Contents

[1. FileNet P8 Architecture 3](#_Toc167524005)

[1.1 What we will cover in this training 4](#_Toc167524006)

[2. Content Platform Engine 4](#_Toc167524007)

[2.1 Walkthrough of Administration Console for Content Platform Engine 4](#_Toc167524008)

[2.1.1 Domain 4](#_Toc167524009)

[2.1.2 Object Store 4](#_Toc167524010)

[2.1.3 Exploring Classes 4](#_Toc167524011)

[2.2 Security 4](#_Toc167524012)

[2.2.1 ACE/ACL (Access Control Entry/Access Control List) 4](#_Toc167524013)

[2.2.3 Role – based access control (RBAC) 5](#_Toc167524014)

[2.2.4 Security Inheritance (Security Proxy) 5](#_Toc167524015)

[2.2.5 Marking Sets 5](#_Toc167524016)

[2.2.6 Security Policies 5](#_Toc167524017)

[2.3 Document Class 5](#_Toc167524018)

[2.3.1 Creation of Property Templates 5](#_Toc167524019)

[2.3.2 Creation of Document Class 5](#_Toc167524020)

[2.3.3 Adding Property Definitions to the Document Class 5](#_Toc167524021)

[2.3.4 Checking the Security 5](#_Toc167524022)

[2.3.5 Creating a Document Instance 5](#_Toc167524023)

[2.3.5 Versioning of Documents 5](#_Toc167524024)

[2.3.6 Implementing Security Proxy 5](#_Toc167524025)

[2.3.7 Implementing Marking Sets 5](#_Toc167524026)

[2.3.8 Implementing RBAC 7](#_Toc167524027)

[2.3.9 Implementing Security Policy 10](#_Toc167524028)

[2.4 Events and Subscriptions 10](#_Toc167524029)

[2.4.1. Introduction 10](#_Toc167524030)

[2.4.2 Creating the Code Module 10](#_Toc167524031)

[2.4.3 Creating the Event Action 10](#_Toc167524032)

[2.4.4 Creating the Subscription 10](#_Toc167524033)

[2.4.5 Testing the Event Action 10](#_Toc167524034)

[2.5 Search Objects in Content Engine 11](#_Toc167524035)

[2.5.1 How to do Object Store Search 11](#_Toc167524036)

[2.5.2 How to do Bulk actions 11](#_Toc167524037)

[2.6 Audit History 11](#_Toc167524038)

[2.6.1 Introduction 11](#_Toc167524039)

[2.6.2 How to Configure and view Audit History in FileNet 11](#_Toc167524040)

[2.7 Sweep Jobs 11](#_Toc167524041)

[2.7.1 Background Search Sweep 11](#_Toc167524042)

[2.8 Content Engine Java API 13](#_Toc167524043)

[2.8.1 Setting up Eclipse 13](#_Toc167524044)

[2.8.2 Creating a FileNet Connection 13](#_Toc167524045)

[2.8.3 Working with Documents 13](#_Toc167524046)

[3. IBM Content Navigator 13](#_Toc167524047)

[3.1 Introduction to Navigator 13](#_Toc167524048)

[3.2 Walkthrough to Navigator Admin desktop 13](#_Toc167524049)

[3.3. Development in Navigator 13](#_Toc167524050)

[3.3.1 Creation of Object Store repositories 13](#_Toc167524051)

[3.3.2 Creation of Desktops 13](#_Toc167524052)

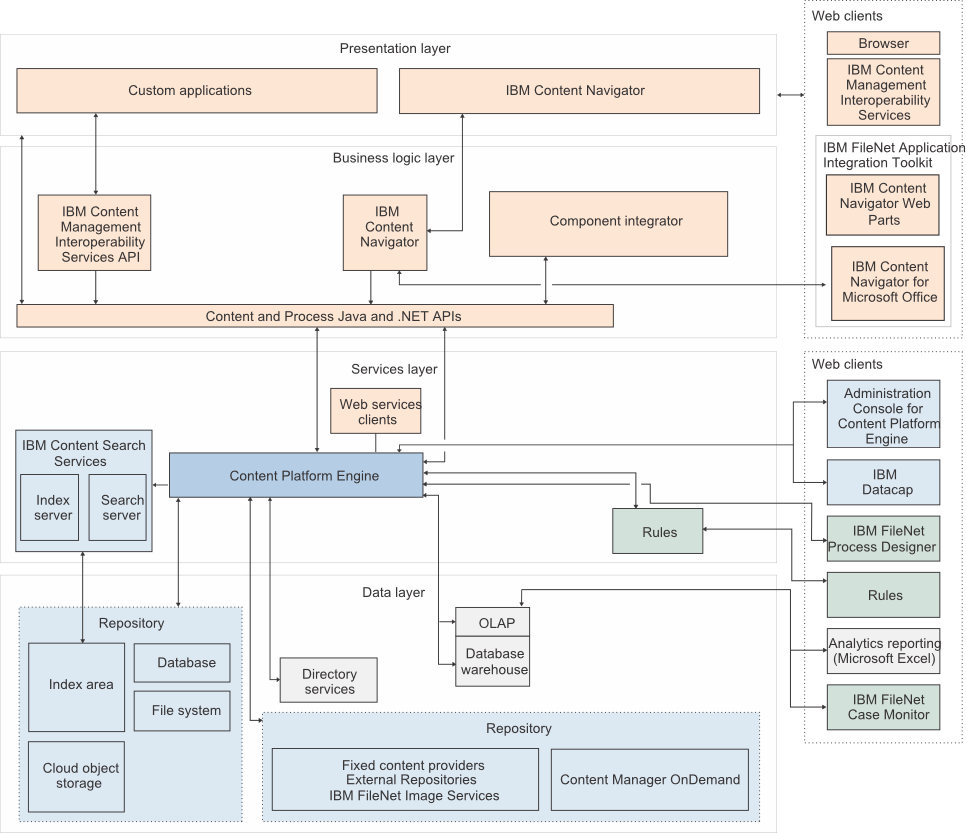
[3.4 Navigator Customizations 13](#_Toc167524053)

[4. IBM Case Manager 13](#_Toc167524054)

IBM FileNet Training

# FileNet P8 Architecture

The FileNet P8 family of products includes back-end services, development tools, and applications that address enterprise content and process management requirements



## What we will cover in this training

The below components we will cover as part of this training

* + 1. **Content Platform Engine (CPE)**
    2. **IBM Navigator**
    3. **IBM Case Manager**

# Content Platform Engine

## 2.1 Walkthrough of Administration Console for Content Platform Engine

### 2.1.1 Domain

### 2.1.2 Object Store

### 2.1.3 Exploring Classes

## 2.2 Security

The Content Engine resides within a Java Platform, Enterprise Edition (Java EE) application server, and uses the Java Authentication and Authorization (JAAS) standard as the basis for authentication. The JAAS programming model is a standard Java framework that manages authentication and authorization.

### 2.2.1 ACE/ACL (Access Control Entry/Access Control List)

**ACE Source:** Every ACE has a source, either Default,Direct, Inherited or Template.

* + 1. **Default:** Permissions are placed on an object by the Default instance security ACL of its class, as well as permissions placed on a sub classs by its parent class. Default ACEs are directly editable. If you do, its source type becomes Direct.
    2. **Direct:** Permissions are added directly to an object. Direct ACEs are directly editable
    3. **Inherited:** Permissions are placed on the object by a security parent or by setting up a relationship with an object-value property whose security proxy type has been set to Inherited. Inherited ACEs are not directly editable.

Example: ***Security Proxy Concept***

* + 1. **Template:** Permissions are placed on the object by a security policy. Template ACEs are not directly editable and do not appear on classes. Rather, a document, folder, or custom object class can have a default security policy which will pass template ACEs to the instances of the class, if all the conditions for the template apply.

Example**:** ***Security Policy Concept***

### 2.2.3 Role – based access control (RBAC)

You can create roles in your object store as another way to grant access to objects. Role-based access depends on three facets of authorization

* What access does the role grant?
* Who, meaning which users or groups, are the members of this role?
* Which objects can be accessed by members of this role

Using roles can simply security in some cases because you can update the role parameters, such a members or what kind of access the role has, without having to update all the objects to which members of the role have access.

### 2.2.4 Security Inheritance (Security Proxy)

### 2.2.5 Marking Sets

### 2.2.6 Security Policies

## 2.3 Document Class

### 2.3.1 Creation of Property Templates

### 2.3.2 Creation of Document Class

### 2.3.3 Adding Property Definitions to the Document Class

### 2.3.4 Checking the Security

### 2.3.5 Creating a Document Instance

### 2.3.5 Versioning of Documents

### 2.3.6 Implementing Security Proxy

### 2.3.7 Implementing Marking Sets

Marking security consists of the Add marking, Remove marking, and Use Marked Objects.

**Add marking and Remove marking**

A user with Add rights to a marking can set the property value associated with the marking, if it has not been set. Only those markings to which the user has Add rights will show in the list of marking values ​​available to be set in a property. A user with Remove rights to a marking can remove the marking value.

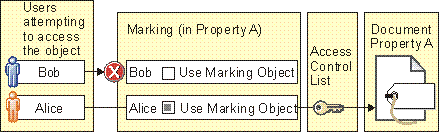
For example:

1. A Document has a property associated with a marking set. No value has been specified for the property.
2. The marking set has markings Red, Blue, and Green.
3. Alice has Use rights to Red; Use and Add rights to Blue and Use, Add and Remove rights to Green.

When Alice views the Document properties, she can set the property value to Blue or Green but not Red. If the property was set to Green, she could alter it to be Blue. If the property was set to Blue, Alice would be unable to alter the property's value.

**Use Marked Objects**

Use right determines whether the presence of the marking on an object constrains access to that object. If the user has Use right to the marking, access to the object will not be constrained.



In this example, Alice has the Use Marked Objects access right which lets her bypass the marking. Her access to the object will be evaluated by the object's ACL. Bob does not have Use Marked Objects and therefore will neither see nor have access to the object, regardless of any permissions the object's ACL might grant him.

Markings and marking sets are Content Platform Engine objects, each with a class description:

* Markings are objects that combine metadata behavior with access control behavior in a way that allows an object's access control to change by changing a property value.
* Marking sets have containers for markings. Marking sets are associated with a Property Template which can then be used to add a property to one or more classes.

Implementation:

Lets assume we have country property on a Covid Guidelines document class, and assume

We have 2 countries

* + 1. India ( intuser1 is the user belongs to india country )
    2. USA ( lucy is the user belongs to USA )

We have one admin user p8 admin:

Scenario: Now whatever Covid Guide lines document created for which country depending on the those country specific users only should be able to see that document.

Lets Achieve this using Marking set

1. Create a Marking Set named “Country Marking”
2. Give two marking india and usa
3. Add the appropriate users to the specific markings and save the marking
4. Create a property template named “Country “ and map the marking set
5. Assign the property template to Covid Guidelines document class
6. Create an instance and test the Marking set security

### 2.3.8 Implementing RBAC

You use the Administration Console for Content Platform Engine to create and assign security roles for you content objects.

When you use role-based access, you create and configure role classes and subclasses, then create instances of the roles. You can associate these role instances with the objects that you want to secure.

Use the following steps to configure role-based access:

1. [**Creating a role class**](https://www.ibm.com/docs/en/SSNW2F_5.5.0/com.ibm.p8.security.doc/p8psh028.htm)  
   You use the Administration Console for Content Platform Engine to create role classes.

Roles are created as subclasses under the Role class. You choose whether to create a subclass of the Static Role class or the Dynamic role class:

**Static Role**

In a static role, users and groups are assigned directly to a role. This method is similar to how ACLs are assigned to an object.

**Dynamic Role**

In a dynamic role, external code is used to determine if a user if a member of a role. This method allows for more dynamic role assignments based on application use cases. The external code returns a yes or no response as to whether the specified user is a member of the dynamic role.

**Procedure**

To create a role class:

1. In Administration Console for Content Platform Engine, expand **Data Design** > **Classes** > **Other Classes** > **Role**.
2. Right click the type of role class that you want to create, **Dynamic** or **Static**, and click **New Class**.
   1. Role classes are hierarchical, which means that you can create new roles under existing roles. If you are creating the role class to add to an existing set of roles, choose the appropriate subclass as the parent for your new class.
3. Enter a name and description for the new role.
4. Click **Finish**, then click **Open** to refine your role definition.
   1. Add or remove access definitions by choosing a class and choosing which types of access the role has for that class.
   2. **Tip:** If the tab for defining access definitions does not display, refresh the view.
5. Save your changes.
6. [**Creating a role instance**](https://www.ibm.com/docs/en/SSNW2F_5.5.0/com.ibm.p8.security.doc/p8psh029.htm)  
   After you create the role class, you use the Administration Console for Content Platform Engine to create a role instance for the object store.

You use an instance of a security role to grant the access defined by the role to a specific set of users and groups. You can create an instance of your security role in the Data Design section of the Administration Console for Content Platform Engine.

**Procedure**

To create a role instance:

1. In Administration Console for Content Platform Engine, navigate to the object store where you want to create the instance, expand the object store, and click **Roles**.
2. Right click the type of role instance that you want to create, **Dynamic** or **Static**, and click **New Role**.
3. Enter a value for the **Display name**, and choose the role class that is the basis for your new role instance.
4. On the **Role Members** tab, add role members.

You can add users and groups directly, add a realm, or add a nested role.

1. Click **Finish** to save your role instance.

**Results**

The created role instance:

* Includes the users and groups that the role class contains, plus any that were added specifically to the instance
* Grants the permissions to the specified classes (for example, Documents) that you configured on the role class
* Grants additional permissions that you specified for the instance. You can use the **Role Access Definitions** tab to specify additional permissions (access rights). For more information, see topic [About access rights](https://www.ibm.com/docs/en/SSNW2F_5.5.0/com.ibm.p8.security.doc/p8psa050.htm).

1. [**Associating a role instance with an object**](https://www.ibm.com/docs/en/SSNW2F_5.5.0/com.ibm.p8.security.doc/p8psh030.htm)  
   After you create a role instance, you assign this role instance to an object to secure the object with role-based access control.
2. [**Creating a dynamic role**](https://www.ibm.com/docs/en/SSNW2F_5.5.0/com.ibm.p8.security.doc/p8psh031.htm)  
   A dynamic role enables custom code to make the determination of whether a given user is to be considered as a member of the role, versus that determination being made solely on the basis of a stored list of members. The custom code might, for example, consult an external database to answer the question of membership.

In this procedure, you create and compile a java class. You can use an editor and your Java SDK to create and compile the class. The rest of the steps for creating the code module, the role membership action, the dynamic role class, and the dynamic role, are completed in the Administration Console for Content Platform Engine.

**Procedure**

1. Create a java file that describes the dynamic role:

package com.sampleclass;

import com.filenet.api.admin.CmRoleMembershipAction;

import com.filenet.api.engine.AuthorizationServices;

import com.filenet.api.engine.RoleMembershipHandler;

import com.filenet.api.security.CmDynamicRole;

import com.filenet.api.security.User;

public class RoleMembershipHandlerImpl implements RoleMembershipHandler {

@Override

public boolean isUserInRole(CmRoleMembershipAction arg0,

CmDynamicRole arg1, User arg2, AuthorizationServices arg3) {

System.out.println("in RoleMembershipHandlerImpl.isUserInRole()");

return false;

}

}

1. Use your Java SDK to compile the file into a Java class, for example, com.sampleclass.RoleMembershipHandlerImpl.class.
2. Save the Java class file to a location that can be accessed by your Content Platform Engine server.
3. In the Administration Console for Content Platform Engine, open the object store where you want to add the Dynamic Role.
4. Navigate to **Browse** > **Root folder**, and right-click **Code Modules**.
5. Click **Actions**, and click **New Document**.
6. On the **Define New Document Object** panel, provide a title, click **Select Class as Code Module**, and click **Next**.
7. Click **Add**, browse to the Java class file that you created, and click **Add Content Element**.
8. Click **Next** to accept the default values in the rest of the wizard, then click **Finish**.
9. In the Object Store view, navigate to and expand **Events, Actions, Processes**, then right-click **Role Membership Actions** and click **New**.
10. On the **New Role Membership Action** screen, click **Class**, and for the **Java class handler value**, enter the name of your newly created Java class, for example, com.sampleclass.RoleMembershipHandlerImpl.class.
11. Click **Configure code module**, then click **Next**.
12. Click **Load Existing**, select the code module that you just created, and click **OK**.
13. Click **Next** and **Finish**.
14. In the Object Store view, navigate to **Data Design** > **Classes** > **Other Classes** > **Role** > **Dynamic Role**.
15. Right-click **Dynamic Role** and click **New**.
16. Provide a name and other details for your new Dynamic Role class.
17. Select the **Role Membership Action** that you just created.
18. Click **Next** and **Finish**.
19. In the Object Store view, navigate to **Roles** > **Dynamic Roles**.
20. Right-click **Dynamic Roles**, click New, and create the dynamic role by using the class that you just created.

### 2.3.9 Implementing Security Policy

#### 2.3.9.1 Versioning Security Templates

## 2.4 Events and Subscriptions

### 2.4.1. Introduction

### 2.4.2 Creating the Code Module

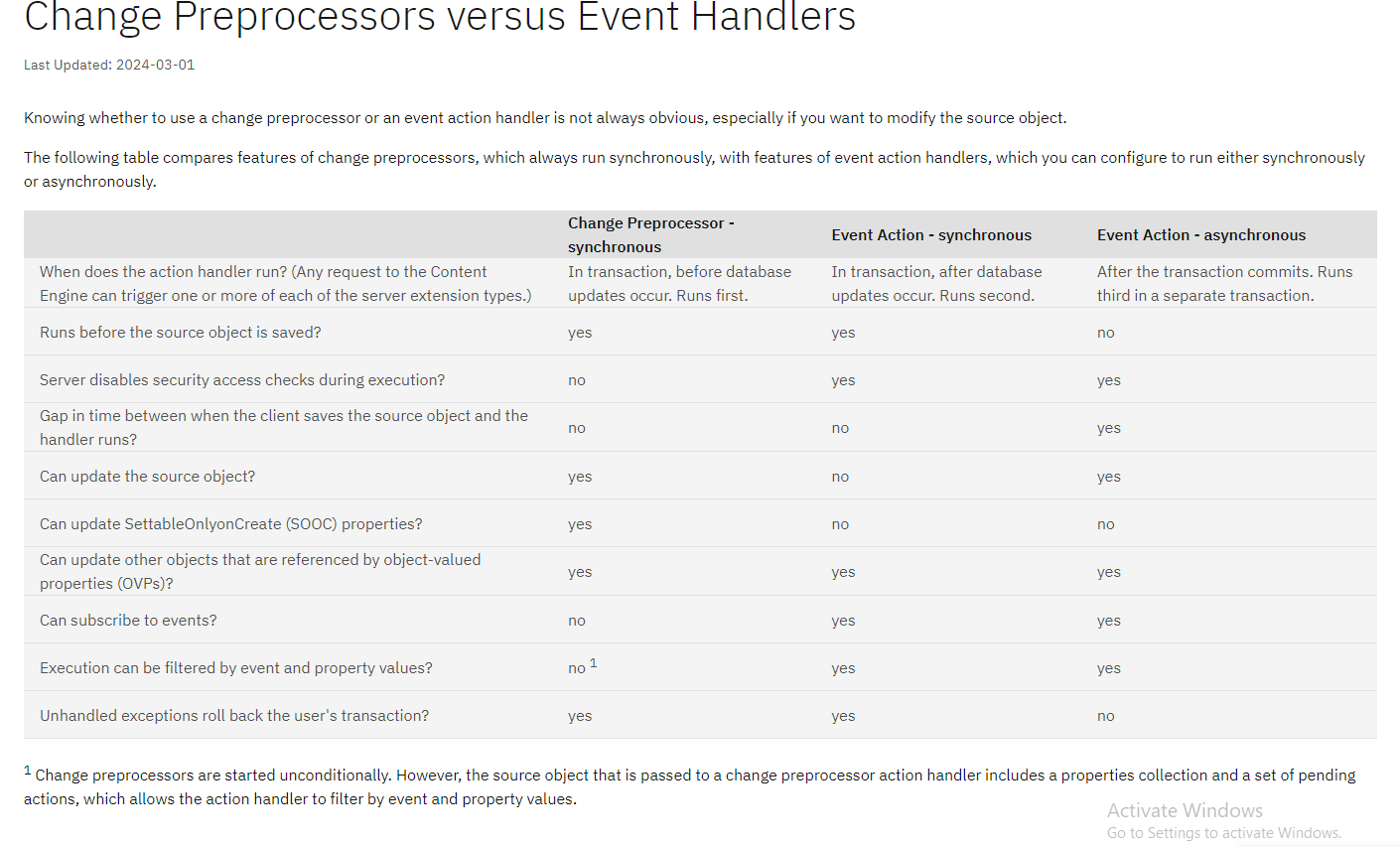
### 2.4.3 Creating the Event Action

### 2.4.4 Creating the Subscription

### 2.4.5 Testing the Event Action

### 2.4.6 Change Pre Processor

### 2.4.7 Difference Between Change Pre-processor and Event Action



## 2.5 Search Objects in Content Engine

### 2.5.1 How to do Object Store Search

### 2.5.2 How to do Bulk actions

## 2.6 Audit History

### 2.6.1 Introduction

Audit History is used to store the event information of the Document, Folders and objects in FileNet P8. With Audit History, one can view historical changes to documents in a FileNet Content Manager repository.

* In IBM ACCE, Audit history is for all types of items like documents, folders and objects.
* In IBM Content Navigator, Audit history is for document only ( can see the document related audit history )

### 2.6.2 How to Configure and view Audit History in FileNet

**In ACCE:**

1. First, need to make the property “Enable Audit History to Yes on the Object Store

2. Go to the respective document class, where you want to enable the audit history tab and then create an Audit definition by choosing what are all the events you want to audit and save it.

3. Then create a document and trigger the respective events which you have configured and then go that document and go the Audit History tab and see whether it is updated or not

**In IBM Content Navigator:**

1. Go to the respective object store repository and enable the “Document History” setting
2. Do the same in Desktop level as well
3. Then go to respective document in the navigator and open properties and see the history tab

## 2.7 Sweep Jobs

### 2.7.1 Background Search Sweep

To create a background search query, you create a template and a template-based sweep. Multiple sweeps can be created from the same template.

**Before you begin**

If you want the search results of your query to be downloadable as a CSV file, install the Reporting Enablement Extensions add-on for the object store. For information about installing add-ons, see [Installing an add-on feature to an object store](https://www.ibm.com/docs/en/SSNW2F_5.5.0/com.ibm.p8.ce.admin.tasks.doc/featureaddons/fa_install_addon.htm#fa_install_addon). For information about other ways that search results can be exported to a CSV file, see [Search result processing](https://www.ibm.com/docs/en/SSNW2F_5.5.0/com.ibm.p8.ce.admin.tasks.doc/p8pcc381.htm#p8pcc381).

**About this task**

The template primarily consists of three interrelated elements:

* The search expression
* Parameter properties
* Search result properties

You specify parameter values when the sweep is created from the template. Also, when you create the sweep, you can limit the number of search results for query testing purposes.

**Important:** Some configuration such as creating database indexes might be necessary to prevent your background search queries from running slowly. For more information, see [Tuning Content Platform Engine background searches](https://www.ibm.com/docs/en/SSNW2F_5.5.0/com.ibm.p8.performance.doc/p8ppt030.htm).

**Procedure**

To create a background search query:

1. Start the New Background Search Class Template wizard in the administration console:
   1. In the domain navigation pane, select the object store.
   2. In the object store navigation pane, select the **Data Design** > **Background Search Class Templates** folder.
   3. On the Background Search Class Templates tab, click **New**.
2. Complete the wizard:
   1. Enter a search expression.
   2. As necessary, edit the attributes of the custom property templates that the wizard creates automatically for parameters and search results.
   3. Optional: Select the **Generate a CSV file** option. The wizard displays this option only if the Reporting Enablement Extensions add-on is installed.
3. [Generating search results with a background search sweep](https://www.ibm.com/docs/en/SSNW2F_5.5.0/com.ibm.p8.ce.admin.tasks.doc/p8pcc382.htm#p8pcc382). Create a template-based sweep.

Example: let us consider we have a Person Class with some properties

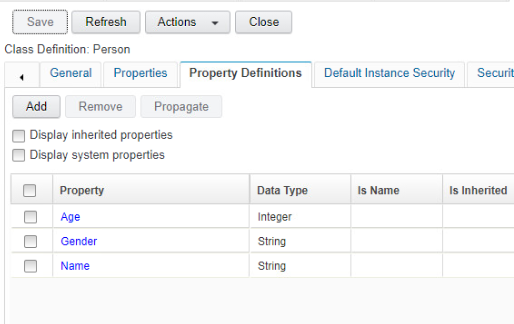
|  |  |
| --- | --- |
| **Document Class Name** | Person |
| **Properties** | Name, Age, Gender, Location |

Scenario: Now for these Person class, there are millions of documents added and I want to retrieve all the documents and prepare a report what are all documents available with their metadata. How can we achieve this?

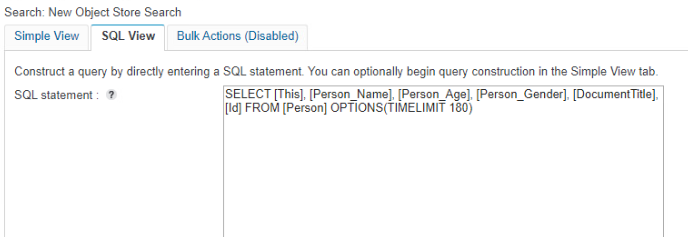
* 1. Using java utility
  2. Using Background Search Sweep Job

**Steps for doing using Background search sweep Job:**

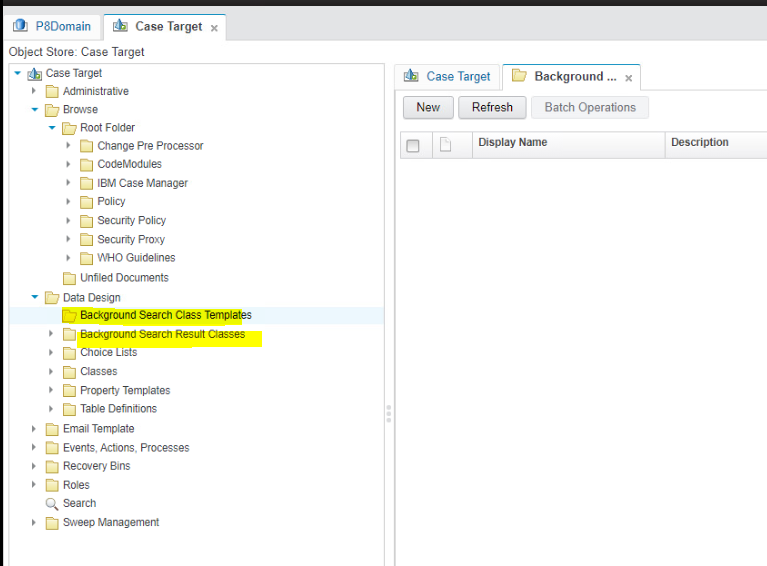
1. **Create the Person class with those Properties and create few document instances**

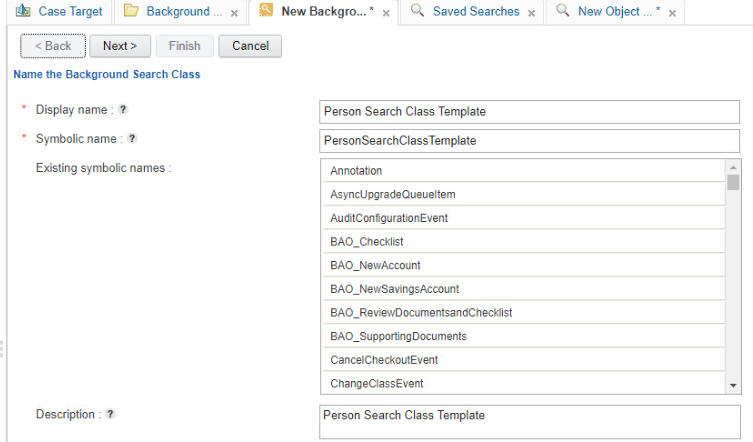


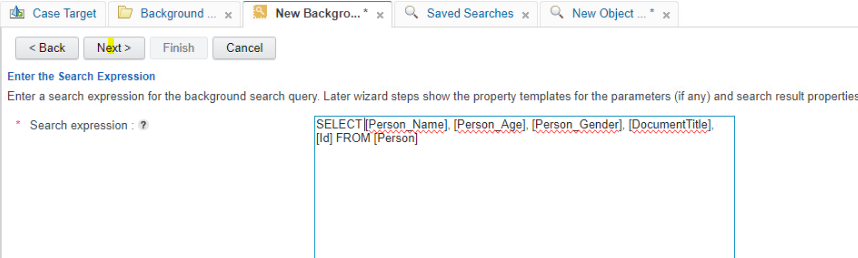
1. **We need to get the sql query for the Person class with the properties we want. We can get this using Object Store search**

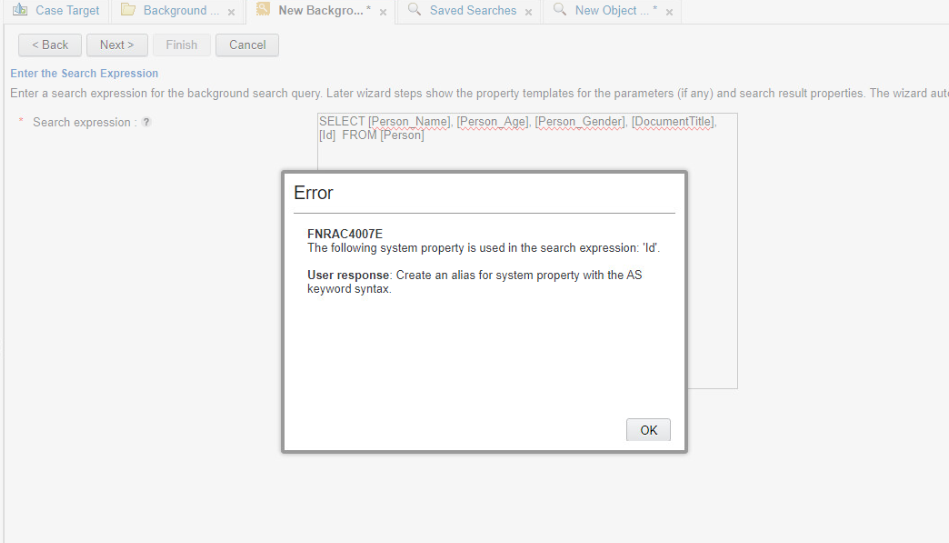


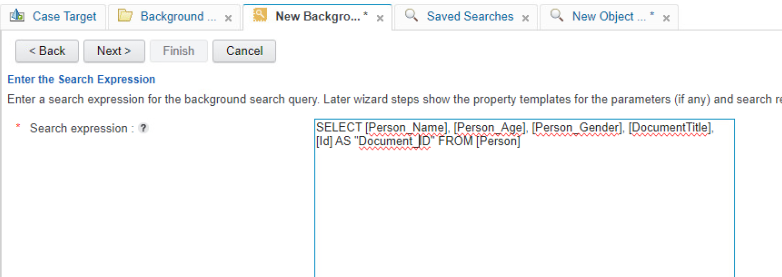
1. **Create the Background Search Class Template and Search Result Class**



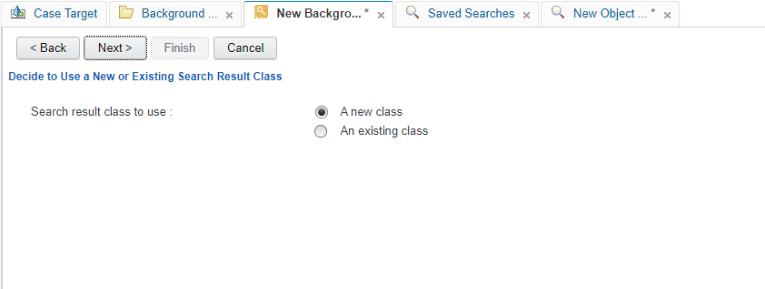


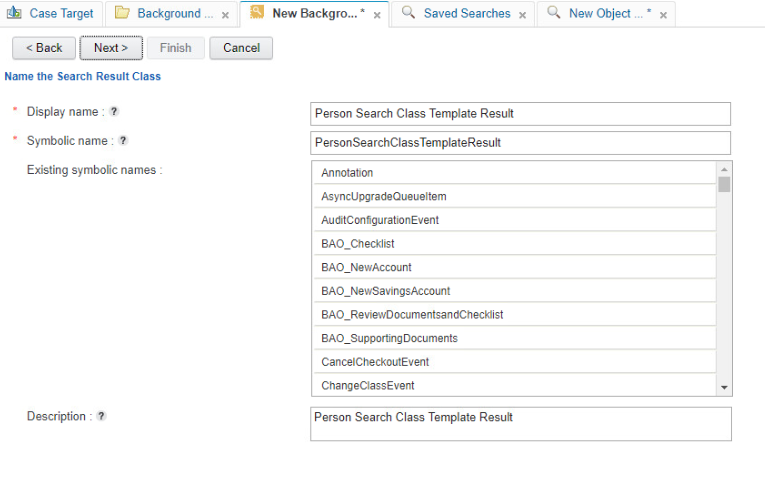


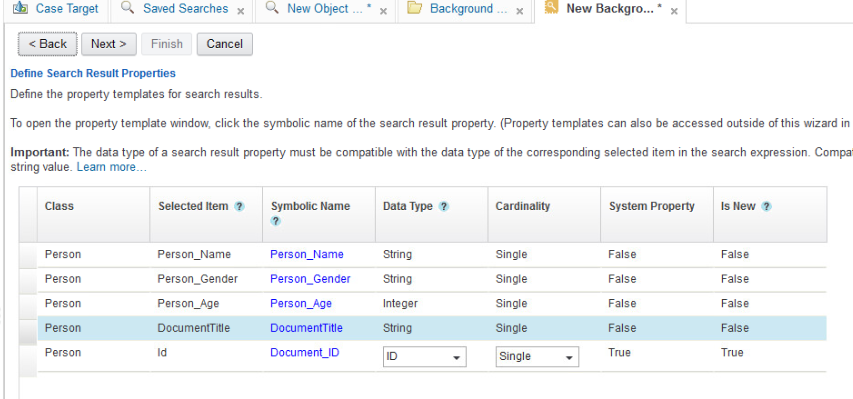


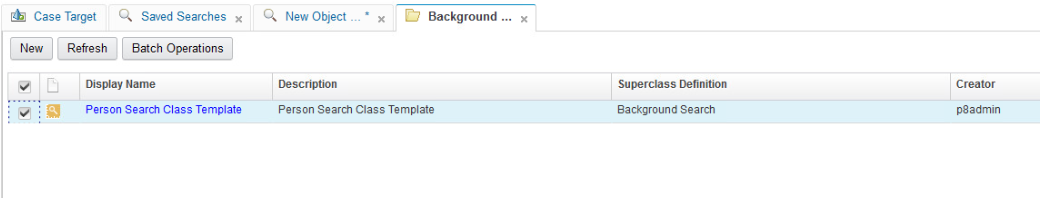


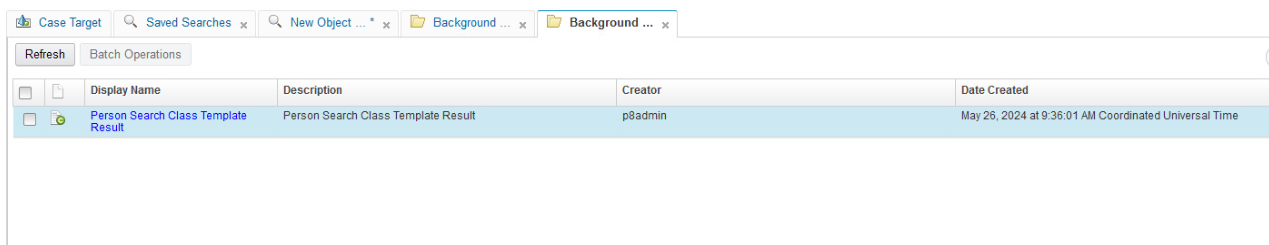
**Remember while giving alias, make sure no spaces in that as it will be created as symbolic name for the property in the search result class**

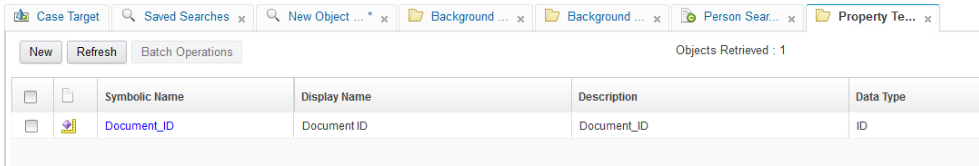






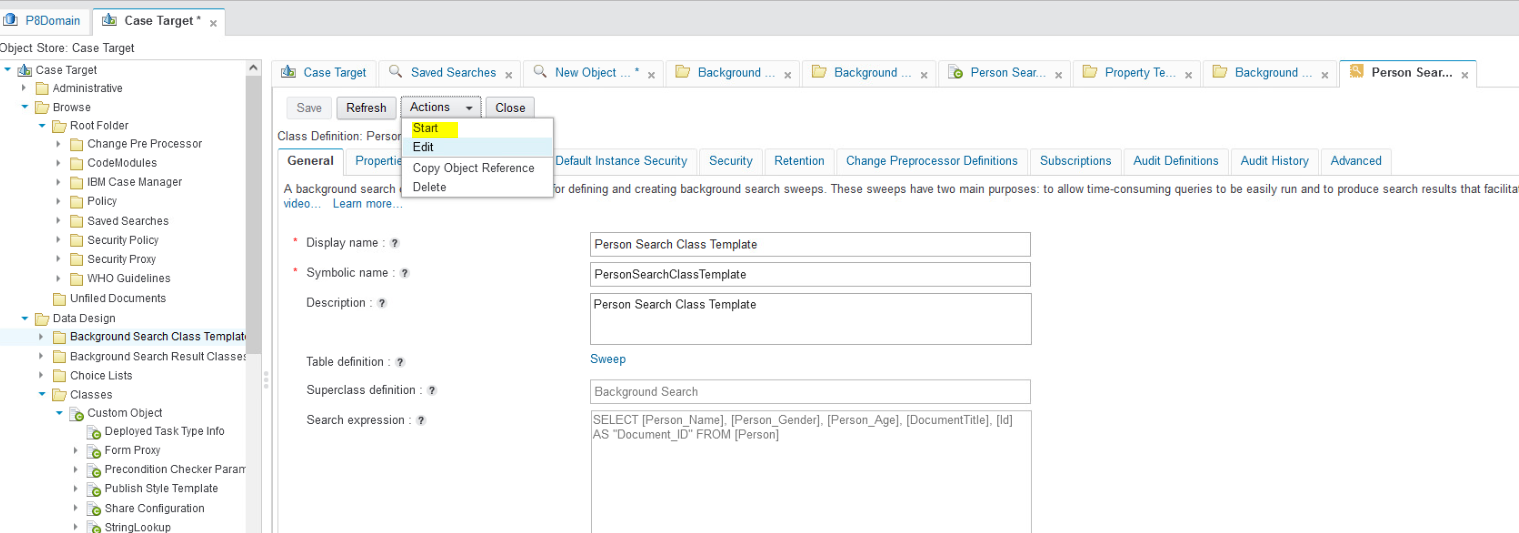


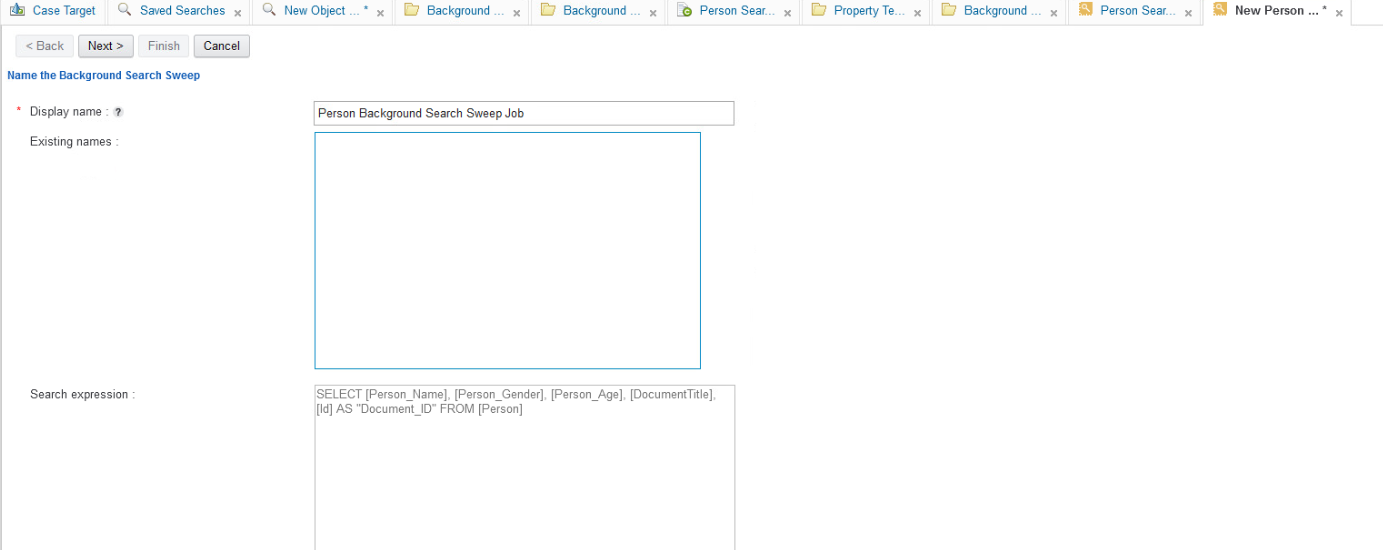


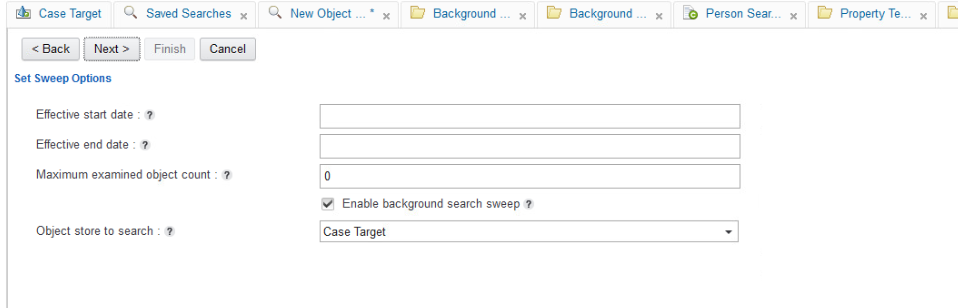


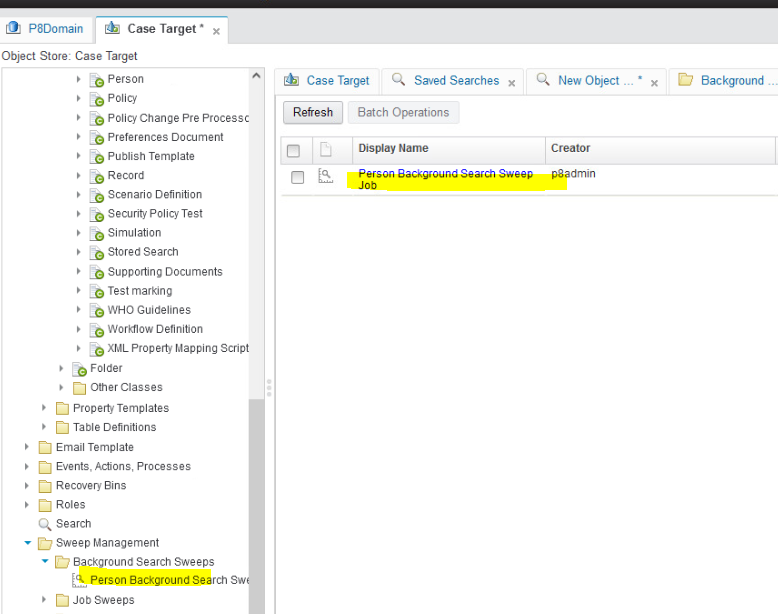
1. **Create the Background Search Sweep Job**

Go to Background Search Class template and open it and start it and it will create a background search Sweep Job

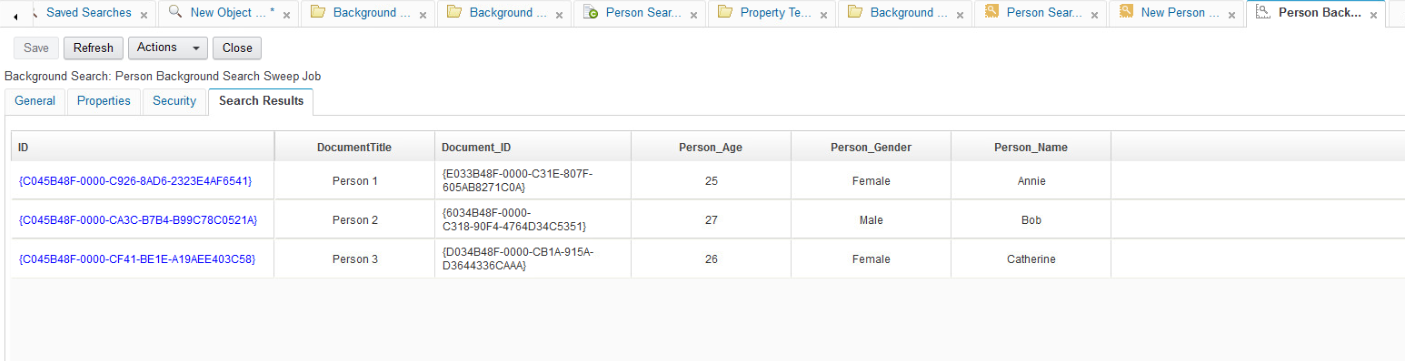




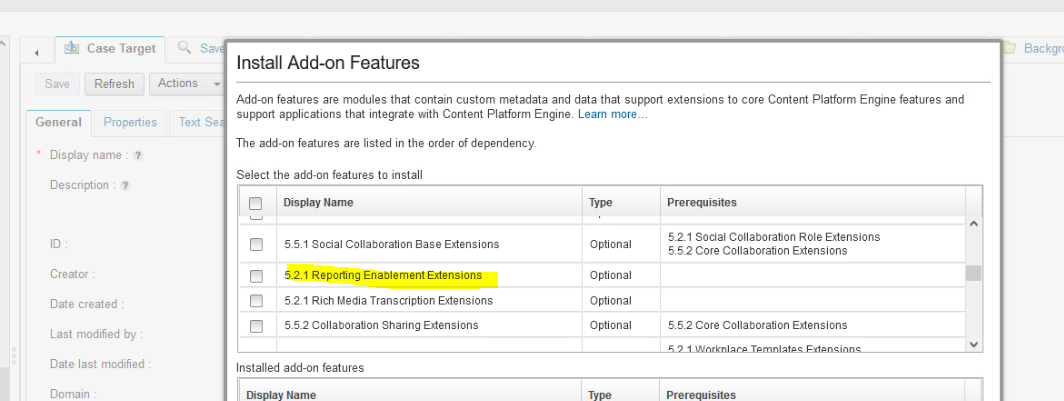




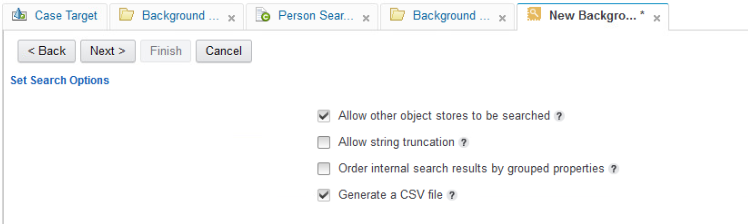
1. Check the results

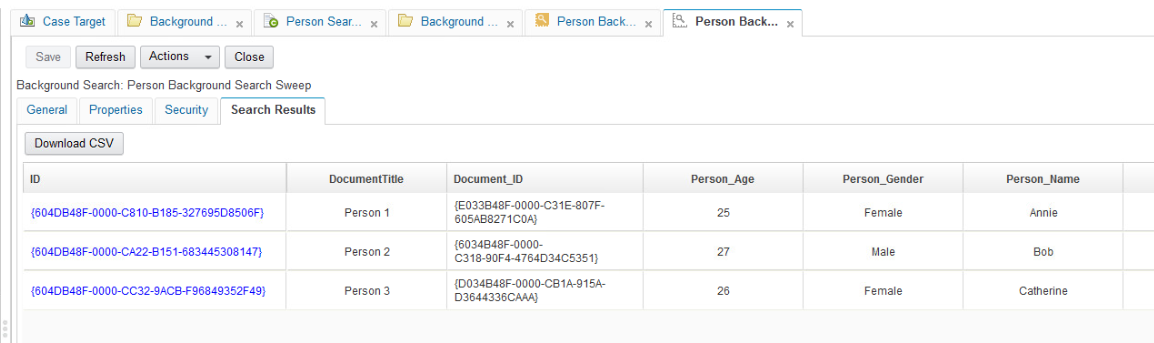


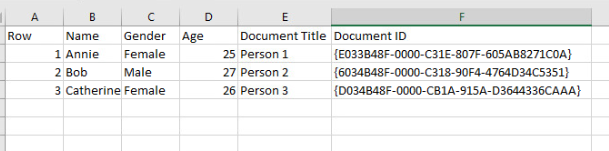
Here, we have drawback as we cannot export the results to any file. For this purpose only we have an feature to export the results to an .csv file but to do that we need to add the add on “ **Reporting Enablement Extensions”**



1. After installing that we need to repeat the steps 1 to 5 to get that option like below by deleting the existing one







## 2.8 Content Engine Java API

### 2.8.1 Setting up Eclipse

### 2.8.2 Creating a FileNet Connection

### 2.8.3 Working with Documents

### 2.8.4 Creating and Raising Custom Events

#### 2.8.4.1 Introduction

One of the nice features of the FileNet P8 Content Engine event mechanism is the possibility to define custom events. Custom events classes are created as subclasses from the CustomEvent class which is a child of the ObjectChangeEvent class. Custom events can be created and raised from code using the API.

A common use case for custom events is to trigger an event action from the client side. A typical scenario would be a business rule demanding that the change an object in the object store requires the change of other objects in the object store. If you are not sure if the initiating user has sufficient access rights on the other object store items to perform the additional changes, the operation may fail. By delegating the changes to the event action you are creating a controlled, server side, environment where you are sure of sufficient access rights to complete the action.

Custom events can also be used for specific auditing needs. Out-of-the box the Content Engine auditing mechanism is coupled to generic events such as creation, checking in and deleting. If in a custom application you are using a specific folder as a trashcan then filing a document in that particular folder signifies some specific action. You could define a “Trashing event” raised each time a document is filed in the folder. Now the trashing action can be audited with the document that was trashed as the source object. The required integer property EventStatus can be used to indicate if the action was successful or unsuccessful.

Here we will use the custom event mechanism in combination with our beloved number dispenser in another manner. Suppose the unique number must be set on a document when it is filed in a particular folder. The anatomy of the FileEvent class is slightly different from the CreationEvent class we saw previously. If a document is filed into a folder the source object points to an instance of a DynamicReferentialContainmentRelationship class (as promised before, an even longer class name). The Head property of this object points to the document being filed. We could change the original event action handler to handle this new situation or introduce a custom event and leave the original event action handler unchanged. As this section deals with custom events we obviously choose for the somewhat artificial option of a custom event. We will see later in this section that this custom event can also be used in a standalone situation.

The class name of the new custom event is DocumentFiledEvent and it is created as a subclass of the CustomEvent class using the FileNet Enterprise Manager. Although possible, no extra properties are added to the event class. We will use an event action to instantiate the custom event. The Java class used for the event action handler is called DocumentFiledEventActionHandler. The code looks like this:

public class DocumentFiledEventActionHandler implements EventActionHandler {

public void onEvent(ObjectChangeEvent event, Id subscriptionId)

throws EngineRuntimeException {

if ( event.get\_SourceObject() instanceof DynamicReferentialContainmentRelationship ) {

DynamicReferentialContainmentRelationship relationship =

(DynamicReferentialContainmentRelationship) event.get\_SourceObject();

IndependentObject object = relationship.get\_Head();

if ( object instanceof Document ) {

raiseCustomEvent((Document) object);

}

}

}

private void raiseCustomEvent(Document document) {

ObjectStore objectStore = document.getObjectStore();

CustomEvent customEvent = Factory.CustomEvent.createInstance(objectStore,"DocumentFiledEvent" );

customEvent.set\_EventStatus( new Integer(0) );

document.raiseEvent(customEvent);

document.save(RefreshMode.NO\_REFRESH);

}

}

The code in the onEvent() method makes sure that we are dealing with a document being filed into a folder. After that is established, the raiseCustomEvent() method creates the actual custom event object. Notice that you have to call the save() method of the document object to actually raise your custom event. This code is coupled to a new Event Action called DocumentFiledEventAction. To achieve what we want, we need two subscriptions:

* First a subscription has to be placed on the folder where you have to file the document receiving the unique number. This can be either a folder class or a specific instance of a folder. This subscription is coupled to the DocumentFiledEventAction and triggered with the “File Event”.
* The second subscription is placed on the class of the documents receiving the unique number. This subscription is coupled to the previously discussed DispenserEventAction and triggered by the custom “Document Filed Event”. Make sure that you supply the correct configuration for the dispenser in the user string.

Now that all the pieces are in place, the document should get a unique number if it is filed in the configured locations. Filing the document in the folder triggers the DocumentFiledEventAction, this creates a “Document Filed Event” triggering the DispenserEventAction.

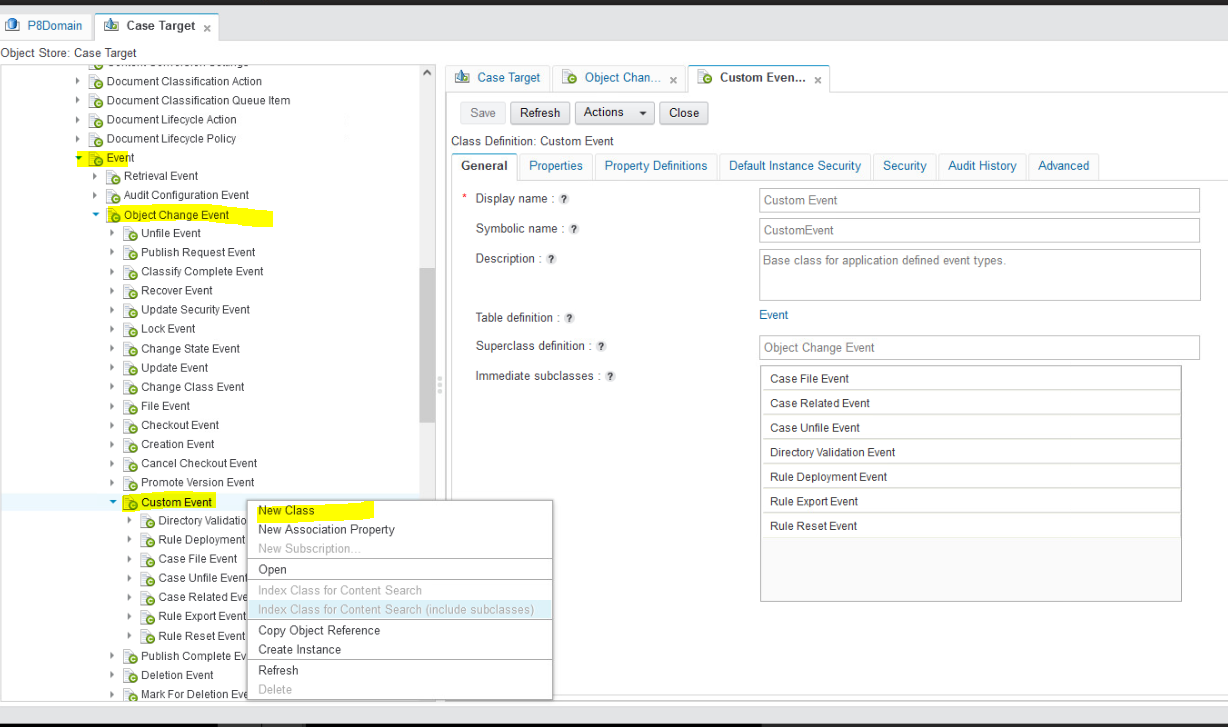
You can also use custom events to trigger work flow subscriptions. The benefit of using the “Document Filed Event” to start a work flow is that the document being filed can be the launching attachment of the work flow. This is in contrast with the normal file event which gives you an instance of a DynamicReferentialContainmentRelationship class. To take this on step further you could use custom events to start work flows based on business logic that cannot be configured with the standard configuration methods. The business logic is placed in client code or an event action handler and a custom event is raised whenever the specific conditions are met. This custom event will on his turn trigger the start of a work flow.

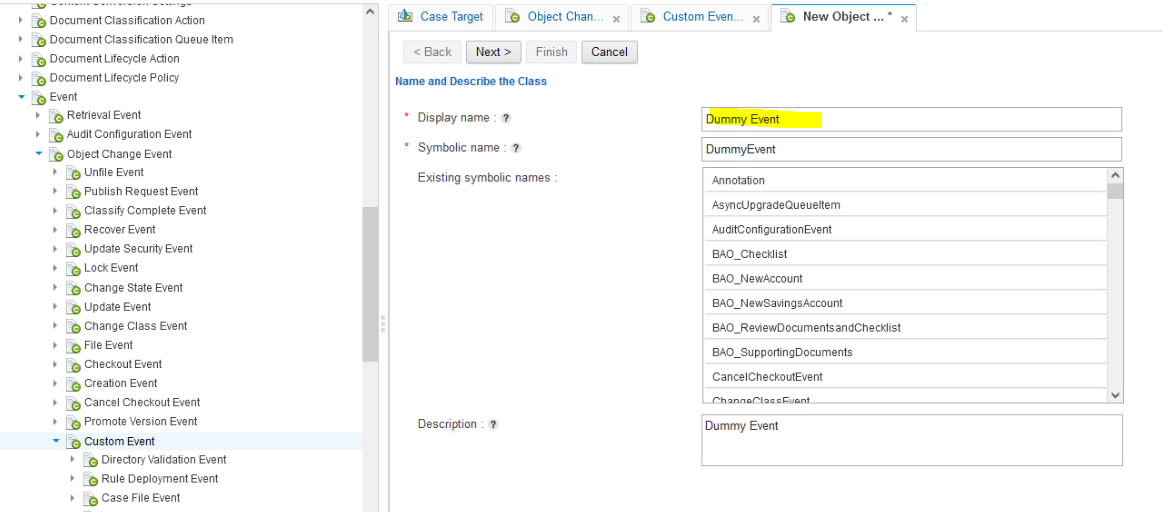
Custom events can be created and raised from code using the API.

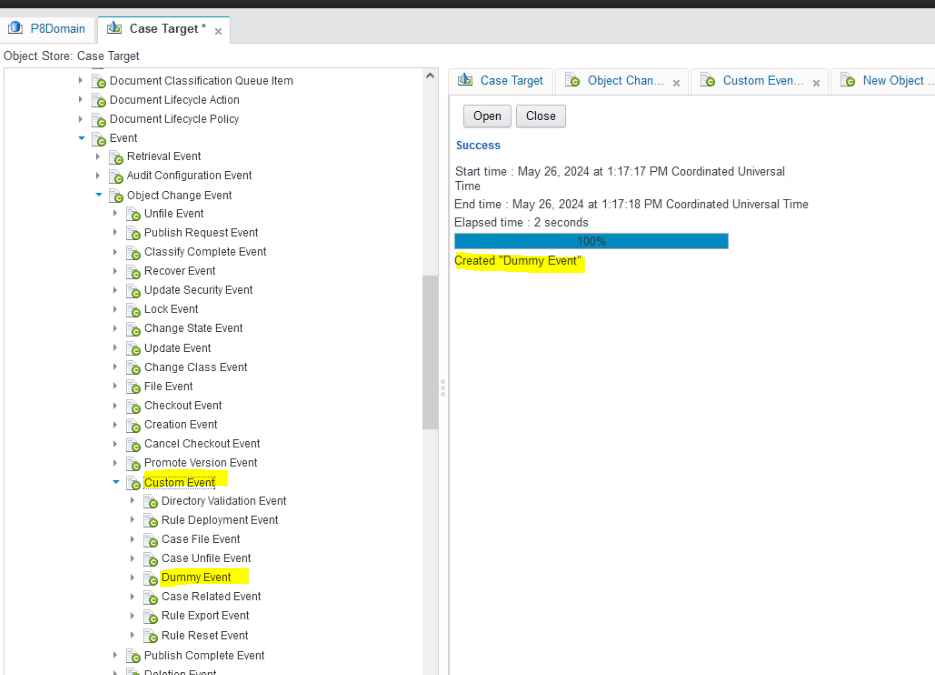
#### 2.8.4.2 Scenario and Example

Let us understand this custom event concept by doing the following

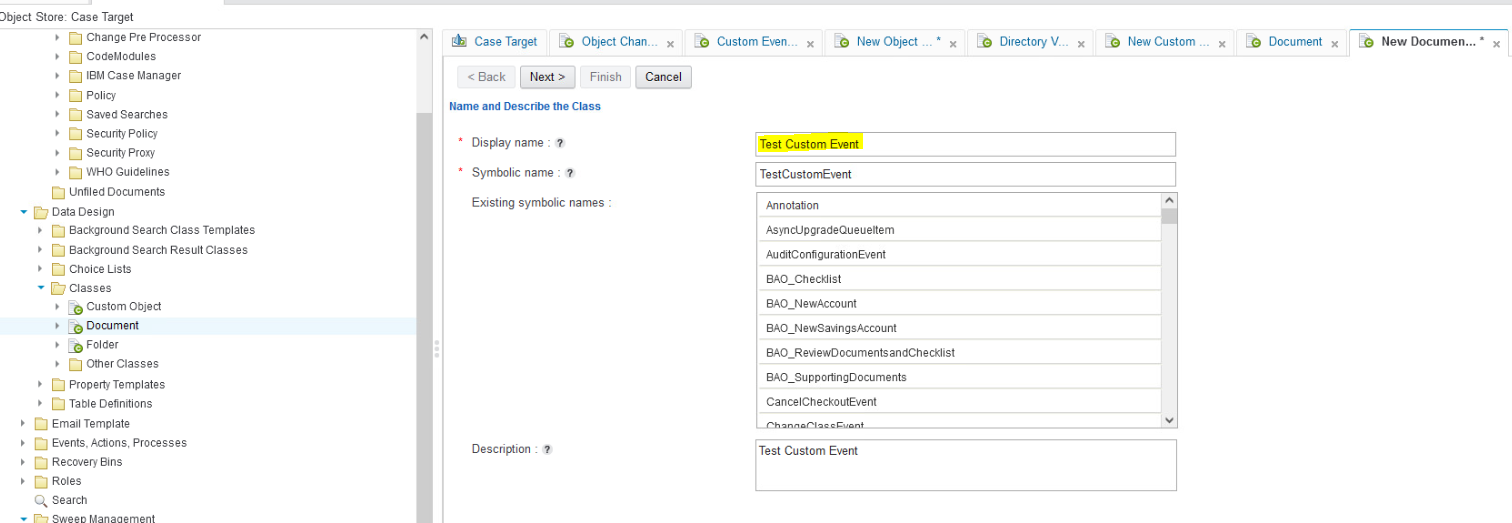
* 1. **Create a custom event class name ( Dummy Event )**



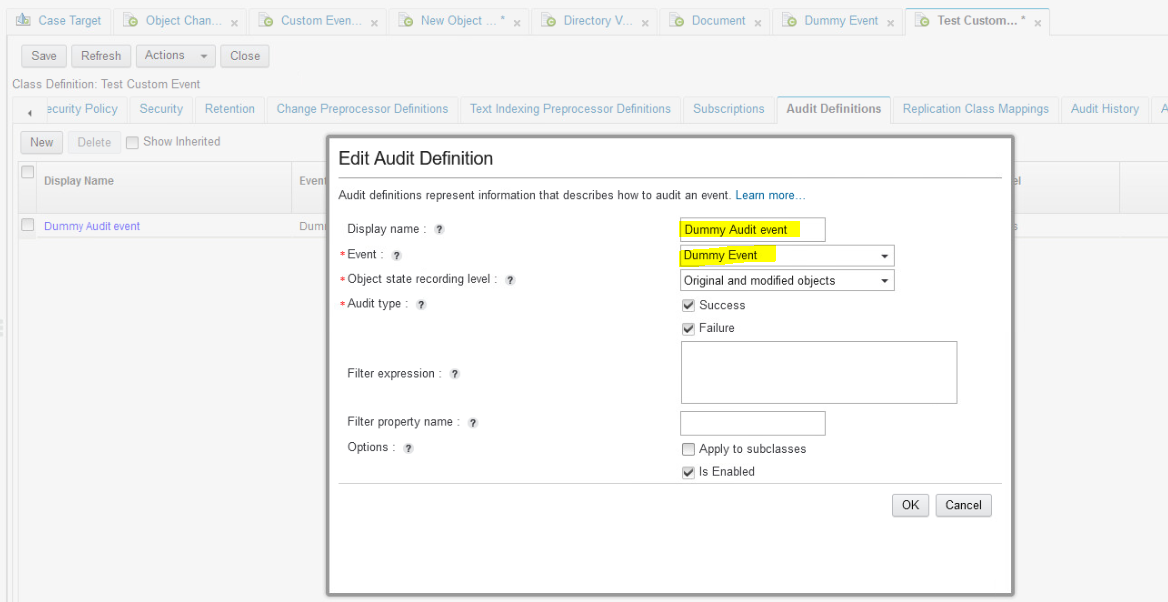




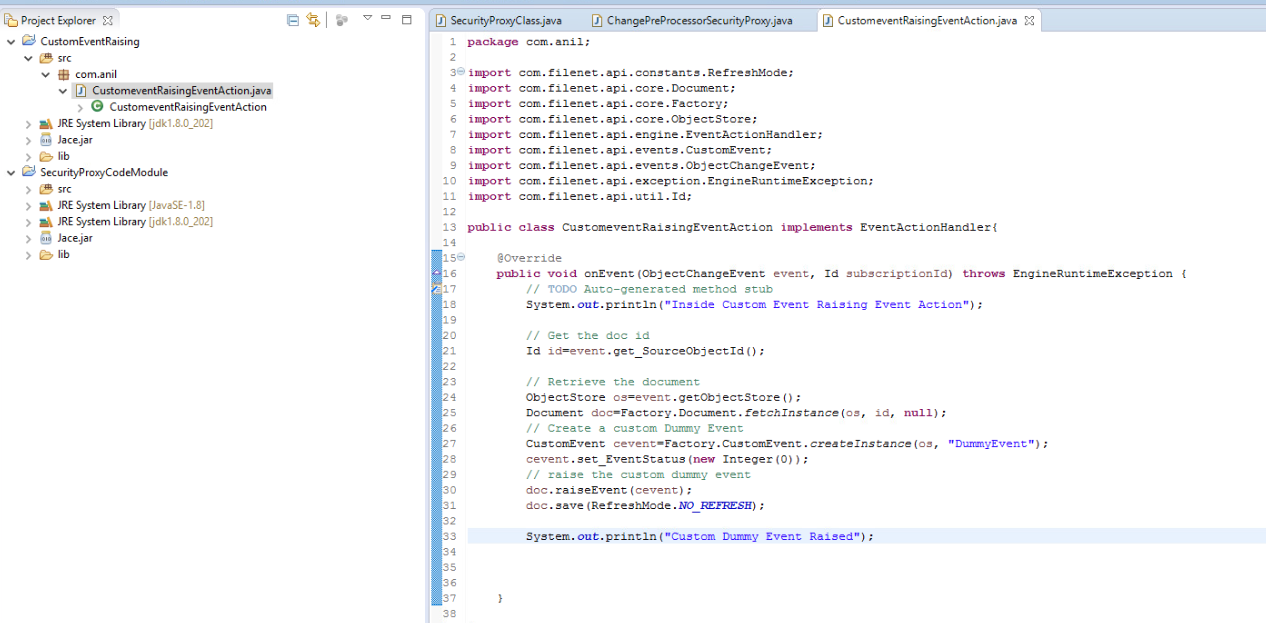
* 1. **Create a New Document class**

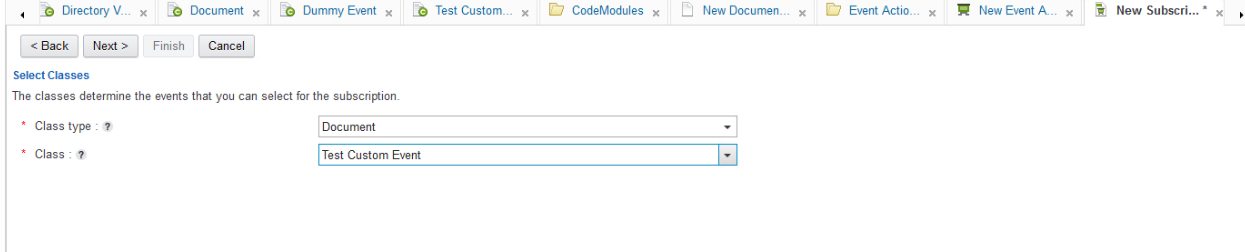
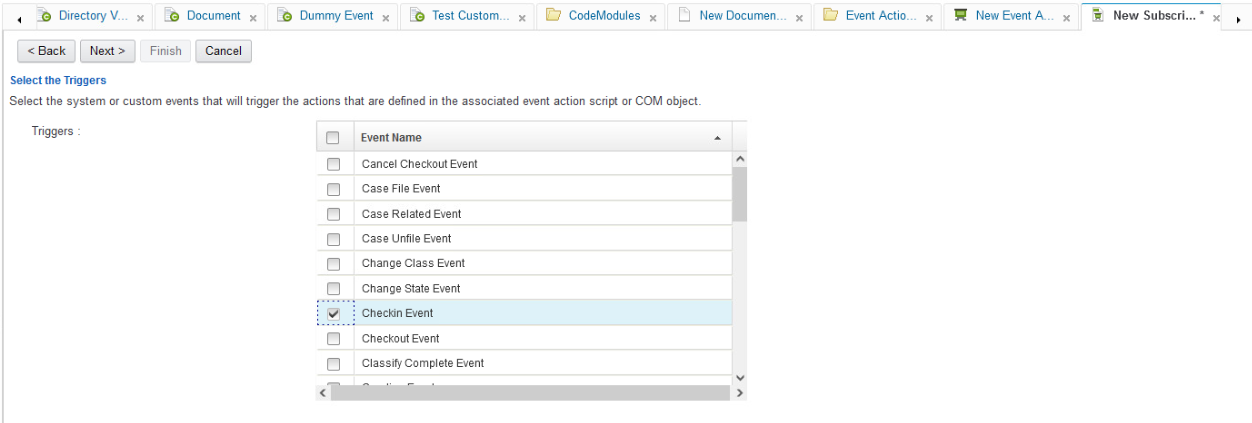


* 1. **Create an Audit Definitions for custom Event for this document class**

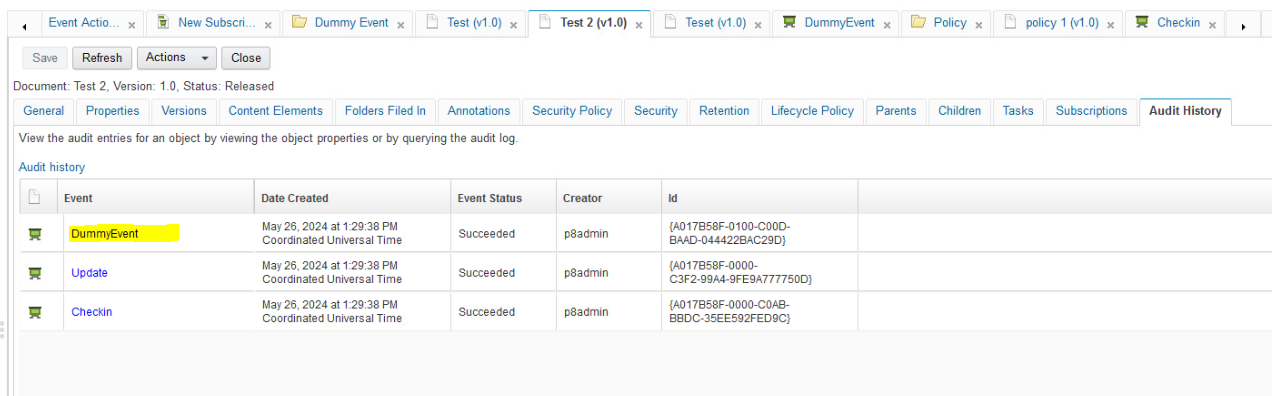


* 1. **Create a event action for the above document class for checkin**



* 1. **Raise the custom event for the same document in the check in event subsctiption**
  2. **Verify whether the dummy event got registerd on the Audit history of the document or not**



# 3. IBM Content Navigator

## 3.1 Introduction to Navigator

## 3.2 Walkthrough to Navigator Admin desktop

## 3.3. Development in Navigator

### 3.3.1 Creation of Object Store repositories

### 3.3.2 Creation of Desktops

## 3.4. Navigator Customizations

### 3.4.1: Creating Request Filter

### 3.4.2 Creating Response Filter

### 3.4.3 Creating Action Plugin

### 3.4.4 Creating Plugin Service

### 3.4.5 Creating Feature Plugin

# 4. IBM Case Manager

# ANNEXURE :

Useful Links:

# **Developing Event Action Handlers for the FileNet P8 Content Engine**

<https://ecmdeveloper.com/contentengine/developing-event-actions/>

* 1. **Developing Change Preprocessors for the FileNet P8 Content Engine**

<https://ecmdeveloper.com/contentengine/developing-change-preprocessors/>