

An event-driven framework for integrating IBM Content Manager with IBM FileNet Business Process Manager

Skill Level: Introductory

[Alan Young \(ayaung@us.ibm.com\)](mailto:ayaung@us.ibm.com)

Senior Software Engineer

IBM

[Mimi Vo \(mimivo@us.ibm.com\)](mailto:mimivo@us.ibm.com)

Senior Technical Staff Member

IBM

18 Jun 2009

IBM® Content Manager is a scalable, high performing content repository, while IBM FileNet® Business Process Manager (BPM) is a comprehensive suite of products for managing business processes involving people, content, and systems. The integration of IBM Content Manager with FileNet BPM enables you to convert content objects into active content by using an event framework. In addition, the events that are supported in Content Manager automatically start processes in FileNet BPM. By using a callback interface (content extended operations), FileNet BPM can access and search documents in Content Manager. Learn how to integrate IBM Content Manager Version 8.4.1 with FileNet BPM, and follow an example auto claims scenario demonstrating how to start a FileNet BPM process from Content Manager.

Introduction

IBM Content Manager Version 8.4.1 is enabled for events. Events on content (such as create, update, delete, and so on) are monitored and can trigger FileNet BPM processes. The event framework is pivotal for the direct integration of IBM Content Manager Version 8 (hereafter referred to as CM8) as a content repository, and FileNet P8 Business Process Manager (hereafter referred to as BPM) as a process engine. At definition time, the framework allows events to be subscribed to a specific

item type in CM8. At run time, the framework provides a foundation for the following:

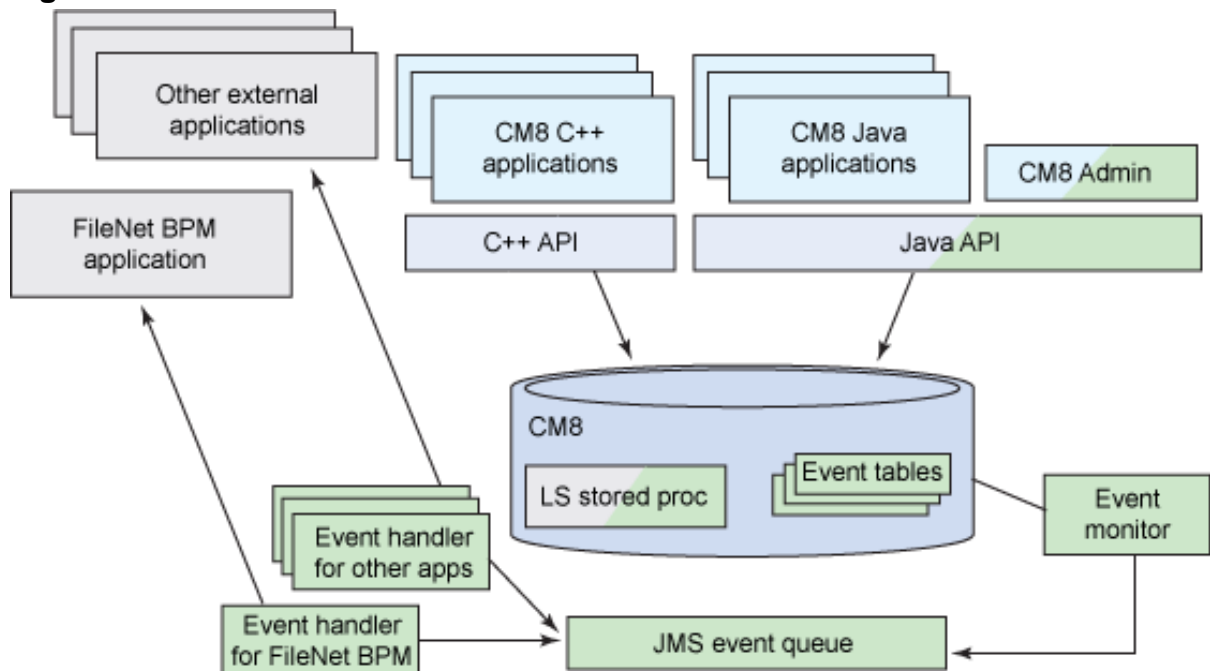
1. Monitoring events from operations in CM8
2. Processing events to enable the launch of BPM processes based on the events in the repository
3. Operating on content from BPM processes

In this article, you will be brought to a deeper understanding of the IBM Content Manager event framework and its integration with FileNet Business Process Manager.

Architecture of an event framework

Figure 1 illustrates the architecture of an event framework supporting the integration of IBM Content Manager and FileNet BPM. To support FileNet BPM integration and integration of other applications in the future, it is important to provide an event framework that does not compromise CM8 current performance, transaction integrity, or published interface. At the same time, it accommodates the openness required for application integration.

Figure 1. Event framework architecture



The event subscription data is stored in the configuration tables in the CM8 library server database. This data is configured by using the CM8 system administration client. Administrators can set up a subscription for the events to be monitored for a

specific item type, and also define the BPM process to be launched.

The library server logs the events when they occur according to the configuration data. Those logged events are placed into an event queue table in the library server database. An event contains event type, event ID, item information, and attribute data.

As shown in the architecture diagram in Figure 1, an event monitor and an event handler are introduced for message generation and message consumption respectively. The event monitor fetches event data from the event queue table based on the configuration data, converts these events into Java™ Message Service (JMS) messages, and puts these JMS messages on a JMS event queue.

In the meantime, an event handler reads JMS messages, which carry the event data, from the JMS event queue and provide the ability to launch BPM processes with CM8 document attributes. Though Content Manager Version 8.4.1 provides an event handler for BPM integration, it is possible to develop custom event handlers for various applications.

Also, for FileNet BPM to have the ability to access and operate on CM8 documents, a set of callback interfaces is provided, which is known as Content Extended Operations. These APIs are called from the BPM engine to access CM8 documents and operate on them as specified by process designer rules. These operations are completed automatically by the BPM engine without end user interaction.

Through CM8 Java and C++ API interfaces, CM8 applications interact with the library server which generates the events for the event monitor based on the specified event subscription definition. The event handler retrieves JMS messages and parses the events to launch the requested BPM processes accordingly.

A scenario

We will start with a typical customer requirement in the insurance industry. Assume that we have a workflow process for auto claim processing. When a customer service representative submits an auto claim, an auto insurance claim process is started automatically. Depending on the claim amount, the claim might go through a different route. For a regular claim amount which is not more than \$2000, the claim is reviewed by an adjuster. For a large claim amount which is more than \$2000, the claim is reviewed by a supervisor. With a decision of approval, a payment request is submitted. Otherwise, with a decision of rejection, a rejection letter is sent to the customer. Regardless of the decision, the workflow reaches the last step and the auto claim processing is completed.

Below is a scenario to illustrate the flow between CM8 and BPM in the document creation case as an example in general. We will use this scenario to take you

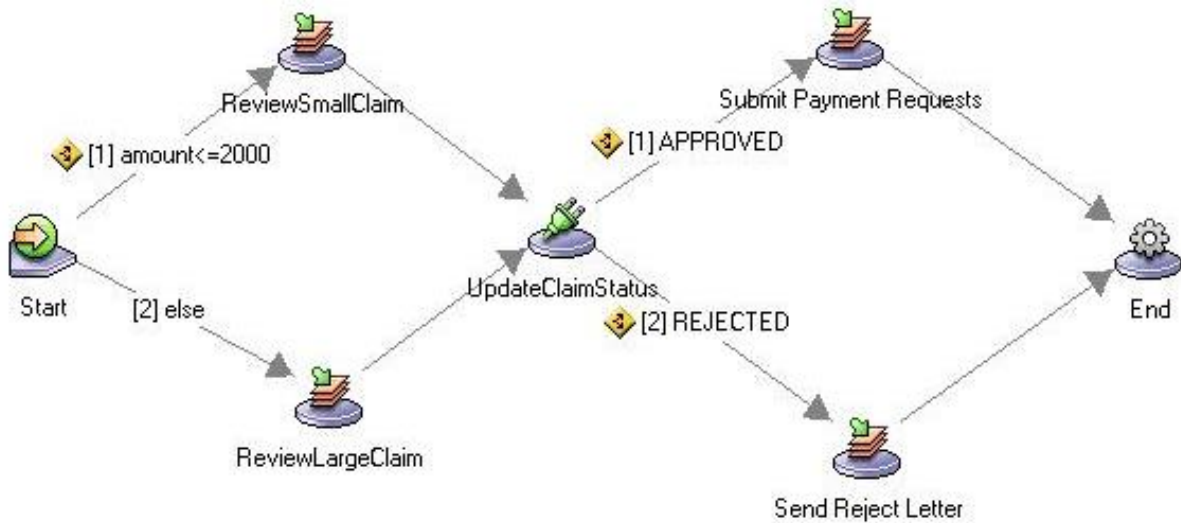
through various steps in the event framework in later sections.

- An auto insurance claim document A of item type "AutoClaim" is created in Content Manager by a customer service representative with a claim amount of \$4000.
- Because the administrator configured this item type "AutoClaim" to monitor upon an "Add Item" event for document creation, the library server inserts a row of event data including attribute values such as "claimID" and "claimAmount" into the event queue table.
- An event monitor scans the event queue table for committed rows, and retrieves this "Add Item" event for document A. The event monitor also retrieves from a configuration table for configuration data such as the BPM process name "AutoInsuranceClaim" to which a reference to document A should be attached at the launch time. It converts the configuration information along with the event data retrieved from the event queue table into a self-describing event message in a common base event format (called CBE format). The event message is then sent to a Java Message Service queue (JMS queue).
- An event handler, which is listening to the JMS queue, retrieves the document creation event from the queue. Next, it parses the event data from the CBE format and then uses this information to call BPM API to launch a BPM process. An instance of the BPM process "AutoInsuranceClaim" is instantiated with an attachment containing a reference to the claim document.
- At the first step in the "AutoInsuranceClaim" process, the claim is evaluated upon the claim amount and automatically routed to a step for a small claim or a step for a large claim. Because the document A's attribute "ClaimAmount" is mapped to BPM process attribute "claimAmount", BPM can determine where the claim should be routed. For a small claim, it is reviewed by an adjuster, while a large claim is reviewed by a supervisor.
- A day later, a supervisor looks at his inbox for work assignments. He reviews and approves the claim document A. At the same time, the claim status of claim document A in CM8 is updated automatically. Since the claim is approved by the supervisor, a payment request is submitted accordingly and the claim process is completed.

Figure 2 shows an auto insurance claim process for the previous integration scenario. This process is modeled with the FileNet Process Designer. A claim goes through the process is either approved or rejected. First, an auto claim is submitted by a customer service representative. The claim is evaluated at the Start step. If the claim amount is greater than \$2000, the claim automatically goes to the

ReviewLargeClaim step for comprehensive review. Otherwise, the claim automatically goes to the ReviewSmallClaim step for regular review. The result of review from either the ReviewLargeClaim step or the ReviewSmallClaim step is collected in the UpdateClaimStatus step. If the result is "APPROVED", a payment requested is submitted. If the result is "REJECTED", a rejection letter is sent. Finally, the claim reaches the end of the auto insurance claim process.

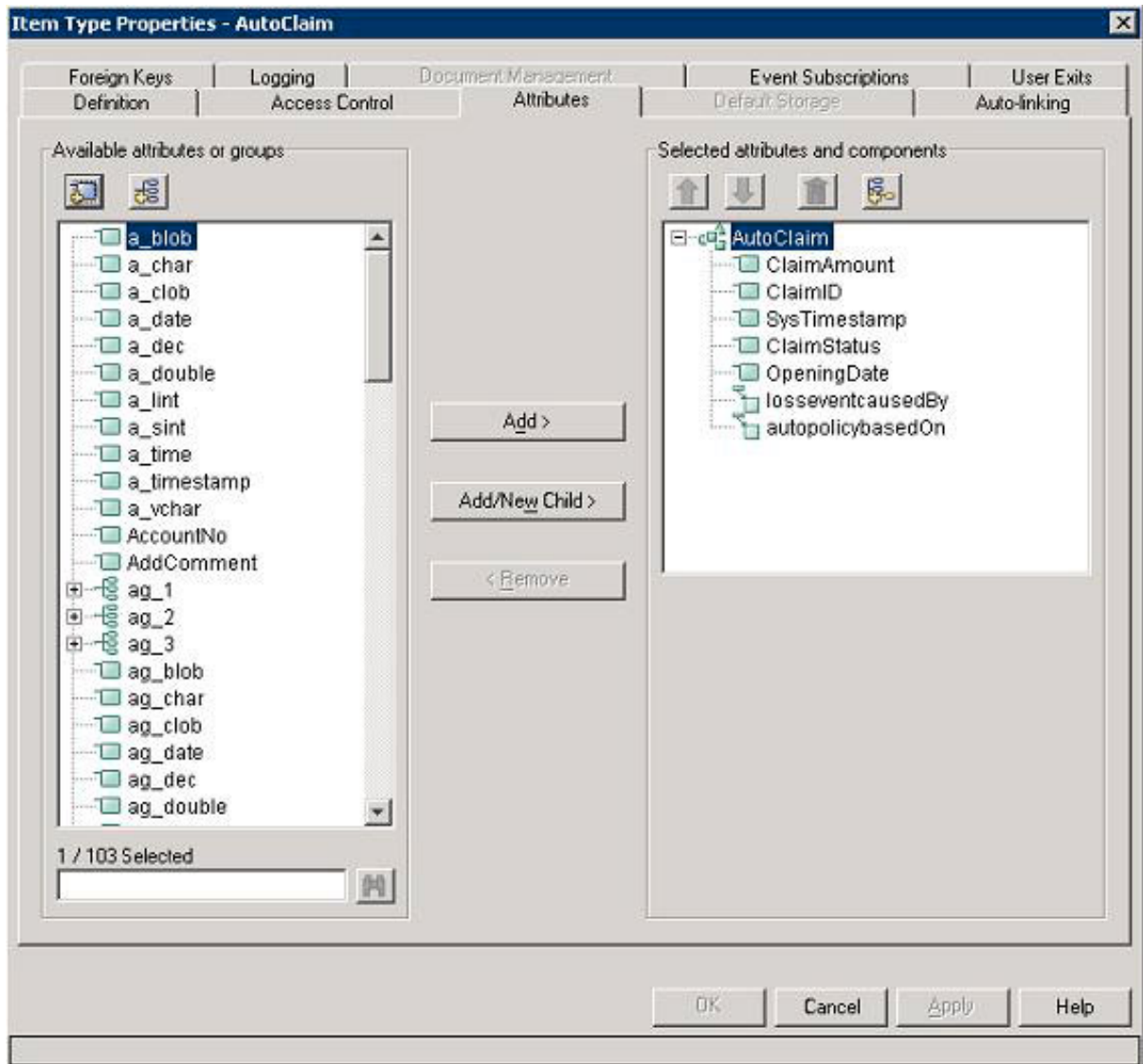
Figure 2. Auto insurance claim process



An auto claim item type is shown in Figure 3 for this scenario. It is defined through the system administration client of CM8. It contains the following attributes:

- ClaimID: Claim ID
- ClaimAmount: Claim amount
- ClaimStatus: Status of the claim
- SysTimestamp: System timestamp
- OpeningDate: Opening date of the claim
- losseventcausedBy: Loss event referenced by the claim
- autopolicybasedOn: Auto Policy referenced by the claim

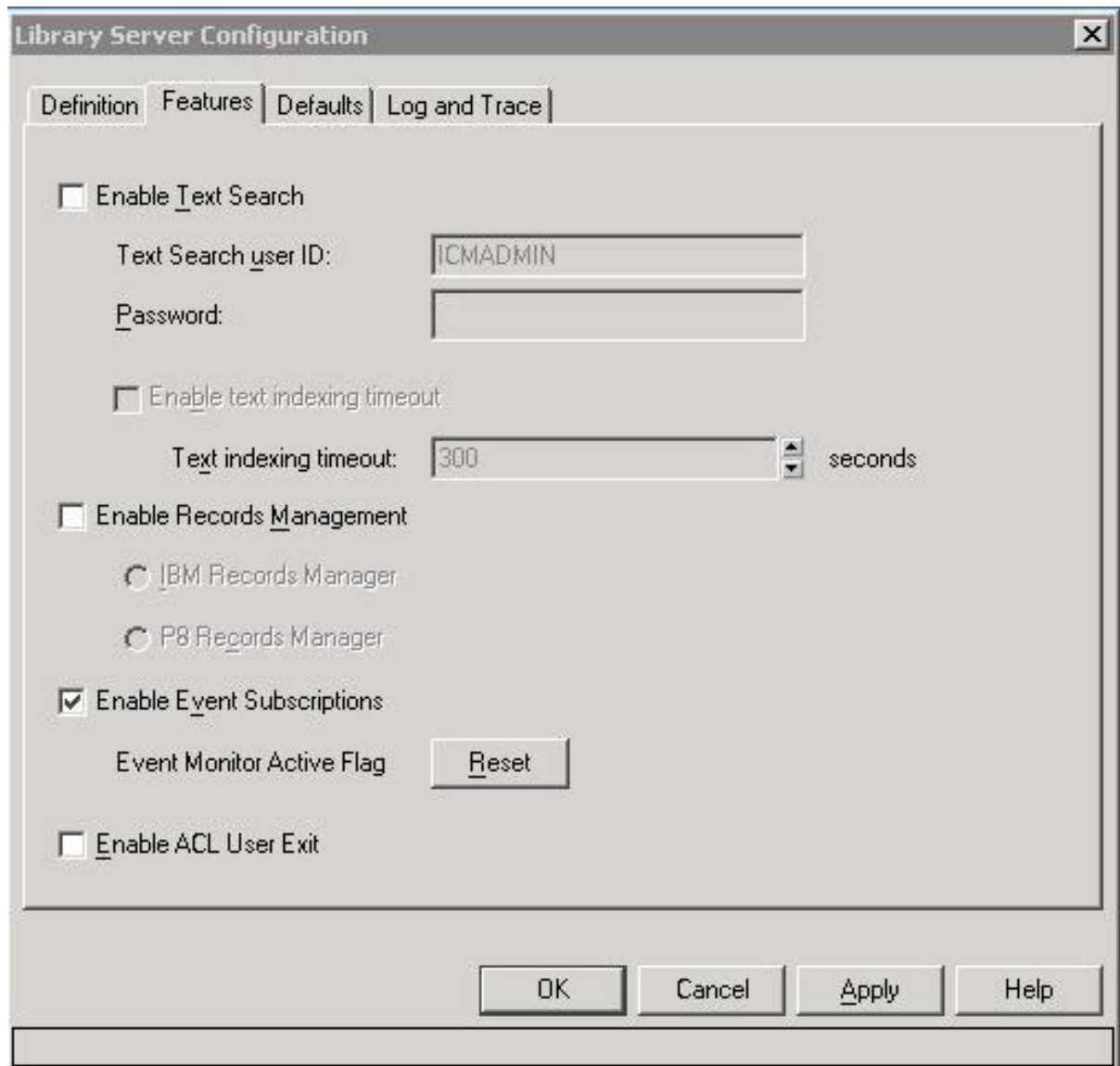
Figure 3. Auto insurance claim item type



Event subscription

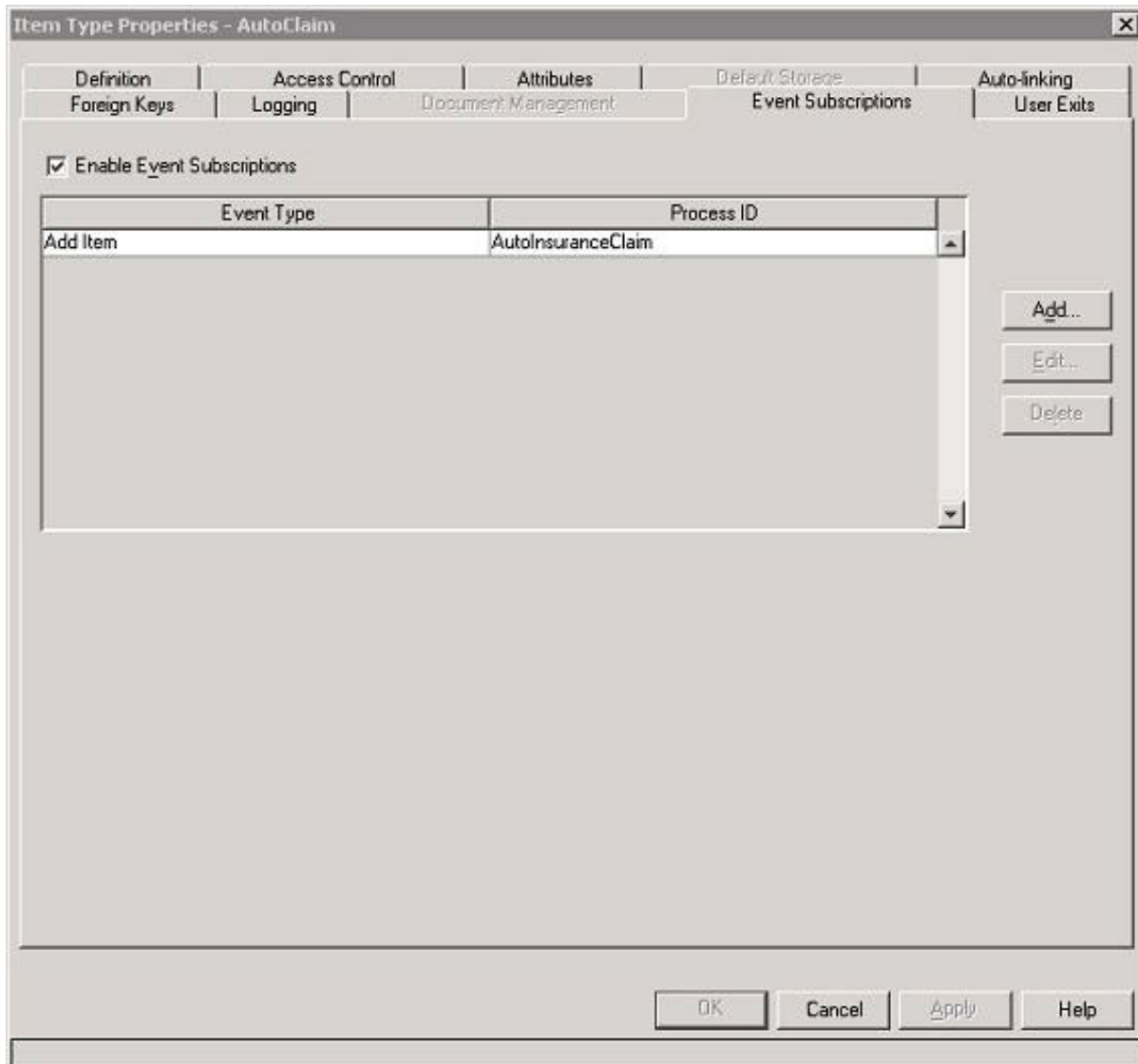
This section illustrates the steps for event subscription. Event subscription is enabled at two levels: library server level and item type level for better granularity. The "Enable Event Subscription" checkbox in the Library Server Configuration dialog (see Figure 4) must be selected to enable the event logging for a library server instance.

Figure 4. Enable the event subscription



At the item type level, Figure 5 shows an example of event subscription of an item type "AutoClaim". The "AutoClaim" item type has been described in Figure 3. Following the scenario, the event to be subscribed for this item type is an "Add Item" event, and the BPM process to be launched is the "AutoInsuranceClaim" process. The "Event Subscription" page provides the support of the item level event subscription.

Figure 5. Event subscription of an item type



When the "Add" button in the "Event Subscription" page is clicked, a "Define Event Subscription" dialog is displayed (see Figure 6). The administrator can select the event type of his interest. There are two kinds of integration:

- Process integration for FileNet BPM integration only
- General integration for general purpose application integration

Because the scenario is related to the launch of a BPM process, the "Process integration" button is selected. The "Process ID" field contains the name of the BPM process "AutoInsuranceClaim", while the mapping between attributes of the "AutoClaim" item type and process attributes is also specified. The mapping behaves like parameter passing in a procedure call. It helps the process to be content aware and reduces the user interaction at the time of process launch. For

this particular example, the mapping is defined as follows:

- "ClaimID" attribute is mapped to process attribute "claimID"
- "ClaimStatus" attribute is mapped to process attribute "claimStatus"
- "ClaimAmount" attribute is mapped to process attribute "claimAmount"

Figure 6. Event subscription of an AddItem event

Define Event Subscription

Event Type:
Add Item

Process integration General integration

Process ID
AutoInsuranceClaim

Mapping of item type attribute and process attribute

Item type attribute:

- autopolicybasedOn
- ClaimAmount
- ClaimID
- ClaimStatus
- losseventcausedBy
- OpeningDate

Process attribute:

Add

| Component Name | Group Name | Attribute | Process Attribute |
|----------------|------------|-------------|-------------------|
| | | ClaimID | claimID |
| | | ClaimStatus | claimStatus |
| | | ClaimAmount | claimAmount |

Remove

OK Cancel Apply Help

Event monitoring

After event subscription is complete for the item type "AutoClaim", the administrator starts the event monitor from a command line interface. When an auto insurance claim document of item type "AutoClaim" is created in a DB2 Content Manager by a customer service representative, the library server logs the document creation event. For this new auto claim document, suppose that the claim amount is \$4000, and the claim ID is CL1240533879046. Next, the event monitor will generate a JMS event

message carrying the event data in a CBE format. An example of the event data is listed, which illustrates how a document creation event is represented as an event message. This event message corresponds to the event subscription in the previous section.

Essentially, the common base event (CBE) format is used to package the actual event data from a library server. A CBE event contains additional information such as execution environment, hostname, product name, and so on as shown in the example. The actual event data is enclosed within the <contextValue> and </contextValue> tags.

Listing 1. Example of event data

```
<CommonBaseEvent creationTime="2009-04-07T18:56:15.281Z" extensionName="CMEvent"
globalInstanceId="A1001001A09D07B15545J3334400000000000000000000285001" priority="100"
version="1.0.1">
  <contextDataElements name="cmevent" type="string">
    <contextValue>
      ICMEMSTART;
      ETYPE=ITEM;
      EACTION=CREATE;
      ECODE=301;
      ITEMID=A1001001A09D07B15545J33344;
      ITEMTYPE=AutoClaim;
      PID=88 3 ICM8 ICMNLSDB9 AutoClaim59 26 A1001001A09D07B15545J3334418
      A09D07B15545J333441 14 1007;
      NCOMPTYPE=1;COMPTYPE=AutoClaim;COMPNUMBER=1;NATTRS=3;
      ATTRNAME=ClaimStatus;ATTRLEN=0;ATTRTYPE=452;ATTRVAL=;
      ATTRNAME=ClaimAmount;ATTRLEN=8;ATTRTYPE=484;ATTRVAL=4000.0;
      ATTRNAME=ClaimID;ATTRLEN=15;ATTRTYPE=448;ATTRVAL=CL1240533879046;
      ETIME=2009-04-07-18.55.45.937000;
      NPROCESS=1;
      PROCESS=AutoInsuranceClaim;PVERSION=;PTYPE=1;NMAPATTRNAME=3;
      MAPATTRNAME=claimID;CMATTRNAME=ClaimID;
      MAPATTRNAME=claimStatus;CMATTRNAME=ClaimStatus;
      MAPATTRNAME=claimAmount;CMATTRNAME=ClaimAmount;
      ICMEMEND
    </contextValue>
  </contextDataElements>
  <sourceComponentId application="Content Manager Event Monitor"
component="Content Manager 8.4.01.000" componentIdType="ProductName"
executionEnvironment="Windows Server 2003[x86]" instanceId="1"
location="cm1141/9.30.152.151" locationType="Hostname"
subComponent="Event Monitor" componentType="ContentManager"/>
  <situation categoryName="OtherSituation">
    <situationType xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="OtherSituation" reasoningScope="EXTERNAL">
      Application Event
    </situationType>
  </situation>
</CommonBaseEvent>
```

The event data consists of three kinds of information:

- Event information - Describes event type, event action, event code, and event time
For example, ETYPE=ITEM indicates the event type, EACTION=CREATE describes the event action, ECODE=301 specifies

the event code, and ETIME=2009-04-07-18.55.45.9370000 represents the event time.

- Item information - Describes item type, item ID, persistent ID and attribute data
For example, ITEMTYPE=AutoClaim specifies the item type, ATTRNAME=ClaimAmount indicates an attribute name, and ATTRVAL=4000.0 describes the attribute value of the ClaimAmount attribute. Similarly, ATTRNAME=ClaimID specifies another attribute name, and ATTRVAL=CL1240533879046 describes the attribute value of the ClaimID attribute.
- Process information - Describes process ID and the mapping between process attributes and item attributes
For example, PROCESS=AutoInsuranceClaim specifies the process name, MAPATTRNAME=claimAmount indicates the mapping attribute, and CMATTRNAME=ClaimAmount represents the process attribute to be mapped.

Event processing

The administrator starts the event handler from a command line interface after the event monitor was started. The event handler listens to the JMS event queue for any incoming event messages which are sent by the event monitor. The event messages are retrieved from the JMS event queue. The event data is then extracted from the event message. The event handler will use the process information, such as process ID and attribute mapping, to launch a FileNet BPM process.

According to the scenario, Figure 7 illustrates the process status after the process was launched from the event handler. Because the customer service representative entered \$4000 for the claim amount, this claim is considered as a large claim. Therefore, the AutoInsuranceClaim process moves automatically from the Start step to the ReviewLargeClaim step. A check icon on the upper-left corner of the Start step indicates that this step is complete. A sand clock on top of the ReviewLargeClaim step shows that this step is in progress. By looking at the process parameters in the ReviewLargeClaim step in Figure 8, the claimAmount field holds 4000.0 and the claimID field holds CL1240533879046. This is consistent with the data in the event message.

Figure 7. Process status during execution

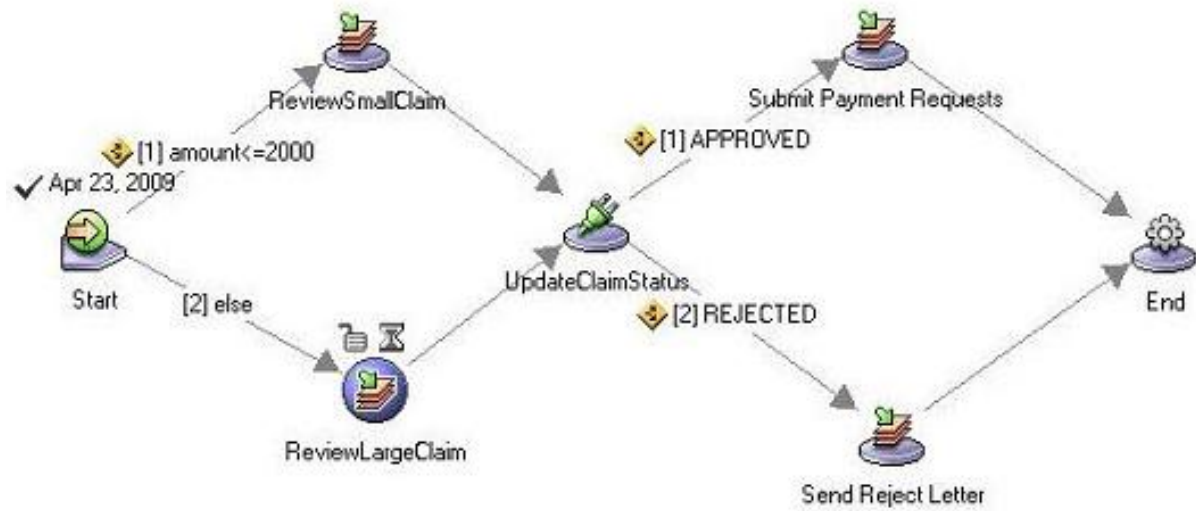


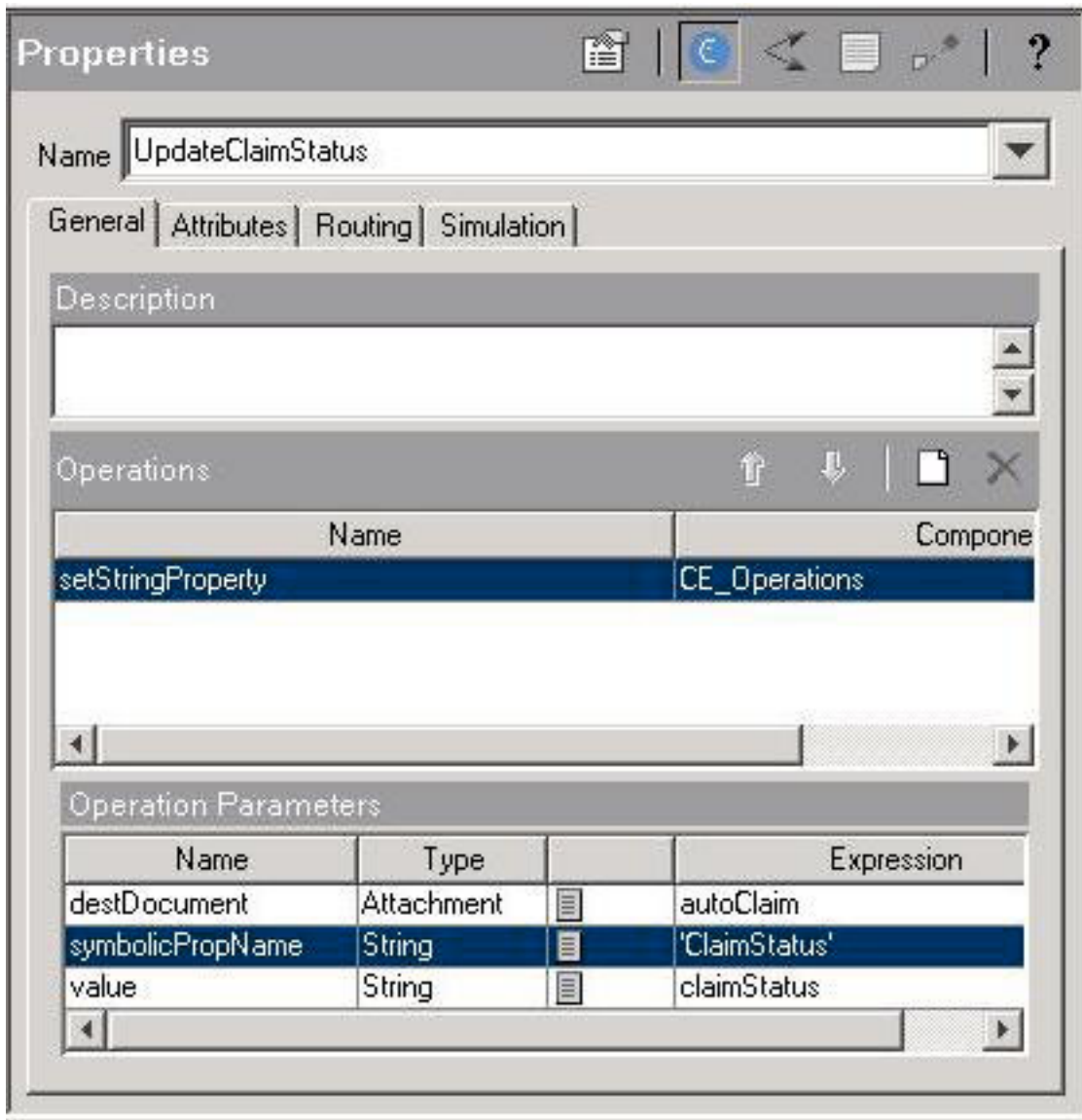
Figure 8. Process parameters during execution

| Workflow | Step | Route |
|--|------------------|-----------------------|
| Name: | ReviewLargeClaim | |
| Occurrence: | 1 | |
| Queue: | AutoClaim | |
| Instructions | | |
| View by: | | |
| <input checked="" type="radio"/> Step <input type="radio"/> Participant [AutoClaim] | | |
| Parameters | | Assignments |
| General | Fields | Attachments Routing |
| | | Workflow Group |
| Name | Type | |
| claimAmount | Float | 4,000.0 |
| claimID | String | CL1240533879046 |
| claimStatus | String | |

Callback interface

A callback interface is available to allow a BPM process to interact with Content Manager. The Component Integrator in BPM supports an implementation of this callback interface, called Content Extended Operations. An implementation of this Content Extended Operations interface is provided with Content Manager, which calls CM8 API to support document operations on the content in the repository.

Figure 9. Update the claim status



Continuing the scenario, now the auto claim is waiting for the review at the ReviewLargeStep step as shown in Figure 7. A supervisor conducts a claim review with an assessment. A completion of the review moves the process to the

UpdateClaimStatus step. Figure 9 illustrates the detail of callback operations in the UpdateClaimStatus step. Based on the review result of the supervisor, the value of the claimStatus parameter is given. The ClaimStatus attribute of the document in CM8 repository will be updated accordingly.

The launch of a FileNet BPM process "AutoInsuranceClaim" was triggered by a document creation operation in the CM8 repository. The update operation modified the "ClaimStatus" attribute of the document in the CM8 repository through a callback interface. This scenario demonstrates a two-way interaction between CM8 and FileNet BPM based on the event framework.

Conclusion

This article provides an overview of an event framework which enables the integration of Content Manager with FileNet Business Process Manager. We illustrated the architecture of the event framework and provided a scenario to guide you through various aspects of the integration, including event subscription, event monitoring, event processing and callback interface. In conclusion, this integration combines a scalable content repository with comprehensive business process management technology in support of the development of better business solutions in the enterprise content management area.

Resources

Learn

- [FileNet Business Process Manager](#): Learn more about FileNet BPM at ibm.com.
- [IBM Content Manager](#): Learn more about IBM Content Manager products at ibm.com.
- In the [ECM Zone on developerWorks](#): Find developer resources, tutorials, and articles for Enterprise Content Management software.
- Browse the [technology bookstore](#) for books on these and other technical topics.

Get products and technologies

- Download [IBM product evaluation versions](#) or [explore the online trials in the IBM SOA Sandbox](#) and get your hands on application development tools and middleware products from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

Discuss

- [Participate in the discussion forum for this content.](#)
- Check out [developerWorks blogs](#) and get involved in the [developerWorks community](#).

About the authors

Alan Yaung

Alan Yaung has extensive experiences in workflow and content management related products. He is the lead developer of the event monitor and the event handler for the integration of IBM Content Manager Version 8 and FileNet Business Process Manager.

Mimi Vo

Mimi Vo is a development architect for IBM Content Manager Version 8. She is the lead architect for the integration of IBM Content Manager Version 8 and FileNet Business Process Manager.