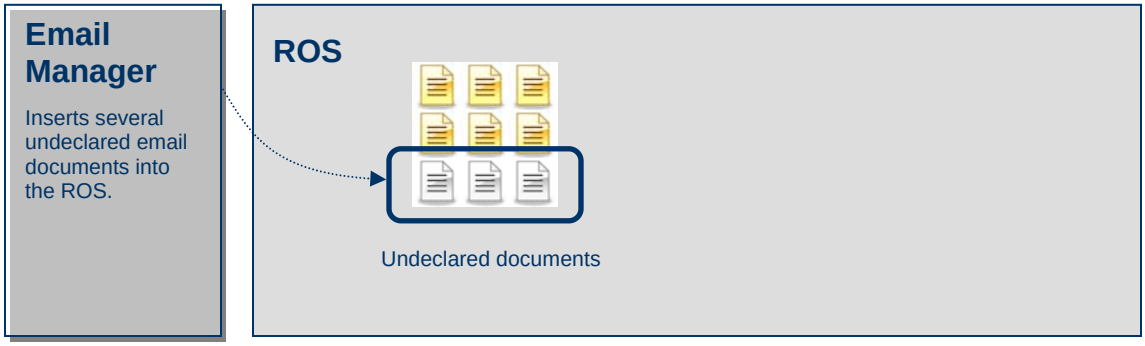
Advantages

- Records are grouped by a common retention period and retention date. This allows for optimized processing over the entire group.
- Record category processing performs better than volume processing.
- Both back file conversion applications and day forward applications can take advantage of this model.

Disadvantages

- Unless the Retention Model logic is implemented in an event, custom programming is required to retrieve the appropriate documents to declare as records.
- Custom programming is required to create the Retention Group categories and to apply schedules to them.

The following example shows a scenario where Email Manager is used to ingest a high volume of email documents into an ROS.



A custom application is used to declare these documents into an FPOS. In this example, the property names, property values, schedules, and dates are arbitrary and are used only for the purposes of this example. In this example

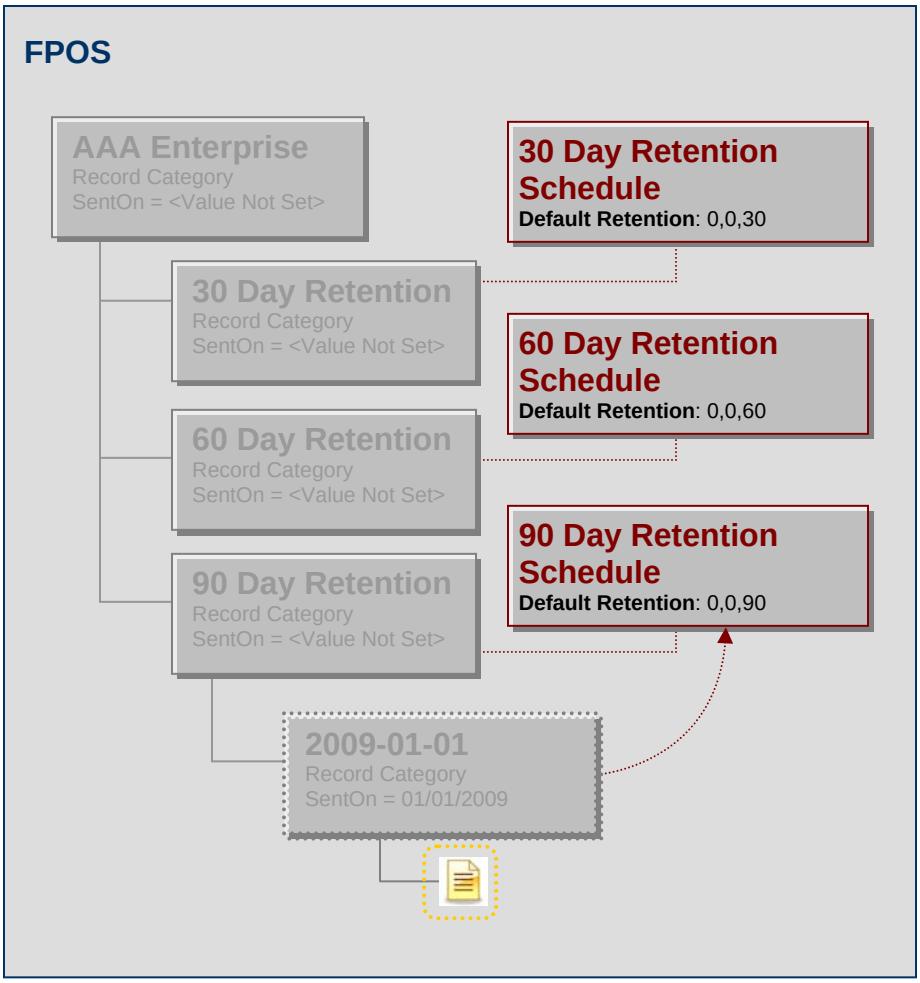
- The EmailSubject property value determines which Retention Period to file the record under
- The SentOn property value determines which Retention Group to file the record into.
- Three Retention Period categories are created, 30 Day, 60 Day, and 90 Day.
- Each Retention Period is associated with a schedule. The schedule for the 30 Day category uses

    - A two-day Disposition Offset Event (0, 0, 2) to allow emails to be added to the category for up to two days after the category was created.
    - The SentOn property as the Cutoff Base property
    - A single Phase that uses a default retention period of 30 days, the Auto Destroy action, and an Internal Trigger in which the ‘SentOn IS NOT NULL’ condition triggers cutoff.

The custom application logic for this example is as follows.

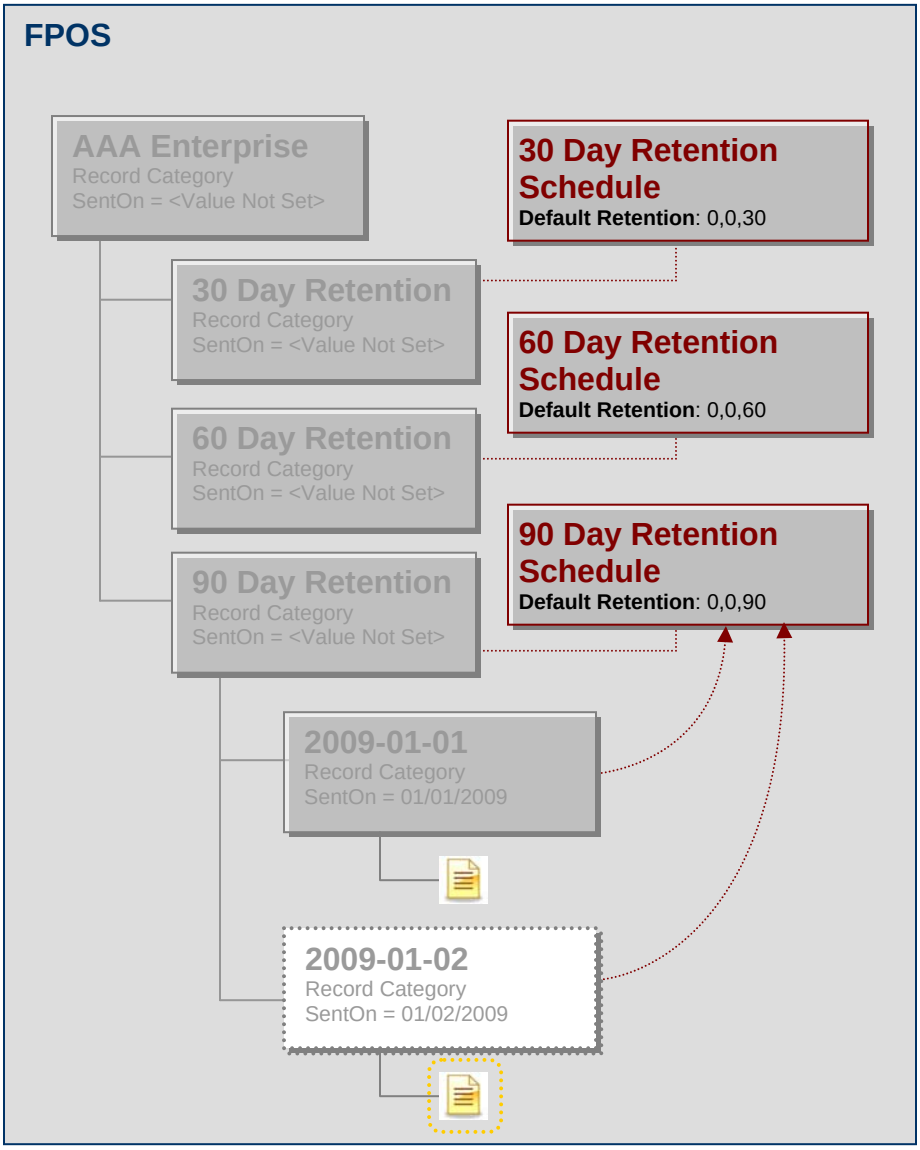
- Use the P8 Java API to query for undeclared email documents.
    - Ensure that the result set is paged in order to prevent using too much memory.
- Iterate through the email document result set.
- For each email document, use the EmailSubject property to determine the retention period and the SentOn property to determine the retention group.
    - The first document has these values
        - EmailSubject = “AAA Enterprise Stock Purchasing Policy”
        - SentOn = 01/01/2009
    - Based on the declaration rules in the custom application, the word ‘Stock’ in the subject line qualifies the email document for the longest retention period of 90 days.
    - Perform a search in the 90 day category for the 2009-01-01 retention group. If the category does not exist:
        - Create the category and
            - Set the SentOn property to 01/01/2009.
            - Apply the schedule that is associated with the parent category, the 90 day category.

- Declare the record into the new category.



- The second document has these values.
  - EmailSubject = “RE: AAA Enterprise Stock Purchasing Policy”
  - SentOn = 01/02/2009
- The word ‘Stock’ qualifies the email document for the longest retention period of 90 days.
- Perform a search under the 90 day category for the 2009-01-02 retention group. If the category does not exist:
  - Create the category and
    - Set the SentOn property to 01/02/2009
    - Apply the schedule that is associated with the parent category, the 90 day category.

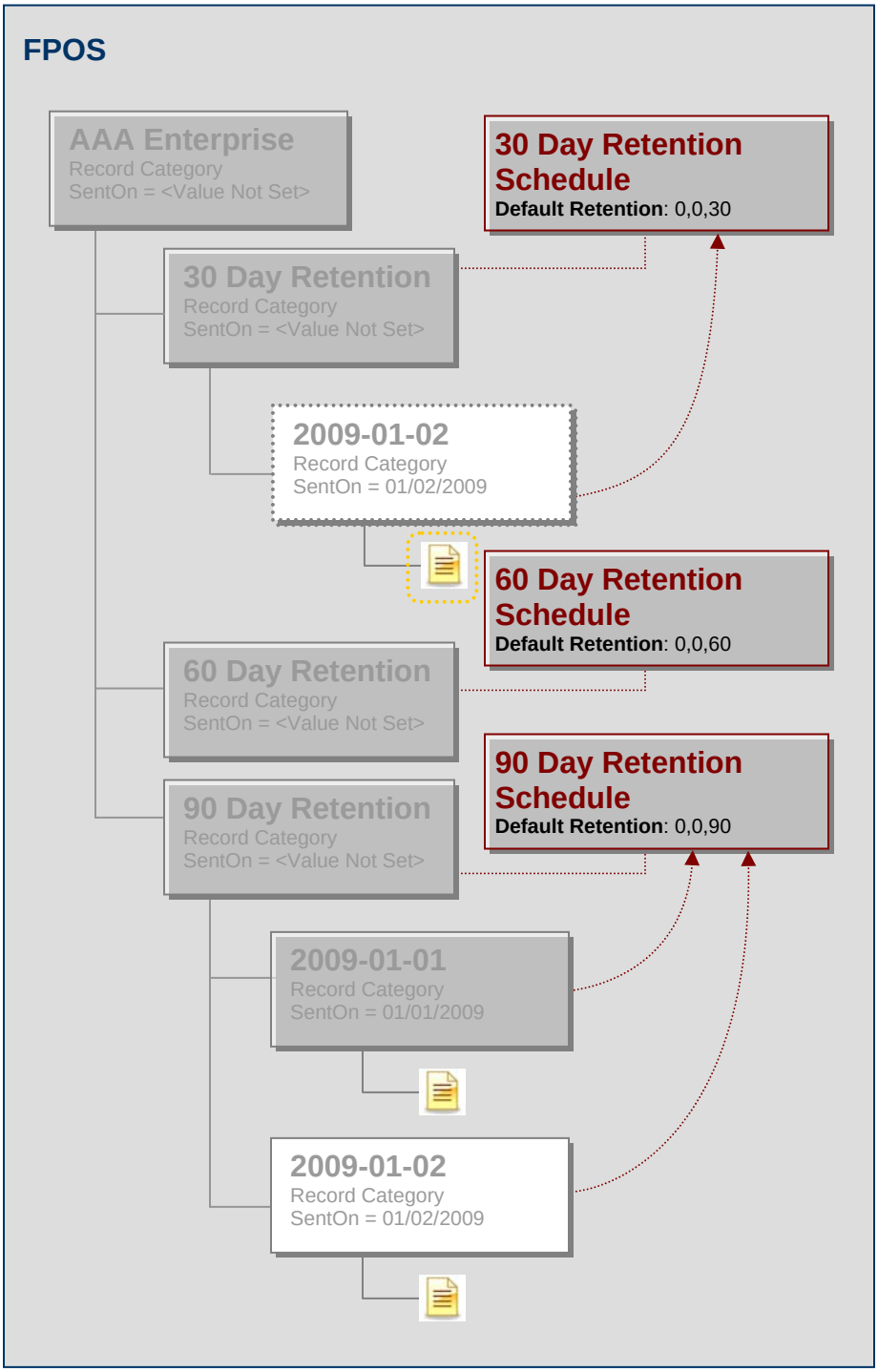
- Declare the record into that category.



- The third document has these values:
  - EmailSubject = “Vacation Schedule”
  - SentOn = 01/01/2009
- No words in the EmailSubject property trigger a rule beyond the default retention of 30 days.
- Search under the 30 day category for the 2009-01-01 retention group. If the category does not exist
  - Create the category
    - Set the SentOn property to 01/01/2009.

- Apply the schedule that is associated with the parent category, the 30 day category.

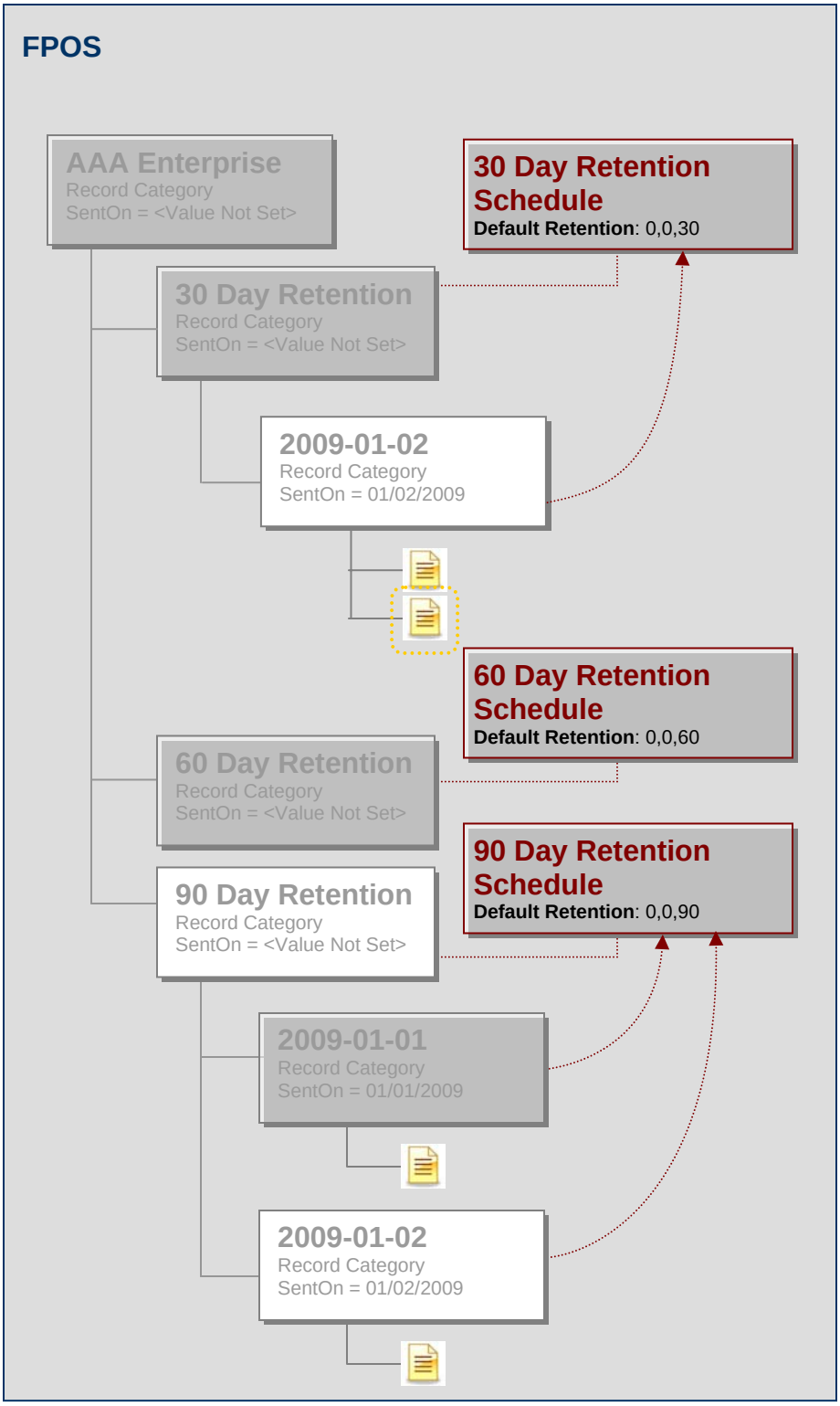
o Declare the record into the category.



The fourth document has these values:

- EmailSubject = “Vacation Schedule”
- SentOn = 01/01/2009

- No words in the EmailSubject property trigger a rule beyond the default retention of 30 days.
- Search under the 30 day category for the 2009-01-01 retention group. If the category does not exist
  - Create the category
    - Set the SentOn property to 01/01/2009.
    - Apply the schedule that is associated with the parent category, the 30 day category.
- Declare the record into the category.



This example has illustrated how to implement the retention model using record categories. This model may also be implemented using a combination of record categories and record folders. The advantage of using record folders is that subclassing for record folders is supported through the IBM Enterprise Records web administration application, which facilitates adding custom trigger date properties to the appropriate containers and managing those containers separately from the higher-level categories. Subclassing of record categories is not supported in the web administration application. However, as shown in the example in this section, you can add a trigger date property to the main Record Category class.

2.2 Record Year Model

In the previous section, the retention model illustrates an approach that is driven by high volume requirements where the business context of each record is not known and records are only classified according to a fixed set of retention periods instead of their business context. Such an approach might be appropriate for a high-volume email archiving solution. However, for a more general records management solution, file plan models that take the business context into account are more common and might be more appropriate. The Record Year model offers such an approach.

Use the Record Year model when the retention requirements are primarily based on year of record and you can group all of the same type of records for a given year together so that they can be destroyed together. For example, if your retention rules require that all correspondence be kept for 5 years, all invoices be kept for 7 years, and all reports be kept for 3 years, you can use record categories (or a combination of categories and folders) to organize these records for efficient disposal.

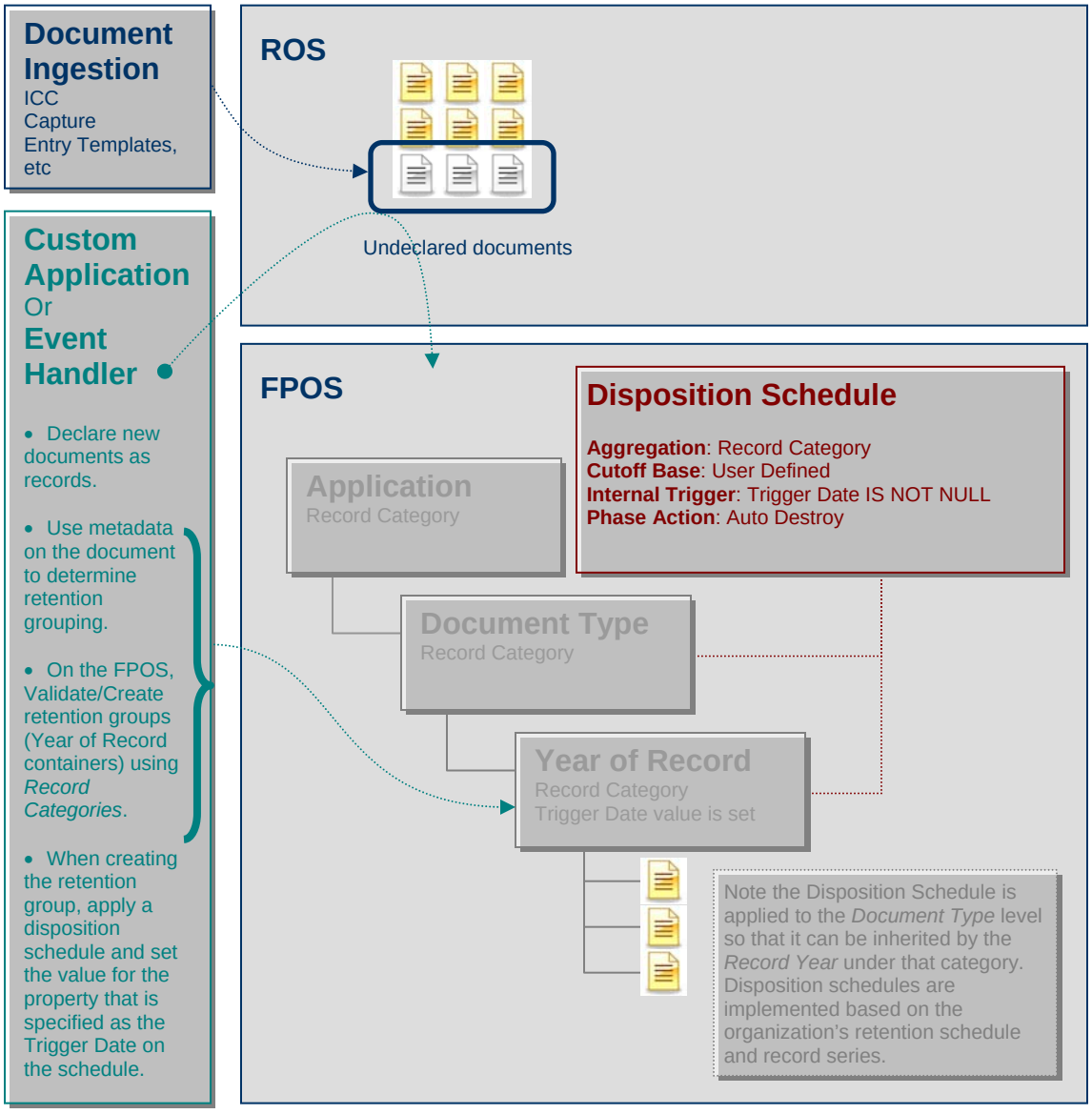
The record year model, at its simplest, can be thought of as a variation of the retention model. The following approach can be used to implement the record year model.

- Three-level category hierarchy:
    - Business function or application – level 1
        - Level 1 of the file plan is used to separate records for different area of the business. For example, HR records are usually kept separate from Finance records.
    - Document or record type – level 2
        - Level 2 categories are used to group records according to their type for purposes of retention. For example, Finance might include Correspondence, Invoices, and Reports.
    - Year of Record – level 3
        - Level 3 containers are used to group records by each year (calendar, fiscal, or however you like to group them). You would create a container for each record year and aggregate your disposition at this container level.
- Automation for declaring records – records are declared into the appropriate year of record container based on level 1 and 2 attributes of the records plus the year of record as determined by some date value or other attribute of the record at the time of declaration. For example, Correspondence for Finance would be declared into the appropriate Year of Record container based on either the date scanned (for paper) or date received (for email), or for go forward applications, you can use the Date Created property on the ROS.

- Automation for creating level 3 containers and setting the appropriate trigger date on each container based on the end of the record year.
- A disposition schedule for each category at the document or record type level – each record year container under a given document type category will inherit the same schedule.
    - Aggregation occurs at the record container level (whichever container type you are using for level 3).
    - Retention period is specified appropriately for the destroy phase such that the retention base is the trigger date of the container.
    - Set the cutoff base and event offset appropriately to make sure the container is not closed to early to accommodate lagging documents if required
- You must define the appropriate trigger date property for the container.
- Trigger date value is only assigned to the Record Year containers. Level 1 and 2 categories should not have a trigger date value.
- You must use auto-destroy for the destruction phase to take advantage of the feature that allows individual records to be placed on hold while not holding up the entire container.

2.2.1 Record Year Model Scenario Example

The following diagram shows the general configuration for the record category implementation of the Record Year Model.



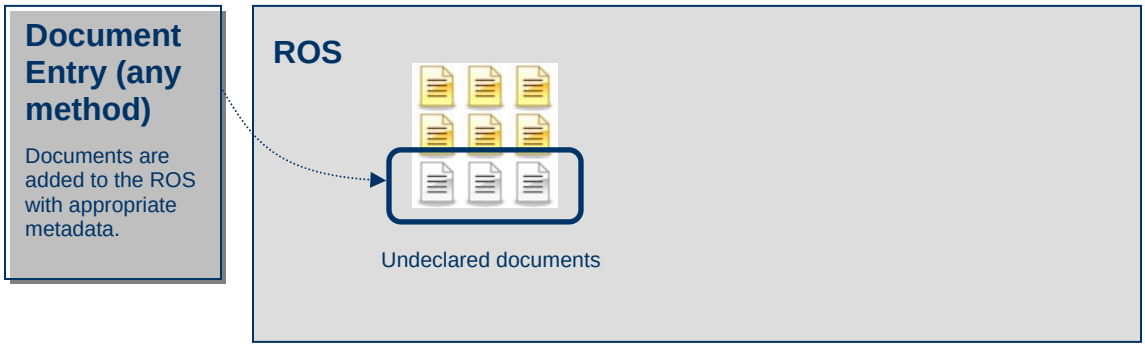
Advantages

- Records are grouped by a common document or record type and retention date. This allows for optimized processing over the entire group.
- Record categories for the retention grouping (Year of Record) can be created ahead of time or as needed depending on level of automation desired. The number of containers required is much less (one per year for each record type) than would be required by the retention model (one per day for each retention group).
- Both back file conversion applications and day forward applications can take advantage of this model.
- Records can be destroyed in aggregate for an entire year.
- Works well in conjunction with auto-destroy, since individual records that are placed on hold will not prevent other records in the container from being destroyed.

Disadvantages

- Unless you are using auto-destroy, records placed on hold within a given year container will implicitly prevent disposition for the entire year.

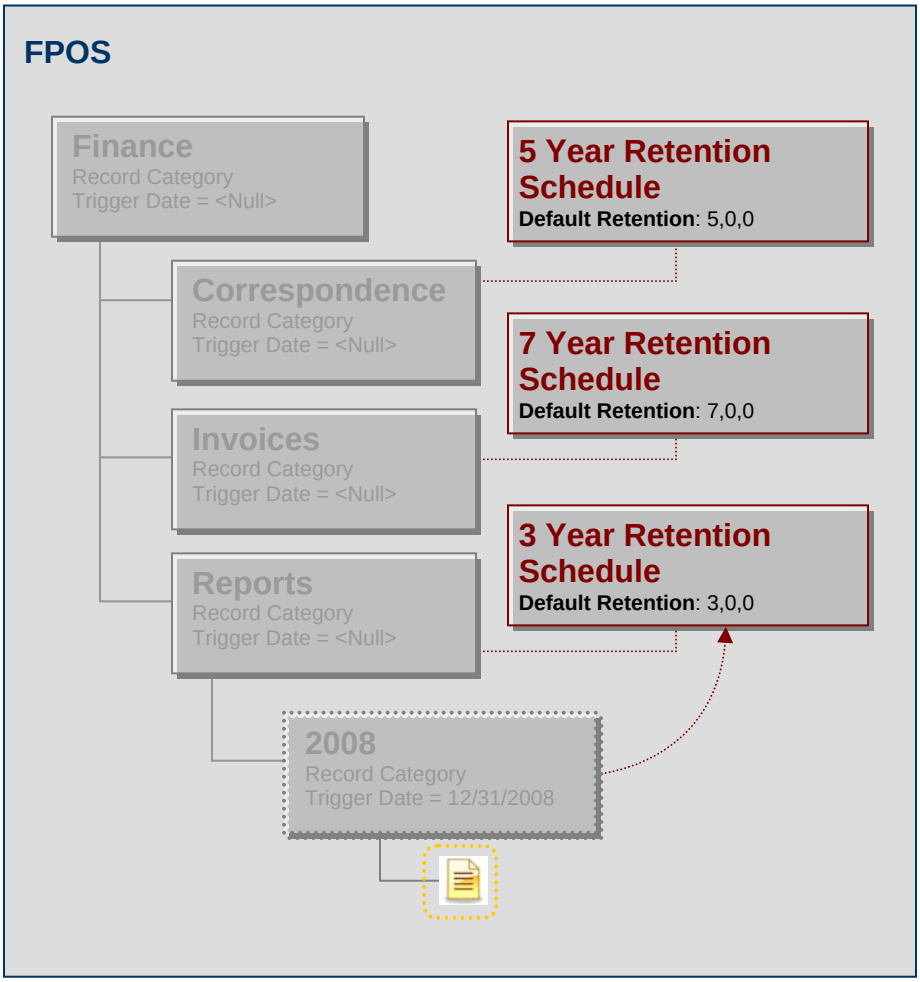
The following example shows a scenario where documents are added to an ROS over time with appropriate metadata.



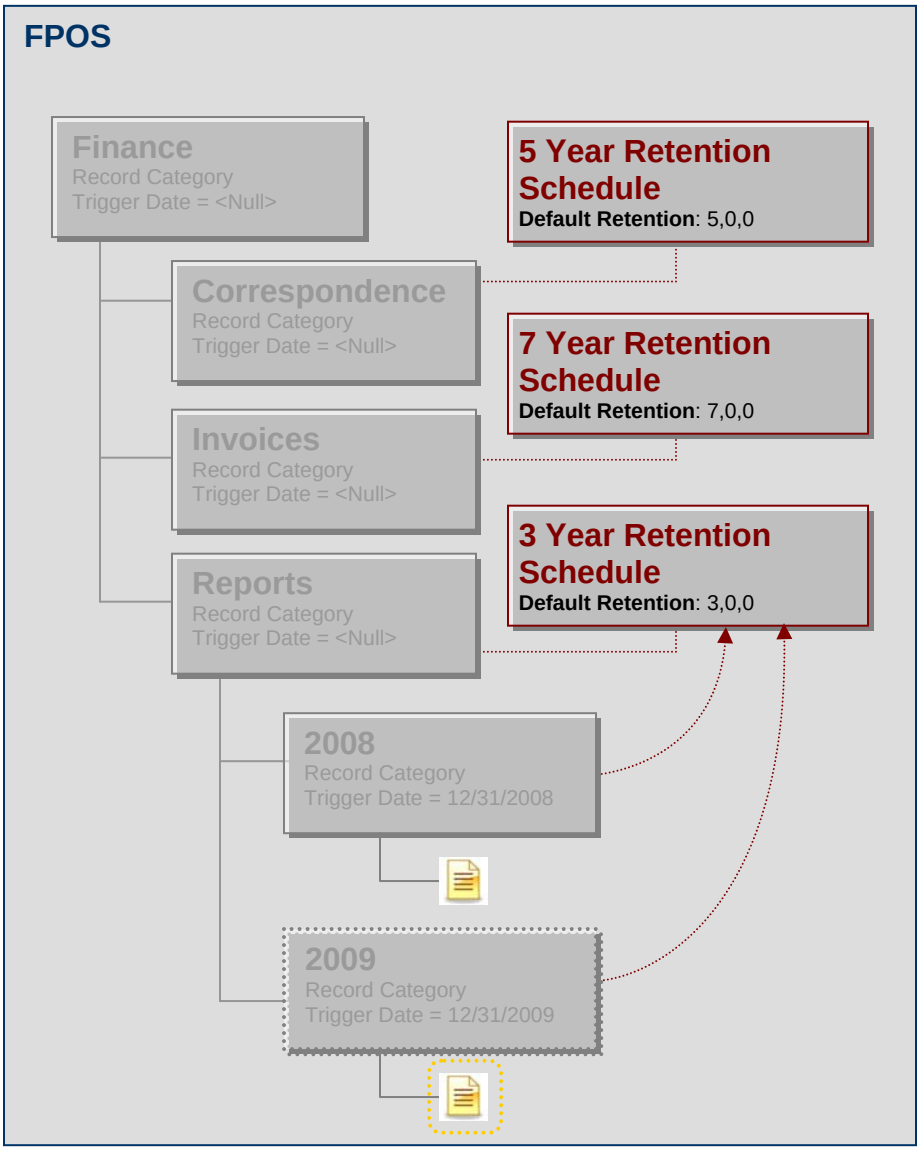
An event handler is used to declare these documents into an FPOS. In this example, the property names, property values, schedules, and dates are arbitrary and are used only for the purposes of this example. In this example:

- The Business Function property value determines the Level 1 category.
- The Document Type property value determines the Level 2 category.
- The Date Created property (the ROS entry date) value determines the Level 3 category (Record Year).

- The event handler will declare each record into the appropriate category based on these 3 properties.
    - For example, the first document has been identified as belonging to Finance and as a Report (based on metadata assigned during document entry or ingestion).
        - Business Function = “Finance”
        - Document Type = “Report”
        - Date Created = 08/26/2008
- The event handler determines that the declaration path should include “/Finance/Reports/2008” based on the metadata provided.



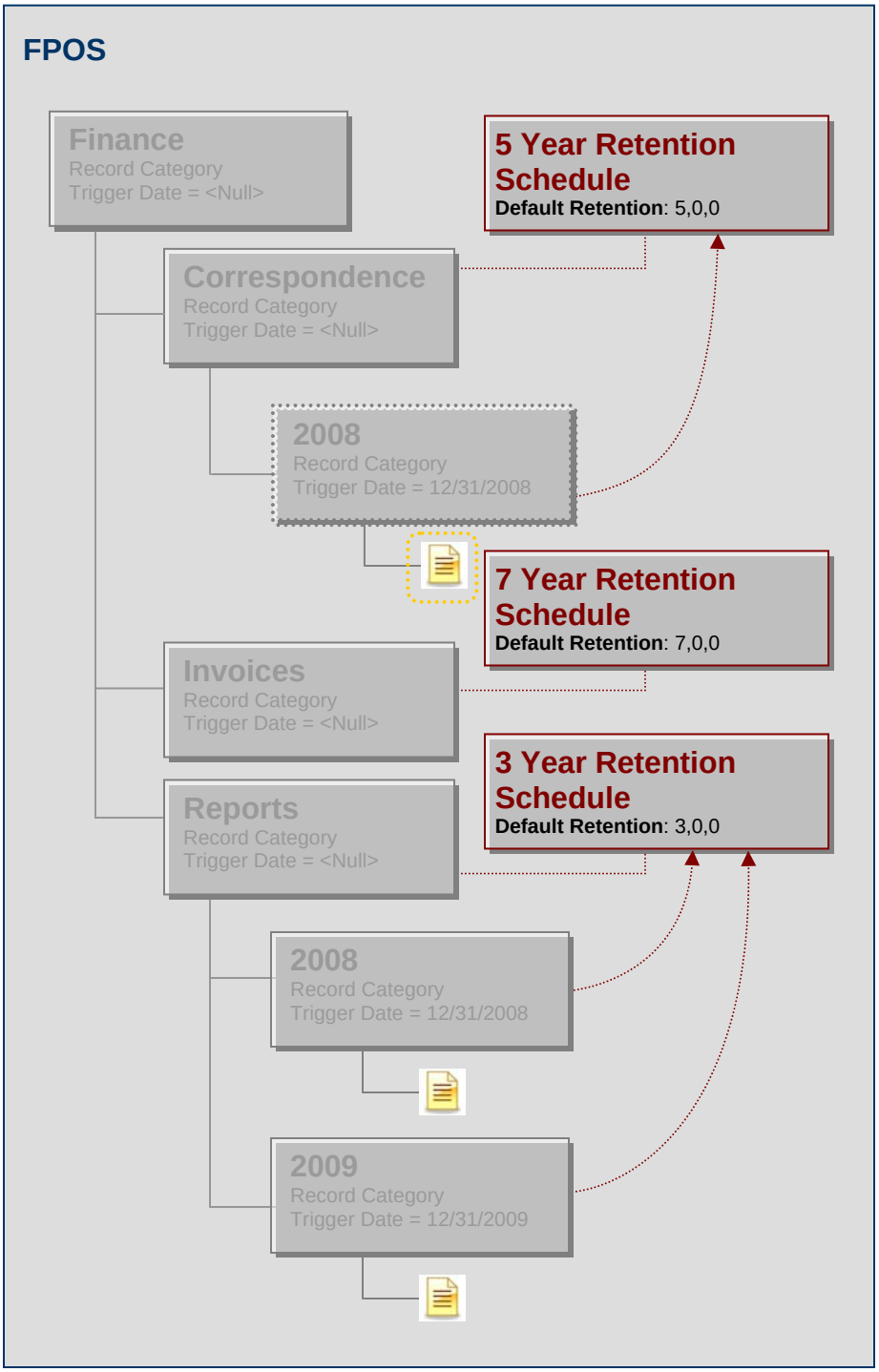
- The second document has these values.
  - Business Function = “Finance”
  - Document Type = “Report”
  - Date Created = 10/13/2009
- The event handler determines that the declaration path should include “/Finance/Reports/2009” based on the metadata provided.



The third document has these values.

- Business Function = “Finance”
- Document Type = “Correspondence”
- Date Created = 12/03/2008

o The event handler determines that the declaration path should include “/Finance/Correspondence/2008” based on the metadata provided.



This particular example relies on the Date Created property on the ROS and will work well for any day forward scenarios or applications. For a back file conversion or ingestion application, you could apply the same approach but would need to base the Record Year on some other date property that was available during ingestion.

2.3 Case Model

Use the case model to group documents on a related subject together; for example, the case model is well suited to handling all documents related to a specific loan, legal case, or building project. In this model while all the documents need to be disposed of at the same time, they might

- Have different formats.
- Be created, reviewed and finalized at different times.
- Be stored in different locations in the same or different records object stores.
- Be ready for record declaration at different times.

In the case model a single record folder (or record folder derivative) is created to hold all the documents related to a specific case and a naming convention is adopted such that each record folder is clearly connected to a specific case.

The association between the documents and the record folder can be tracked using a property value on the documents and on the record folder.

The trigger to initiate disposition is an update to a property value on the record folder. This update on the record folder can be performed directly, or could be the result of an update on a document or a folder in the records object store.

Other characteristics of the case model are

- The disposition schedule is applied to the parenting record category and is inherited by the record folder.
- Aggregation occurs at the record folder level.
- All schedule details such as cutoff base, internal trigger, phases, and disposition offset event are user defined.

To further optimize performance, consider using the following with the case model:

- A single phase action set to autodestroy.

  The case model concept is that all the documents filed in a record folder have been selected specifically because they need to be handled using the same disposition process. Therefore processing the records through additional phases and reviewing each record individually prior to destruction should not be necessary.

- Set the disposition offset event in the schedule to a value other than null.

  This allows the disposition process to be triggered, but keeps the record category open so that additional records can be filed in the category for the period of time identified in the offset.

To implement this model efficiently, some customization is required to

- Automatically create record folders for new cases.
- Detect when case documents are ready to be declared as records.
- Initiate disposition by updating a property on the record folder.

The customization can be created using a combination of all or any of the following:

- Content Engine events
- IBM Enterprise Records Java API
- IBM Enterprise Records BDS API

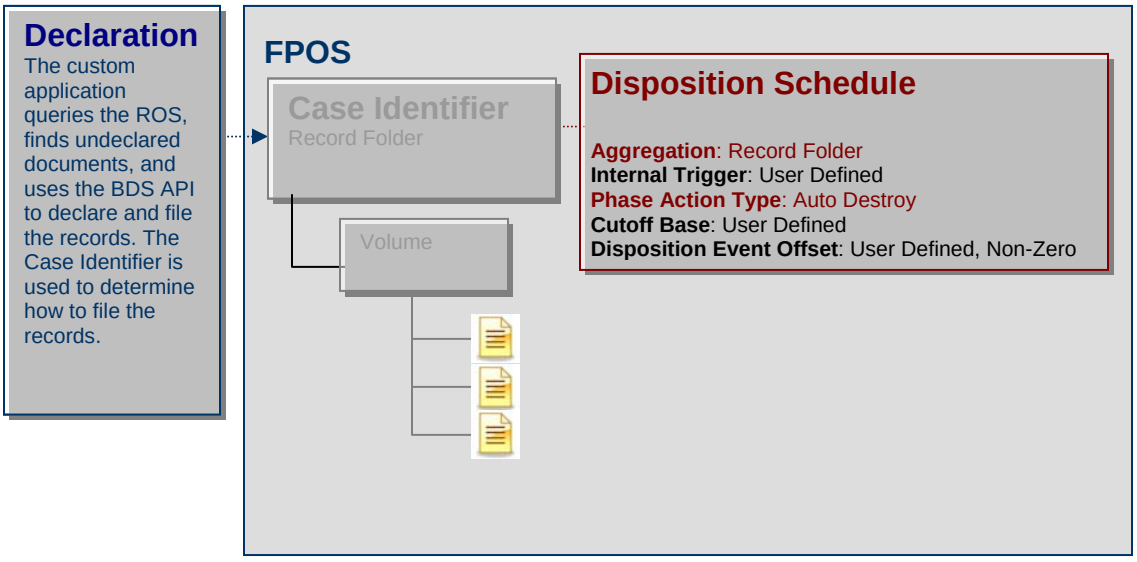
Advantages

- Records are grouped by a common case identifier. This allows for optimized processing at the container level.
- Naturally balances the records across several containers.

Disadvantages

- Sufficient information must be known prior to declaring the documents as records so that the declaration code can determine
  - Which record folder to file each record in
  - That a document is ready to be declared a record
- For high volume scenarios, custom programming is required.

The following diagram illustrates the case model.

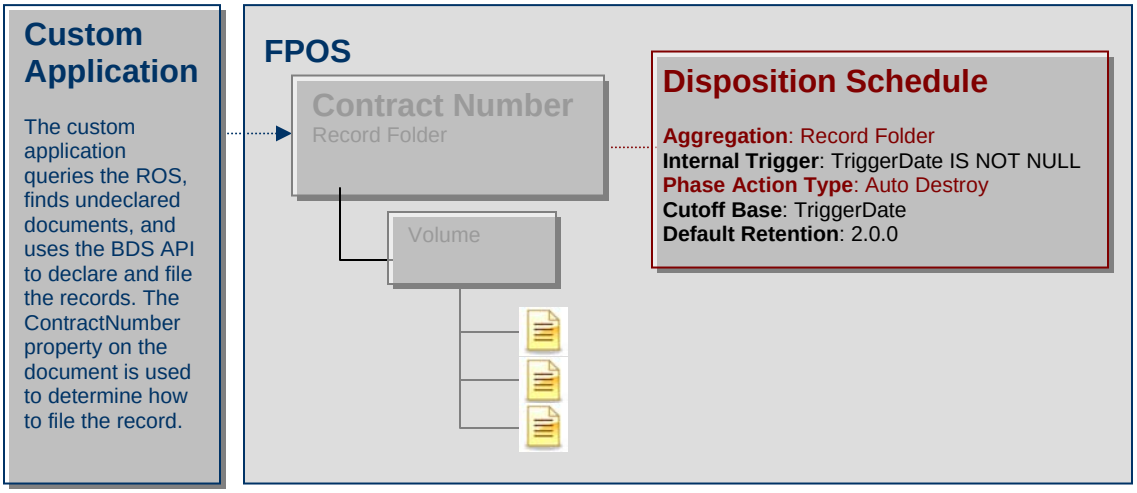


2.3.1 Case Model Scenario Example

In this scenario,

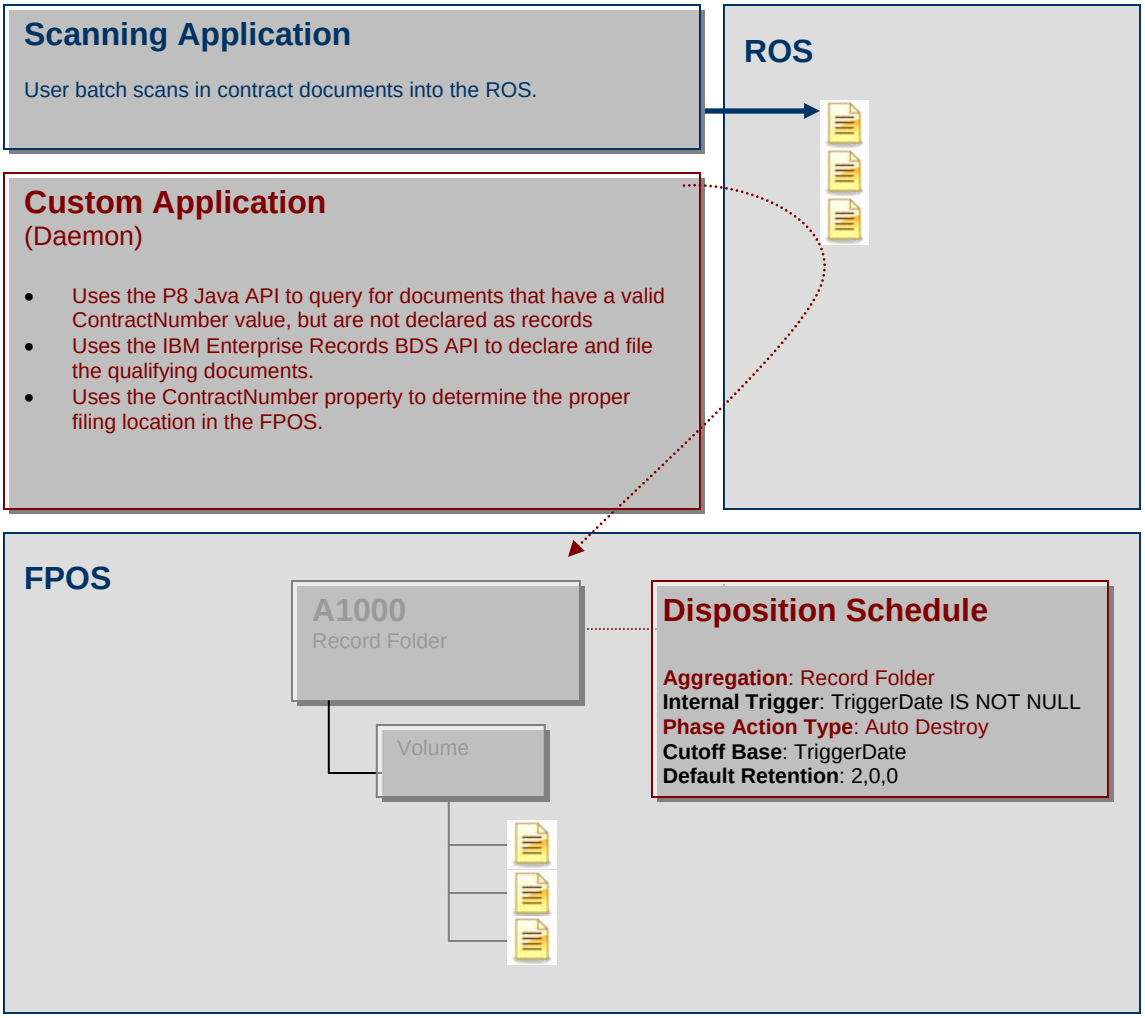
- Documents related to specific contracts are being declared as records.
- The record folder name is set to the contract number.
- A custom property on the record folder (TriggerDate for instance) is used to trigger disposition.
- The disposition cycle starts when the custom property (TriggerDate for instance) is not null.
- All the documents associated with the case are automatically destroyed two years after the date provided by the customer property (TriggerDate for instance).

The following diagram shows the general file plan construction and configuration for this example.



In the following diagrams, three documents are found belonging to contract number A1000. They are declared and filed using the BDS API. Later, the contract is closed and the record folder under which the records are filed starts its disposition cycle. After two years, the record folder and the records it contains are destroyed.

The following diagram shows the flow of declaration and filing.



In this example, the Content Engine SQL query to select qualifying documents is similar to the following:

```
SELECT
      Id,
      ContractNumber
FROM
      Document
WHERE
      ContractNumber IS NOT NULL AND
      RecordInformation IS NULL
```

Note that the 'RecordInformation IS NULL' condition distinguishes declared documents from documents that are not declared. The document has been declared a record when RecordInformation IS NOT NULL.

The results from this query can be used with the IBM Enterprise Records BDS API to declare and file the documents.

The A1000 record folder has the following disposition cycle:

- The contract is closed.

- A user or application updates the TriggerDate property on A1000 record folder to reflect the contract close date.
- Disposition Sweep executes and the A1000 record folder qualifies to move to the first phase of its disposition.
- The phase execution date (the date when the phase is actually scheduled to execute) is set to Cutoff Base + phase default retention.
- After two years, when Auto Destroy executes, the phase execution date on the A1000 record folder is less than or equal to the current date. Therefore, the phase is executed and the record folder and its entire contents are destroyed.

2.4 Record Driven Model

The Record Driven model applies to documents under these conditions:

- The documents follow a disposition cycle that is based on metadata changes.
- The metadata changes are unique to the documents.

In the Case Model example, multiple documents were grouped by a single contract number. However, it is not usable in situations where every document represents a unique contract and closes independently of other contracts.

The Record Driven Model has a negative impact on performance. If possible, implement a model that uses container level aggregation. If that is not possible, then use the following best practices to ensure acceptable performance.

The Record Driven Model has the following attributes:

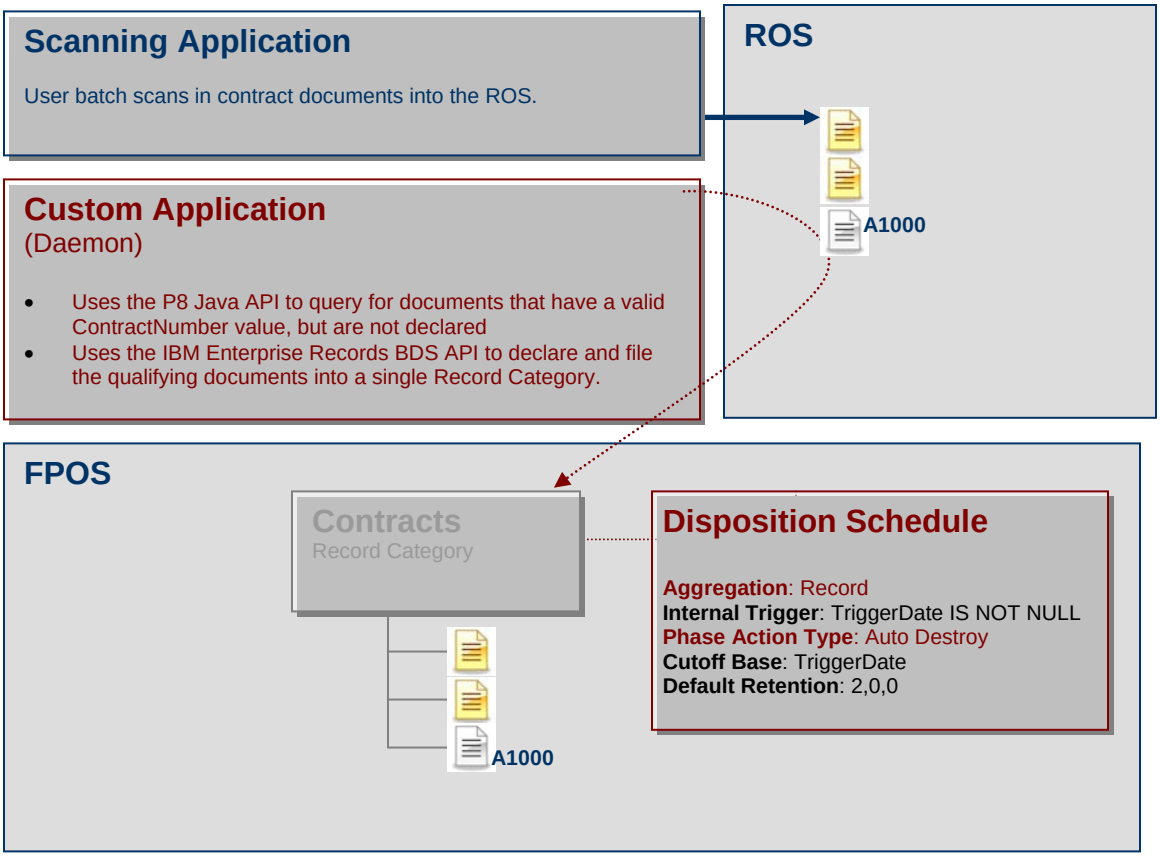
- All records that have the same disposition processing requirements are filed under a single record category. If this is not possible, then records can be filed in as few record categories as possible.
- A Disposition Schedule is applied to the record category.
- Aggregation occurs at the record level.
- There is at least one Phase where the Phase Action is set to Auto Destroy.

The Record Driven Model has the following advantages:

- When aggregating at the record level, on-hold records do not affect other records in the parent container. Other records can be destroyed that are in the same container as on-hold records. This is in contrast to container level aggregation, where a single on-hold containee prevents all other containees from being destroyed, even if they are not on hold.
- The model is simple and custom programming is not required to create Record Folders or Volumes.
- For Disposition Sweep, the Record Driven model has the disadvantage of having to process every qualifying record. This is in contrast to container level aggregation where only the parent container requires processing.

2.4.1 Record Driven Model Scenario Example

In this example, each document represents a loan contract that closes independently from other contracts. The example follows a single contract document from declaration to destruction. The contract number for this document is A1000. The disposition cycle starts when the TriggerDate property is set to a non-null value and the document is destroyed two years from that date.



In this example, all contract documents are declared into a single record category called Contracts. If there are different types of contract documents that require different disposition schedules, then each contract type needs to be filed into a distinct record category that have the appropriate disposition schedule.

After declaration, the A1000 contract document follows this disposition cycle:

- The contract is closed.
- A user or application updates the TriggerDate property on the A1000 record in the FPOS in order to reflect the contract close date.

- Disposition Sweep executes and the A1000 record qualifies to move to the first phase of its disposition.
- The phase execution date (the date when the phase is actual scheduled to execute) is set to Cutoff Base + Default Retention. For this example that is two years from the TriggerDate property value.
- After two years, Auto Destroy executes. The phase execution date on the A1000 record is less than or equal to the current date. Therefore, the phase is executed and the record is destroyed.

2.5 Delayed Aggregation Model

This model is an alternative to the Record Driven model. Using this model, a scenario that would otherwise require record level aggregation can employ the better performing container level aggregation. This model is used under the following circumstances.

- The event that qualifies the record for disposition processing occurs after declaration.
- When the event occurs and the metadata is established to qualify the record for disposition processing, the record can be grouped with other records that follow the same disposition cycle. For example, if the Cutoff Base property for all of these records is TriggerDate and many records share the same close date, then those records can then be grouped by TriggerDate.

Because the records can be grouped by a common property value when the event occurs, the records can also be moved into a container that is associated with a Disposition Schedule that is set to use container level aggregation.

Consider the cost of creating containers and moving the records against the cost of using record level aggregation. For example, if it is necessary to create several containers and they will only contain a few records, then using this model may result in higher processing costs than using the Record Driven model.

Advantages

- Records are grouped by a common retention period and retention date. As a result, processing is optimized by using container level aggregation.

Disadvantages

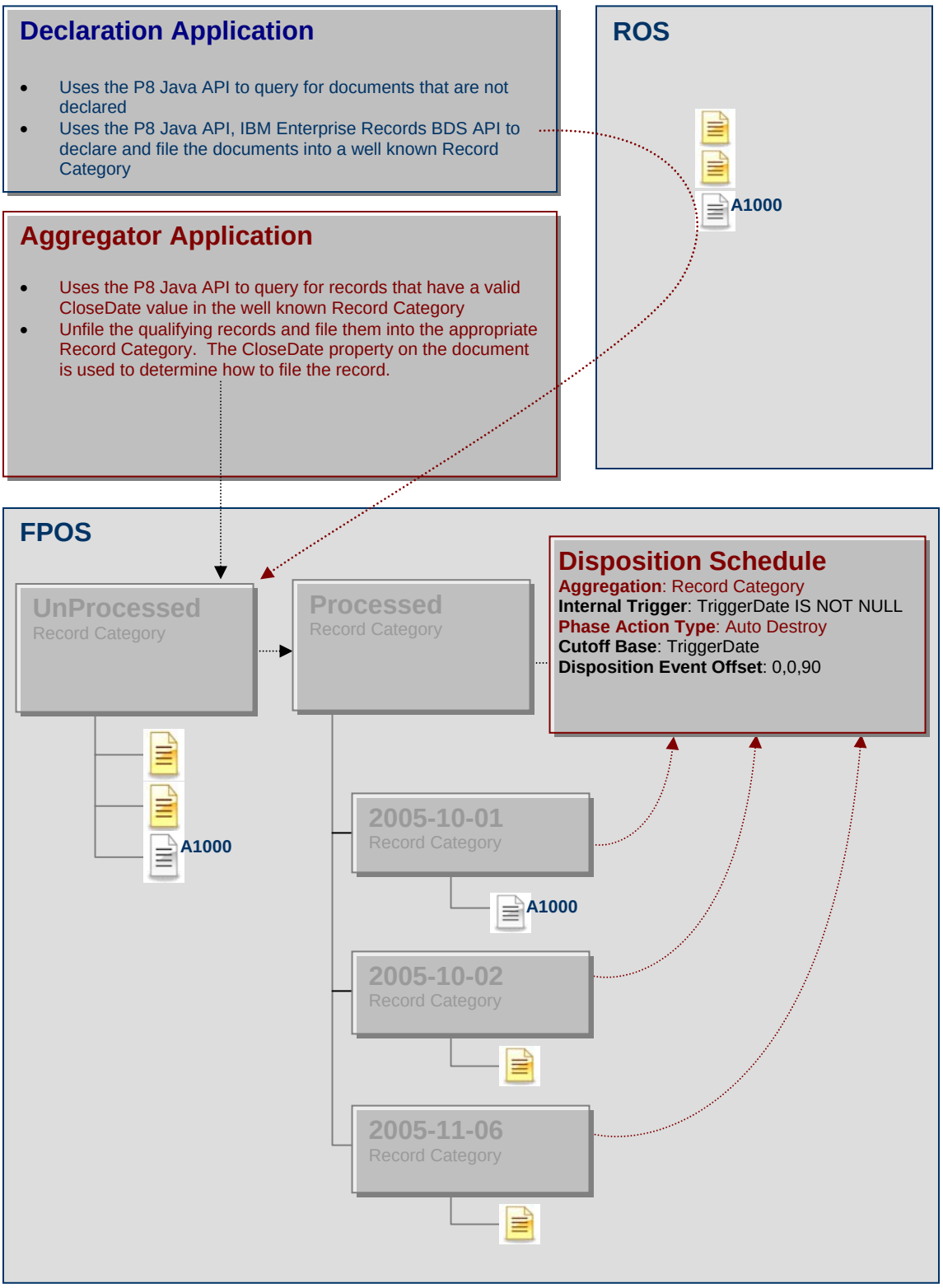
- Custom programming is required to create record category and move Records from one record category to the other record category.
- Under certain circumstances this model may require more processing than the Record Driven model.

2.5.1 Delayed Aggregation Model Scenario Example

In this example, each document represents a loan contract that closes independently from other contracts. The example follows a single contract document from declaration to destruction. The contract number for this document

is A1000. The disposition cycle starts when the TriggerDate property is set to a non-null value and the document is destroyed 90 days from that date.

The following diagram depicts the flow of declaration and filing.



The Content Manager SQL query to select qualifying documents for declaration is similar to the following:

```
SELECT
      Id
FROM
      Document
WHERE
      RecordInformation IS NULL
```

The Content Manager SQL query to select qualifying records to be grouped is similar to the following:

```
SELECT
      Id, CloseDate
FROM
      RecordInfo
WHERE
      TriggerDate IS NOT NULL AND
      INFOLDER '/UnProcessed'
```

In this example, all documents are declared into a single record category called UnProcessed. A user or application updates the TriggerDate property on A1000 record to 10-01-2005. The Aggregator Application queries all records in the UnProcessed record category that have a TriggerDate. Based on the value of TriggerDate property, the Aggregator Application moves the A1000 record into the 2005-10-01 record category that is associated with a Disposition Schedule.

The 2005-10-01 record category follows this disposition cycle.

- Disposition Sweep executes and the 2005-10-01 record category qualifies to move to the first phase of its disposition.
- The phase execution date is set to Cutoff Base + Disposition Event Offset. For this example, that is 90 days from the TriggerDate property value.
- After 90 days, Auto Destroy executes. The record category and its entire contents are destroyed.

3 File Plan Object Store

File plan organization depends on several factors including performance. This section provides information that can guide file plan design decisions to develop a well performing Records Management system.

3.1 General Principles and Recommendations

3.1.1 Scale the Content Manager

IBM Enterprise Records is a vertical application on top of Content Manager (CM). When the CM is scaled, IBM Enterprise Records scales with it. One scaling example is to create a CM farm with two or more instances of the CM server. For example, multiple declaration applications can execute against this CM farm, increasing the overall declaration throughput. This scaling concept applies to the IBM Enterprise Records Web Application and all IBM Enterprise Records stand-alone applications such as Disposition Sweep, Auto Destroy, Hold Sweep, and the File Plan Import/Export tool.

3.1.2 Ensure Proper RDBMS Statistics Sampling

Declaring the majority of RIO instances into one container can cause a database skew in the *Relationship* table. This is a general database problem that occurs when the database optimizer generates an inefficient execution plan because the statistics sampling is too small. The optimizer becomes susceptible to this problem when the uniqueness of items in a particular column is low. For example, the uniqueness of the *tail_id* column of the *Relationship* table becomes extremely low due to uneven filing into a single container. This can have a significant performance impact on the ability to browse containers that have very few RIO instances or none at all. Furthermore, this could significantly impact other queries against the *Relationship* table.

In the case where uneven filing into one container is unavoidable, ensure that database statistics are gathered at a significant sampling size for the *Relationship* table. For more information on this topic, see the *Avoid index skew* section of the P8 Performance Tuning Guide.

3.1.3 Add Appropriate Indexes

IBM Enterprise Records System Indexes

It is critical for proper performance that certain columns are indexed appropriately. IBM Enterprise Records has a number of system level columns that required indexing. There are also a number of indexes to optionally consider for indexing based on how the IBM Enterprise Records application is used.

Disposition, hold, and destroy processing rely on user defined searches. For example, Disposition Sweep is driven by a schedule which, in turn, typically uses an internal event to build a SQL statement to select entities that are ready for disposition processing. In this case, the event contains one or more properties that are evaluated according to the conditions defined by a user. In most cases, the properties used for evaluation benefit greatly from being indexed. The records administrator and the database administrator need to ensure that all properties used in these user defined conditions are properly designed for use as search keys and are properly indexed.

3.2 Using the File Plan

3.2.1 Isolate the FPOS

Use the FPOS to store only RIO instances. It is possible to use the FPOS to store non-IBM Enterprise Records application data, such as documents. However, this could cause unintended consequences including performance problems.

3.2.2 Avoid Multi-Filing RIO Instances

RIO instances are always filed into one or more file plan containers. Filing is a resource intensive operation. Multi-filing the RIO instance increases the number of filed objects and therefore increases the number of operations during declaration and destruction. This also increases the size of the *Relationship* table and could impact queries against that table. Consider using alternate retentions instead of multi-filing if possible.

3.2.3 Aggregate at the Container Level

Aggregating at the container level ensures that, during disposition processing, updates are at a minimum. As a consequence, if any record in that container is placed on hold, the entire contents of the container must be held according to the rules applied when aggregating at the container level.

Models that naturally support aggregation at the container level are the case management and email archiving models described previously.

For best performance, use Container level aggregation where possible. In some business cases, record level aggregation can not be avoided. For example, many applications use a custom record property, such as a *TriggerDate*, to indicate when a document is closed and ready for disposition. The document might be a contract that is closed when it is either canceled or fulfilled. Both of these events (canceled, fulfilled), typically, are not known when a record is created and therefore cannot be filed into containers according to a *close date*. In this situation, record level aggregation cannot be avoided. However, if there are several documents that share the same close date, such as in a case management system, then use a container to group the RIOs and use a close date on that container in order to use container level aggregation.

Both the Retention Model and the Case Model support Container level aggregation and are recommended, from a performance perspective, over using the Record Driven Model.

3.2.4 Minimize the Number of Containers with Disposition Schedules

For every Container in the FPOS that has a disposition schedule (including inherited schedules), will be processed by Disposition Sweep. When Disposition Sweep processes a container, a number of validation and search-related queries execute. Therefore, increasing the number of containers with disposition schedules can have a significant impact on Disposition Sweep performance. By minimizing the number of Containers with Disposition Schedules, processing can be minimized, which improves overall performance. However, when an Entity GUID is specified in the Disposition Sweep configuration, Disposition Sweep only processes the containers that are located below the container specified by the Entity GUID.

Both the Retention Model and the Case Model are designed to minimize the number of Containers associated with Disposition Schedules. The Taxonomy Model is not a performance-centric model. In the Taxonomy Model, organization drives the File Plan hierarchy. In such a File Plan, there might be thousands of Containers associated with Disposition Schedules (directly or indirectly through inheritance). When this occurs and Disposition Sweep is not targeted to process any particular section of the File Plan using the Entity GUID option, then there will be an impact on performance.

4 Record Object Store

4.1 Filing Documents

How the document (not the RIO) is stored has almost no functional consequence to the IBM Enterprise Records application. However, overall performance is significantly affected during ingestion and destruction if the document is filed. There is a cost to create the filing relationship and a cost to remove it. Therefore, if there is no business need or browsing requirement to file the document, then leave the document unfiled. This guidance is most applicable to the email archiving scenario. In most cases, users will not browse or even search for the archived email records and in a relatively short period of time, the record will be destroyed. In cases where the records do need to be discovered, a conditional search can be executed to find the documents directly, or the documents can be found indirectly by browsing or searching for the RIO instances managing those documents.

5 Disposition and Hold Sweep

This section documents a number of best practices when using the IBM Enterprise Records Disposition and Hold Sweep stand-alone applications. These best practices are related to how to configure the applications, the IBM

Enterprise Records file plan, and the disposition schedule, to achieve the optimal performance.

5.1 Disposition Sweep Connection Option

Disposition Sweep supports using either an http/https connection (e.g., WSI transport) or an iiop/t3/jnp connection (e.g., EJB transport). For optimal performance, it is highly recommended that the EJB transport be configured for Sweep for all disposition and auto-destroy processing. This is because the EJB transport allows Sweep to fully utilize the EJB base load balancers to perform high-volume record update and auto-destroy operations.

5.2 Entity GUID Option

Use the Entity GUID configuration option in Disposition Sweep. This limits the scope of the file plan the Disposition Sweep application has to process, thus reducing the query and processing time. This option refers to the ID of a container in the IBM Enterprise Records file plan that Disposition Sweep has to process. The container in this case could be a category, a folder or a volume. The option implicitly depends on the Object Store option being provided.

If possible, file all records with the same disposition schedule into the same category or folder. This facilitates the use of the Entity GUID option. Additionally, it can help minimize the impact on performance when many containers are assigned schedules.

5.3 Hold Names/GUIDs option

For Hold Sweep, use the Hold Names/GUIDS option to limit the number of dynamic holds that the Hold Sweep application has to process. This includes putting entities on hold and removing holds.

Specify more than one Hold ID or Hold Name by separating them with a vertical bar.

5.4 Thread Count Option

For both Sweep applications, use the thread count option to take advantage of the CPU resources in the P8 environment. As a general rule, the number of threads should be set to the number of logical CPUs available on the CM server. Internal performance testing shows this option significantly improves the processing rate of both Sweep applications.

5.5 Disposition Schedule

The IBM Enterprise Records Disposition Schedule determines the amount of work that Disposition Sweep needs to do. A schedule can be assigned to a folder, category or a record type. This section introduces scenarios that improve Disposition Sweep's performance based on where the user assigns a disposal

schedule. In addition, some basic rules on disposal scheduling precedence are discussed.

5.5.1 Records Filing

File records that have the same disposition period in the same folder or category. This not only facilitates the Entity GUID option but also allows these records to be controlled by one schedule.

File records that have no disposition requirement, and that remain as permanent records, under a container or group of containers that have no associated schedule.

5.5.2 Setting Aggregation Level of Internal Event

Generally, set the aggregation level (of the internal event) to the highest entity possible in the IBM Enterprise Records inheritance hierarchy. The goal is to ensure that Disposition Sweep updates as few entities as possible.

For example, if there are 100 records that have the same retention period, then file them under the same folder associated with a schedule whose internal event has folder level aggregation. With this setup, Disposition Sweep only has to update and manage disposition for the folder. Otherwise, if the schedule has an internal event with record level aggregation, then Disposition Sweep has to update and manage disposition at the record level, which, in this case, has 100 records.

5.5.3 Ineffective Schedule

When selecting an aggregation level for the internal event, the schedule can become ineffective if the selected aggregation level does not match where the schedule is assigned. For example, if a schedule's internal event has category level aggregation and the schedule is assigned to a folder, then the schedule is ineffective, which means that all entities filed under the folder and the folder itself will not be processed for disposition. Disposition Sweep still has to look at the schedule and its aggregation, but nothing is processed. To be effective, the schedule either has to be assigned to a category, or its aggregation has to be changed to a value other than category level.

The table below indicates the relevancy of the schedule aggregation when assigned to a category, folder or record type.

The "YES" text means the schedule is effective and the "NO" means ineffective schedule:

Aggregation Level	Schedule assign to Category container	Schedule assign to Folder container	Schedule assign to Record Type
Category	YES	NO	NO
Folder	NO	YES	NO
Volume	NO	YES (to all volume under the folder, i.e., in-folder search)	NO
Record	YES (only immediate children, i.e., in-folder search only)	YES (to all children under all volume, i.e., in-subfolder search)	YES

6 Record Declaration and Destruction

6.1 Options for Declaring Records

6.1.1 BDS API Custom Application

A custom application using the BDS API has the best performance for declaring records compared to using IBM Enterprise Records API or the other methods mentioned below. The declaration performance is affected if it runs with other applications, such as Disposition Sweep, Hold Sweep, and Auto Destroy. The best practice is to create a schedule to run document creation, record declaration, hold processing and record disposition, in separate time slots.

However, in this approach, there is a delay between the time that records are created and the time that there are declared. For some business cases, this leaves documents unsecured and is not acceptable. In that case, the other declaration options may be more suitable.

6.1.2 Synchronous BDS API Declaration Event

The performance of a synchronous event that uses the BDS API to declare records is not as good as a custom application that uses the BDS API. The synchronous event method has almost the same performance as an asynchronous event that uses the BDS API to declare records. A synchronous

declare action event runs in the same transaction as the originating activity on the target object.

If an event is used to declare records, employ the synchronous mode if possible.

6.1.3 Asynchronous BDS API Declaration Event

The performance of an asynchronous event that uses the BDS API to declare records is not as good as a synchronous event that uses the BDS API. An asynchronous event allows the subscription processor to continue without waiting for the results of the action, but the action will not be in the same transaction as the originating activity.

If an event is used to declare records, employ the synchronous mode if possible.

6.1.4 BDS API Declaration Event Subscription with Workflow

The BDS API can be used to declare records following some specific workflow, but overall declaration performance will not be optimal.

6.2 BDS Guidance

6.2.1 Restricting Batch Size

During each BDS API batch execution, all operations in the batch are sent as a single payload to the CM and are processed there as a single transaction. As the number of operations within a single batch increases, the likelihood of a CM transaction time-out (or other resource-limitation error) also increases. It is not possible to predefine the upper limit or optimum value for the number of operations per batch because that depends upon many application and site-specific characteristics such as:

- Using the *createDocument* and *createDocumentAndDeclareRecord* operations require more resources than simply declaring records.
- Performance and load characteristics of the CM and its database server.
- Network performance.

One might first assume that the largest number of operations per batch guarantees the best declaration performance. This is not necessarily true because larger transaction sizes might not scale accordingly on the CM due to resource issues, or there might be detrimental effects on other P8 system operations.

A best practice is to design your application such that the BDS API batch size is a tunable parameter. You can then adjust this value in the actual production environment for best overall effect. As a general rule, a batch size of 25 or less operations per batch is a typical value.

6.2.2 Scaling Guidance

To utilize the multi-core CPU servers typically used in enterprise environments, design BDS client applications to use multiple threads or to allow multiple application processes to be simultaneously executed. In this manner, multiple CM transactions can simultaneously take advantage of server resources for better overall system throughput. Such scenarios allow smaller BDS batch sizes to be used and reduce the likelihood of any transaction timeout while still gaining improved overall system performance and utilization.

For typical BDS applications that perform multiple batch executions, reuse *BulkDeclarationService* instances whenever possible. A transport connection initialization cost is incurred whenever a new *BulkDeclarationService* instance first communicates with the CM. Ideally, reuse each such *BulkDeclarationService* instance on the same thread on which it was first created. NEVER share a given *BulkDeclarationService* instance simultaneously across multiple threads.

6.2.3 Using BDS in Federated Environment

By default, when calling *BulkDeclarationFactory* to get *BulkDeclarationService* instance by passing the *contextInfo*, the value *BDSConstants.TRANSPORT_TYPE_BDP40_WSI* is set to the key *BDSConstants.CONTEXT_TRANSPORT_TYPE* in the *contextInfo*. But when records are declared for documents that reside in a P8 Content Federated Services (CFS) environment, the value *BDSConstants.TRANSPORT_TYPE_BDP40_JACE* must be set to the key *BDSConstants.CONTEXT_TRANSPORT_TYPE* in the *contextInfo*.

For situations in which the *BDSConstants.TRANSPORT_TYPE_BDP40_JACE* is used, establish the CM Java API authentication *UserContext* for each thread within the client application and do not supply any username/password credentials in the BDS API context Map. This ensures that a *UserContext* is only created once per thread rather than for every BDS API batch execution.

6.2.4 Transport Type Selection

When you choose from *BDSConstants.TRANSPORT_TYPE_BDP40_WSI, BDSConstants.TRANSPORT_TYPE_BDP40_JACE* with WSI transport and with EJB transport, consider the following factors:

- TRANSPORT_TYPE_BDP40_JACE uses the CE Java API, which supports both EJB and HTTP protocol (through its own WSI API implementation). The EJB protocol provides the best overall performance and works with EJB based load balancers.
- TRANSPORT_TYPE_BDP40_WSI uses the WSI API, which supports the HTTP protocol and provides the ability to use HTTP based load balancers.
- Network topology / firewall constraints limit the communications to the Web Services protocol.

- Overall architecture of the application environment directs the use of one protocol over another.
- Heterogeneous application servers are being used as the front ends.

6.2.5 P8 Containment Naming

Whenever a record or document object is filed into a P8 Object Store container, a containment name must be established that is unique relative to any existing siblings within the container. The CM automatically generates a unique containment name.

Because documents can be filed to a folder and RIOs are filed into some container in the file plan, this automatic containment name generation can result in reduced document creation or record declaration performance due to the extra checks performed during the filing task. To reduce this overhead, make the "*DocumentTitle*" property value assigned to the new document/RIO unique within the object store (ROS and/or FPOS).

Starting with the IBM InfoSphere Enterprise Records V4.5.1 release, to provide another method to control the containment name, the BDS *ContainerReference* interface provides getter/setter methods for two new fields:

- *boolean autoUniqueNaming* – Controls the CM automatic containment name generation process described above. Is set to true by default for backwards compatibility.
- *String containmentName* – Allows the BDS client to specify a containment name independent of the object's special "name" property. The default value of this field is null for backwards compatibility.

For best BDS API filing performance, the BDS client sets the *autoUniqueNaming* field to false and provides a unique string value for the *containmentName* field value. This applies to both document creation and record declaration.

6.3 Destroying Records

6.3.1 Record Destruction using BDS

In addition to declaring records, BDS can be used to dispose or destroy records. When compared to IBM Enterprise Records API, destroying records through a BDS custom application yields the best performance. Similar to declaring records, the records destroy performance is affected if it runs with other applications, like Disposition Sweep, Hold Sweep, and record declaration. The best practice is to create a schedule to run records disposal in a separate time period.

Thread Count Option

When using the BDS API to perform record disposal, a thread count option can be specified to improve the performance of the disposal. Optimally, the number of threads are equal to the number of the logical CPU at the CM server.

This option is only relevant to the record disposal interface in the BDS and it does not apply to the record declaration part of the interface.

6.3.2 Record Destruction using the Auto-Destroy Action

This action should be used when destroying records through a disposition schedule. The action can be specified in a phase of the schedule which can then be assigned to a category or folder to be processed by the Disposition Sweep tool. By specifying the Autodelete action in a schedule, When the schedule retention period has expired, Sweep automatically disposes the records without requiring any manual initiation of disposition from the IBM Enterprise Records user interface.

6.3.2.1 Allow Auto-Destroy On Non-On-hold Containees Option:

This was a new option in IBM InfoSphere Enterprise Records V4.5.1 that is currently only available in the Auto-Destroy action. For optimal performance, users can specify this option in the Auto-destroy action so that it removes an IBM Enterprise Records restriction of performing an auto-destroy on a container (e.g., category or folder), and the application not completing the auto-destroy if there is any on-hold containees in the container. By selecting this option in the auto-destroy action, it allows IBM Enterprise Records and Disposition Sweep tool to continue doing auto-destroy on other non on-hold containees. In addition, this option allows Disposition Sweep to avoid doing any on-hold validation that could impact the performance.

6.3.2.2 Non-Page Query Max Size Option:

For best performance, the Disposition Sweep tool uses a non-page query during auto-destroy processing. This means that the query depends on the CE option called Non-Page Query Max Size to determine the number of IBM Enterprise Records entities to be retrieved per batch. By default, the option value is 5000 and it can be configured at the CE Enterprise Manager to a higher value for improved performance.

7 Troubleshooting IBM Enterprise Records Performance

When performance issues arise, review the IBM Enterprise Records documentation for possible solutions and for instructions on how to gather the necessary information if IBM support is required. Also, review the CM documentation such as the P8 Performance Tuning Guide for additional guidance.

Links to these useful documents can be found in the References section below.

8 CM SQL Paging Guidance

When customizing your own application using the IBM FileNet Content API, it may be necessary to page results sets in order to prevent both the client and server from consuming too much memory. The IBM FileNet Content API provides a paging mechanism. The following should be observed when using the paging mechanism.

1. Discover the paging query using an RDBMS monitoring tool.
2. Create an index that that optimizes the paging query. Ensure that the columns used in the ORDER BY clause are appended as the last items in the index. Ensure that these columns in the index are appended in the same ordinal order and sort order as the ORDER BY clause.
3. Once the index has been created, run the paging query and use an RDBMS tool to analyze the execution plan. Ensure that sorting has been avoided (ignore RID sorting).
4. Avoid query joins. It is not possible to create a cross-table index and therefore creating an index that avoids sorting is extremely difficult.

9 RDBMS Tablespace Recommendation

Under high volume processing, it is typical that a number Content Manager tables will be active. For example the DocVersion and Relationship table will be active during high volume declaration and destruction. It is recommended that the active tables be placed in their own tablespaces. The tablespaces should be associated with independent volumes. This will reduce contention and increase parallel I/O.

10 DB2 Reorganization Recommendations

Reorganize DB2 tables and indexes when they become significantly large.

10.1 Autodestroy Reorganization Guidance

When the FPOS Relationship table becomes significantly large and when executing Autodestroy, reorganize the FPOS Relationship table in this manner:

1. Reorganize the I_Relationship27 index in the FPOS.
2. Reorganize the Relationship table in the FPOS according to the I_Relationship27 index.

11 References

P8 4.x Performance Tuning Guide
ftp://ftp.software.ibm.com/software/data/cm/filenet/docs/p8doc/40x/p8_400_performance_tuning.pdf

P8 5.0 Performance Tuning Guide
ftp://ftp.software.ibm.com/software/data/cm/filenet/docs/p8doc/50x/p850_performance_tuning.pdf

P8 5.1 Performance Tuning Guide
http://pic.dhe.ibm.com/infocenter/p8docs/v5r1m0/topic/com.ibm.p8.performance.doc/p8ppt000.htm

MustGather: FileNet Content Engine Database
http://www-01.ibm.com/support/docview.wss?rs=3278&context=SSNVNV&context=SSTHRT&context=SSNVUD&context=SSS236&context=SSNW2F&q1=mustgather&uid=swg21313976&loc=en_US&cs=utf-8&lang=en

MustGather: Read First for IBM Enterprise Records
http://www-01.ibm.com/support/docview.wss?rs=3278&context=SSCPNJV&uid=swg21323065&loc=en_US&cs=utf-8&lang=en

DB2 9 for z/OS Performance Topics
http://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/sg247473.html?Open

IBM FileNet P8 Information Centers
http://pic.dhe.ibm.com/infocenter/p8docs/v4r5m1/index.jsp
http://pic.dhe.ibm.com/infocenter/p8docs/v5r0m0/index.jsp
http://pic.dhe.ibm.com/infocenter/p8docs/v5r1m0/index.jsp

Product Documentation for IBM Enterprise Records
http://www.ibm.com/support/docview.wss?rs=3286&context=SSNVVQ&uid=swg27010387

Hardware and software requirements for IBM FileNet P8 Versions 3.5, 4.0, 4.5, 4.5.1, 5.0, and 5.1
http://www.ibm.com/support/docview.wss?rs=3278&uid=swg27013654