Java CAPS 6
Using JCA, Note 7

Batch Inbound-Triggered JCA (Non)Transacitonality

Michael Czapski, July 2008

Table of Contents

1 Introduction

In Note 6 we explored the transactional behavior of a JCA MDB invoked by a JMS Adapter and orchestrating an Oracle JCA Adapter (a transactional end point) and a Batch Local File JCA Adapter (a non-transactional end point). In this Note let's explore the issue of transactionality of a JCA Message-Driven Bean invoked by a non-transactional end point.

Let's take the example from the "*Java CAPS Basics*: *Implementing Common EAI Patterns* Companion CD" book, ISBN: 0-13-713071-6, Chapter 11 "Scalability and Resilience", Section 11.2 "Exception Handling", subsection 11.2.1 "Exceptions in Java Collaborations", 11.2.1.2 "Other Java Collaborations". The book from which this section comes is available on the Companion CD. Let's re-work this example using Java CAPS 6 JCA Adapters.

2 Solution outline

This example illustrates exception processing behaviour involving a Batch Inbound Adapter-triggered JCA Message-Driven Bean.

The Batch Inbound Adapter is designed to poll a directory for a file. When it finds a file that matches the name, or the name pattern, it immediately renames the fie by prepending a GUID to the original name then triggers the MDB and passes to it the original name of the file, the current name of the file and the directory in which the file was found. This behavior prevents other possible file pollers from getting hold of the file and gives the first comer exclusive access to the file. The MDB is designed to take the name of the file as given and rename the file to the original name with the suffix "~.in" appended, to indicate the file was read and processed. To do this the MDB will use the Batch Local File JCA Adapter's capability to post-process the file by renaming or deleting it. To explore the transactional behavior the MDB will explicitly throw an exception after it renames the file.

3 Connection Pools and JNDI Resources

The MDB will use a Batch Inbound JCA Adapter and a Batch Local File JCA Adapter.

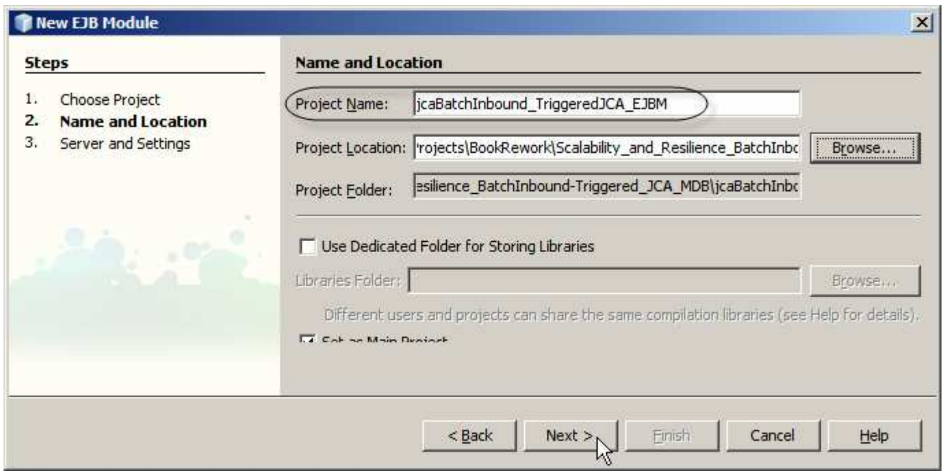
Configuring the Batch Inbound Adapter's _does not_ require Connection Pool or related resources. All of the configuration information that this Adapter requires is provided through the JCA MDB creation wizard dialog boxes and is changeable, after the MDB is created, through the Java Collaborations node of the EJB Project where the JCA MDB is created.

We must have a Connection Pool to configure the Batch Local File JCA Adapter, whether statically or dynamically. Because we will be configuring the Adapter dynamically we don't have to have a Connection Pool just for this solution. We can re-use any Connection Pool created for a Batch Local File Adapter. One created for Note 6 in this series would do just fine. In fact, this is what we will do in this Note. We will assume that the Batch Local File Adapter Connection Pool, "BatchLocal-c/temp/jc6jca/jmstriggeredjca_nn.dat", already exists and let's use it. The related JNDI Name reference, "jndi-BatchLocal-c/temp/jc6jca/jmstriggeredjca_nn.dat", also is assumed to exist.

4 Project Group and Project

As I am in a habit of doing, let's create a Project Group to contain the projects that will form part of this solution. Let's call this project group Scalability_and_Resilience_BatchInbound-Triggered_JCA_MDB.

In the newly created project group let's create an Enterprise -> EJB Module project called jcaBatchInbound_TriggeredJCA_EJBM. In this project we will create all other artefacts. Figure 4-1 illustrates this.



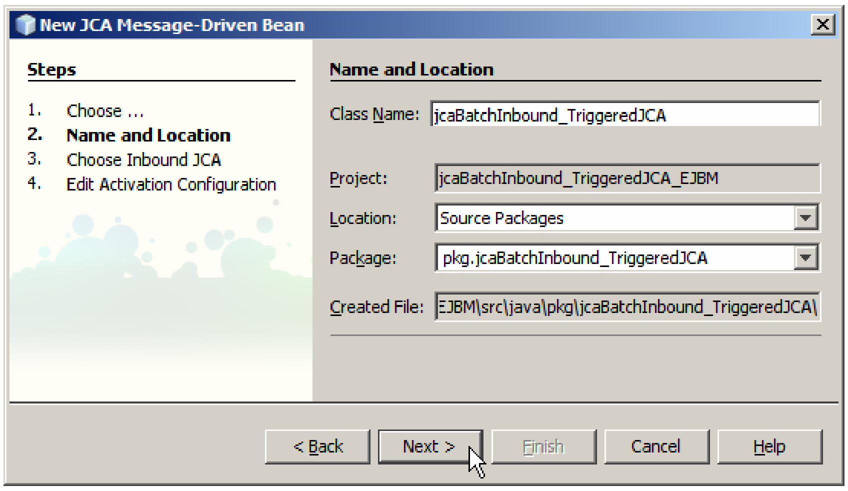
Project Name: jcaBatchInbound_TriggeredJCA_EJBM
Project Location: 'rojects\BookRework\Scalability_and_Resilience_BatchInbc
Project Folder: esilience_BatchInbound-Triggered_JCA_MDB\jcaBatchInbc
☐ Use Dedicated Folder for Storing Libraries
Libraries Folder:
Different users and projects can share the same compilation libraries (see Help for details).
☑ Set as Main Project

Figure 4-1 Create and name the Enterprise -> EJB Module project

5 MDB Logic

We will develop the Message Driven Bean, jcaBatchInbound_TriggeredJCA, a step at a time, with illustrations following.

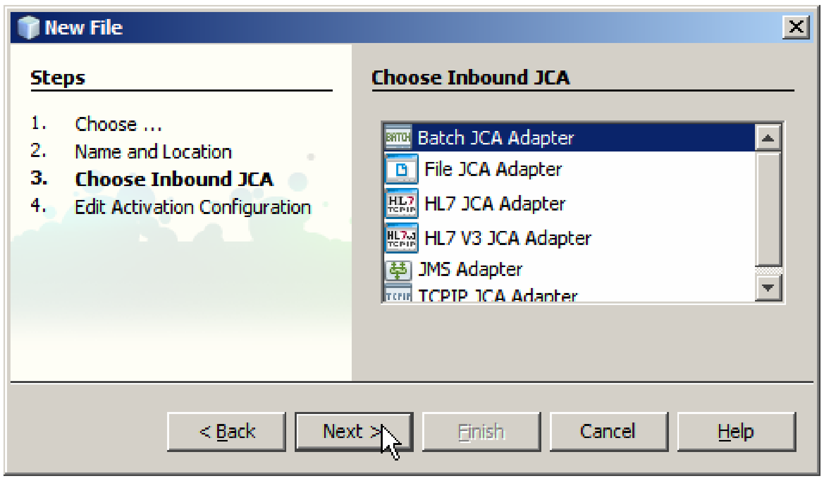
Let's start by creating a JCA MDB as shown in Figures 5-1 through 5-4.



Name and Location

Class Name:	jcaBatchInbound_TriggeredJCA
Project:	jcaBatchInbound_TriggeredJCA_EJBM
Location:	Source Packages
Package:	pkg.jcaBatchInbound_TriggeredJCA
Created File:	EJBM\src\java\pkg\jcaBatchInbound_TriggeredJCA\

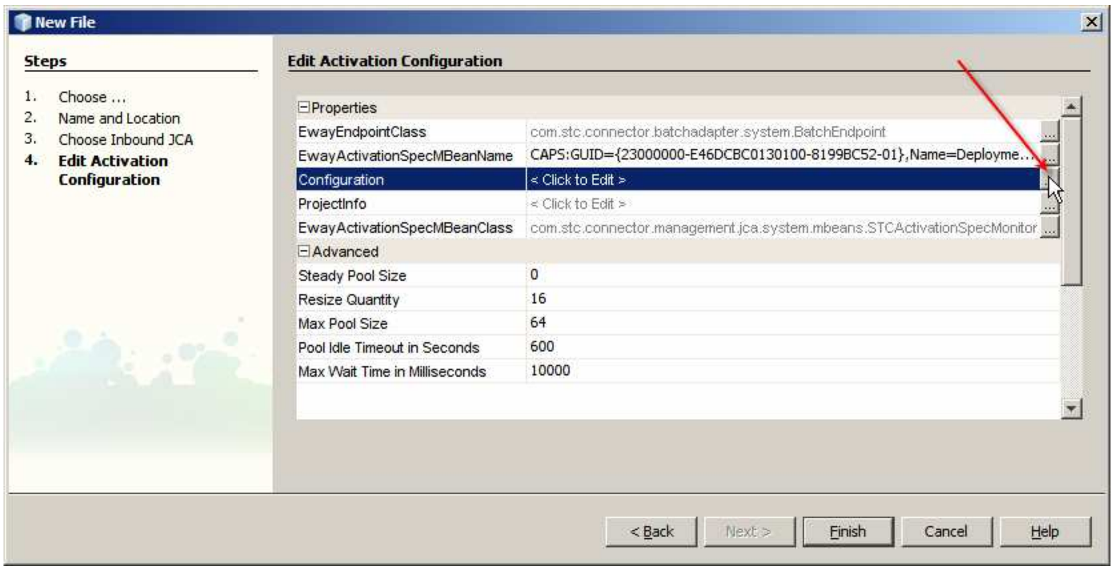
Figure 5-1 Name the JCA MDB



Choose Inbound JCA

- Batch JCA Adapter
- File JCA Adapter
- HL7 JCA Adapter
- HL7 V3 JCA Adapter
- JMS Adapter
- TCPIP JCA Adapter

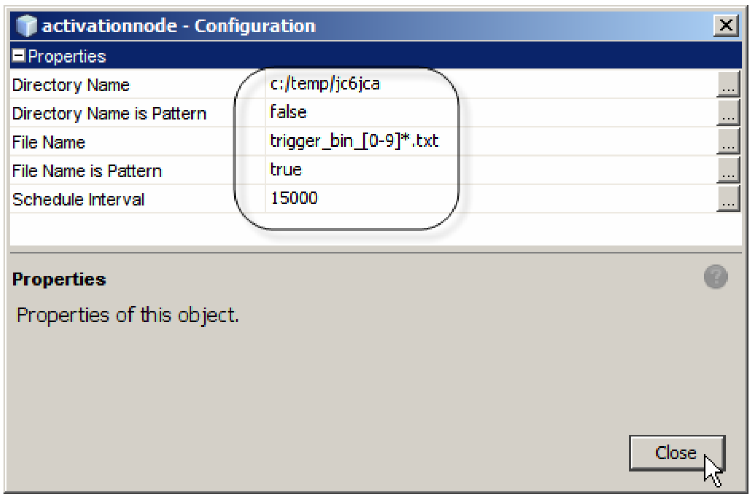
Figure 5-2 Choose the Batch Adapter



Edit Activation Configuration

⊟Properties	
EwayEndpointClass	com.stc.connector.batchadapter.system.BatchEndpoint
EwayActivationSpecMBeanName	CAPS:GUID={23000000-E46DCBC0130100-8199BC52-01},Name=Deployme...
Configuration	< Click to Edit >
ProjectInfo	< Click to Edit >
EwayActivationSpecMBeanClass	com.stc.connector.management.jca.system.mbeans.STCActivationSpecMonitor
⊟Advanced	
Steady Pool Size	0
Resize Quantity	16
Max Pool Size	64
Pool Idle Timeout in Seconds	600
Max Wait Time in Milliseconds	10000

Figure 5-3 Choose to edit Activation Configuration


activationnode - Configuration

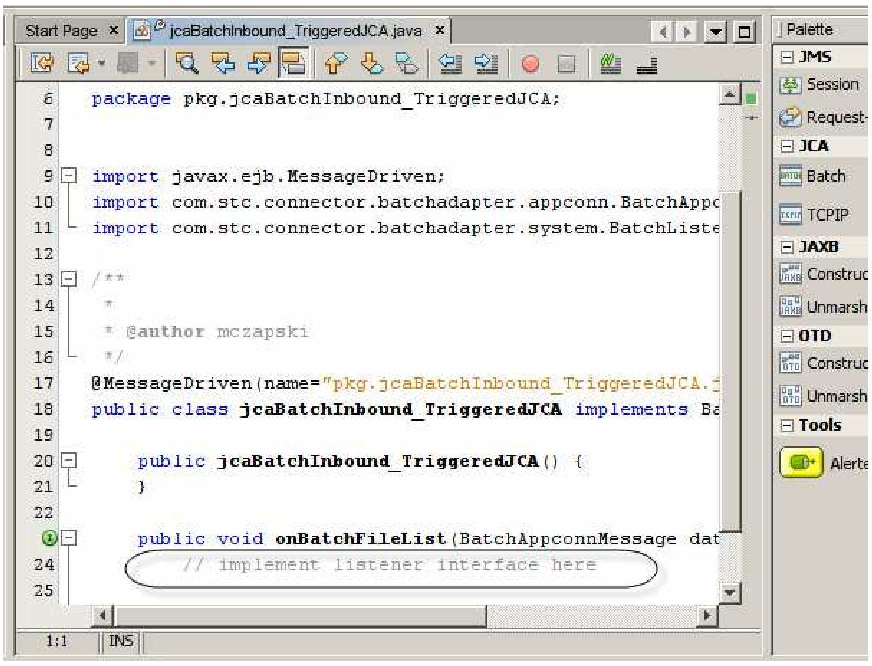
Properties

Directory Name	c:/temp/jc6jca
Directory Name is Pattern	false
File Name	trigger_bin_[0-9]*.txt
File Name is Pattern	true
Schedule Interval	15000

Properties

Properties of this object.

Figure 5-4 Configure File and Directory names, Close and Finish

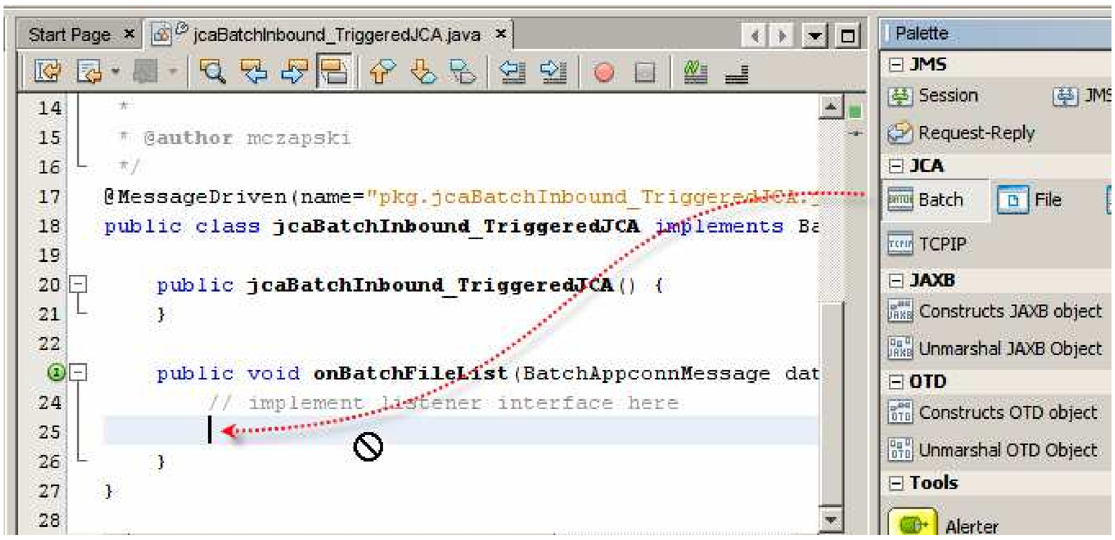


```
package pkg.jcaBatchInbound_TriggeredJCA;


import javax.ejb.MessageDriven;
import com.stc.connector.batchadapter.appconn.BatchAppc
import com.stc.connector.batchadapter.system.BatchListe

/**
 *
 * @author mczapski
 */
@MessageDriven(name="pkg.jcaBatchInbound_TriggeredJCA.j
public class jcaBatchInbound_TriggeredJCA implements Ba

    public jcaBatchInbound_TriggeredJCA() {
    }

    public void onBatchFileList(BatchAppconnMessage dat
        // implement listener interface here

```

Figure 5-5 Boilerplate JCA MDB code

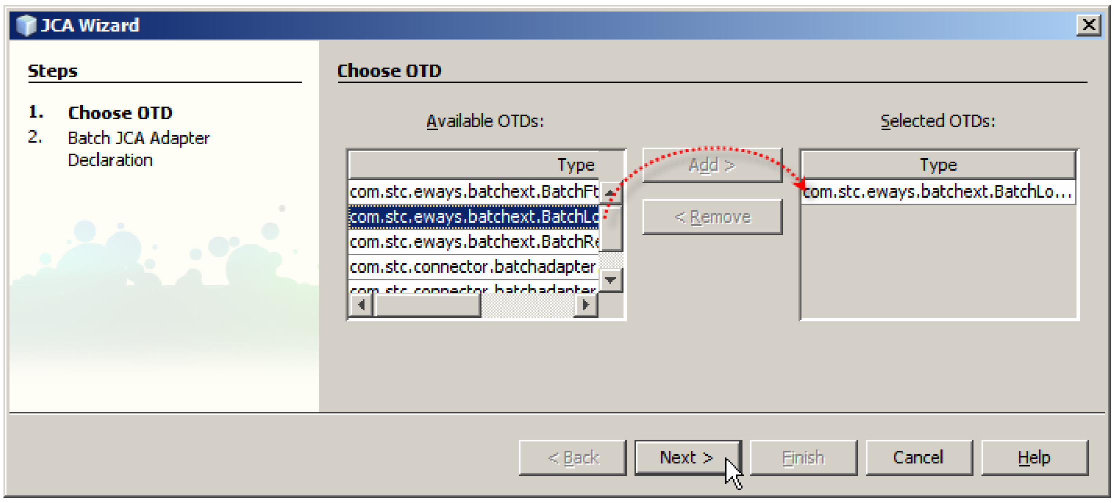
Once the wizard completes the JCA MDB code will be available for editing – see Figure 5-5.

Let’s now add the Batch JCA. Figures 5-6 through 5-8 illustrate the process.



```
 *
 * @author mczapski
 */
@MessageDriven(name="pkg.jcaBatchInbound_TriggeredJCA:
public class jcaBatchInbound_TriggeredJCA implements Ba

    public jcaBatchInbound_TriggeredJCA() {
    }

    public void onBatchFileList(BatchAppconnMessage dat
        // implement listener interface here

    }
}
```

Figure 5-6Drag the Batch JCA Adapter to the source window



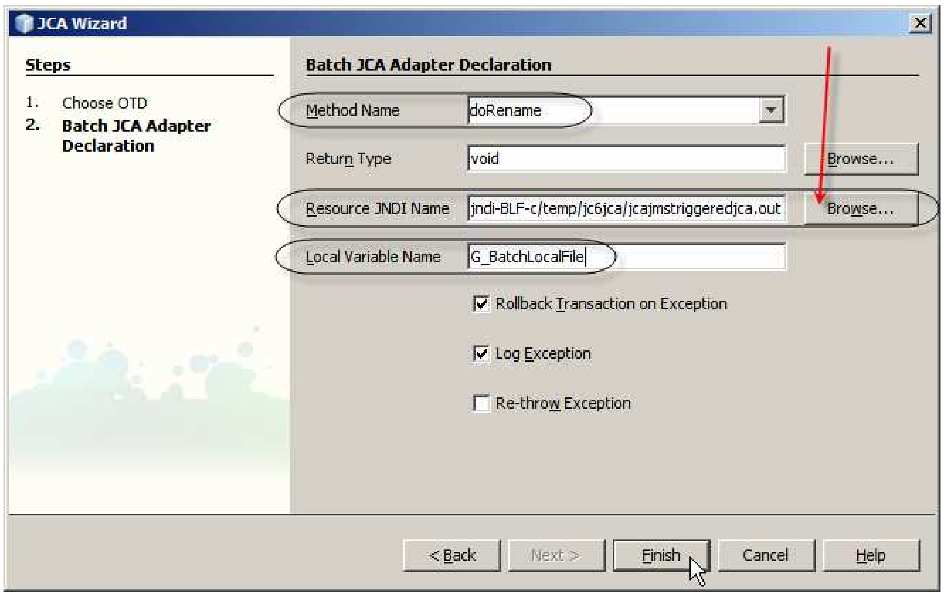
Available OTDs:

Type
com.stc.eways.batchext.BatchFt
com.stc.eways.batchext.BatchLc
com.stc.eways.batchext.BatchRe
com.stc.connector.batchadapter
com.stc.connector.batchadapter

Selected OTDs:

Type
com.stc.eways.batchext.BatchLo...

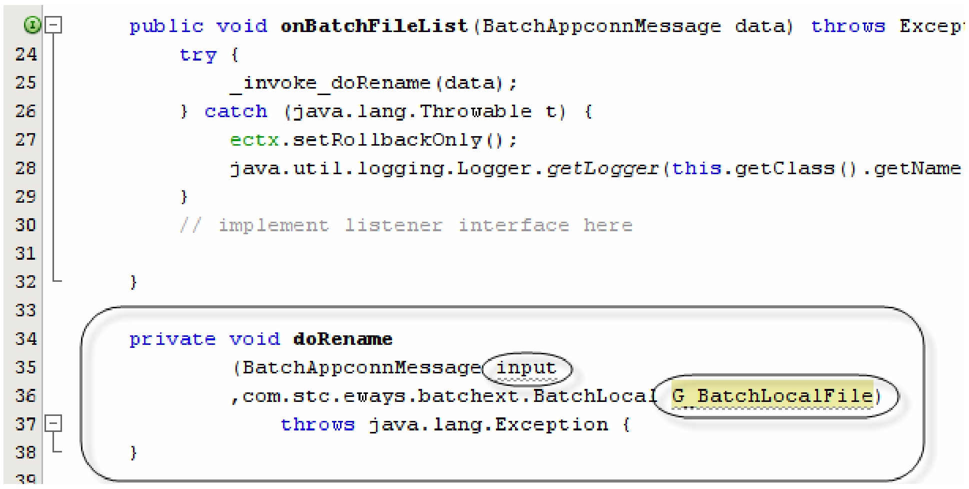
Figure 5-7 Choose the BatchLocal OTD and click Next



Method Name: doRename
Return Type: void
Resource JNDI Name: jndi-BLF-c/temp/jc6jca/jcajmstriggeredjca.out
Local Variable Name: G_BatchLocalFile
[✓] Rollback Transaction on Exception
[✓] Log Exception
[ ] Re-throw Exception

Figure 5-8 Enter new method name, choose JNDI name and provide the name for the variable

Notice that the wizard added a new method, doRename, and added some boilerplate code that causes that method to be invoked. Our "creative code" will go into this new method. Let's rename the arguments to the doRename method to make it appear like the corresponding "receive" method in a Batch Inbound-triggered JCD would in 5.1. Let's change the name "data" to "input" and "G_BatchLocakFileOTD" to "G_BatchLocalFile". Figure 5-9 show the method and its signature.



```
    public void onBatchFileList(BatchAppconnMessage data) throws Excep
        try {
            _invoke_doRename(data);
        } catch (java.lang.Throwable t) {
            ectx.setRollbackOnly();
            java.util.logging.Logger.getLogger(this.getClass().getName
        }
        // implement listener interface here

    }

    private void doRename
            (BatchAppconnMessage input
            ,com.stc.eways.batchext.BatchLocal G_BatchLocalFile)
                throws java.lang.Exception {
    }
```

Figure 5-9 receive method signature with all JCA Adapters included

To complete the MDB let's add the slab of code from Listing 5-1 as the doRename method body.

Listing 5-1

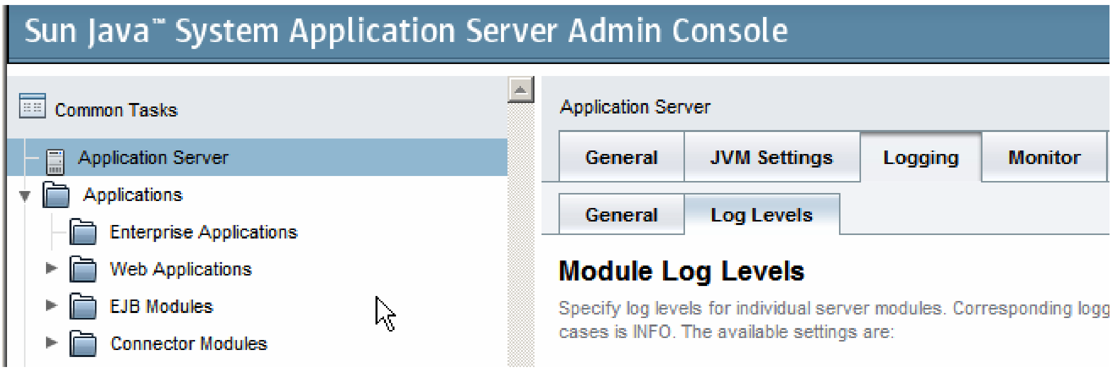
```
G_BatchLocalFile.getConfiguration().setTargetDirectoryName(input.getPathDirName());
G_BatchLocalFile.getConfiguration().setTargetDirectoryNameIsPattern(false);
G_BatchLocalFile.getConfiguration().setTargetFileName(input.getGUIDFileName());
G_BatchLocalFile.getConfiguration().setTargetFileNameIsPattern(false);

G_BatchLocalFile.getConfiguration().setPostDirectoryName(input.getPathDirName());
G_BatchLocalFile.getConfiguration().setPostDirectoryNameIsPattern(false);
G_BatchLocalFile.getConfiguration().setPostFileName(input.getOriginalFileName()
                                                    + ".~in");
G_BatchLocalFile.getConfiguration().setPostFileNameIsPattern(false);
G_BatchLocalFile.getConfiguration().setPostTransferCommand("Rename");

G_BatchLocalFile.getConfiguration().setPreTransferCommand("None");

G_BatchLocalFile.getClient().get();

(Logger.getLogger(this.getClass().getName())).warning
       ("\n===>>> Batch Inbound message with file details"
        + "\nGUID File Name: " + input.getGUIDFileName()
        + "\nOriginal File Name: " + input.getOriginalFileName()
        + "\nPath Name: " + input.getPathDirName());

throw new Exception("I don't wish to do this ...");
```

Note what is happening. The input argument provides the components of the name of the file that was found and that triggered this MDB. The original name, the current name and the directory. We use these values to dynamically set configuration of the Batch Local File to read the GUID-named file in the specified directory then to rename it to the original name with the literal "~.in" appended. The GET operation of the Client() perform the actual 'transfer'.

Once the transfer is completed we throw an exception to see what the container will do.

Build and deploy the project.

If you are interested in seeing what the MDB does at runtime enable verbose logging for selected logger categories. For example set the following using the Application Server Admin Console: Application Server -> Logging -> Log Levels, see Figures 5-10.



Module Log Levels

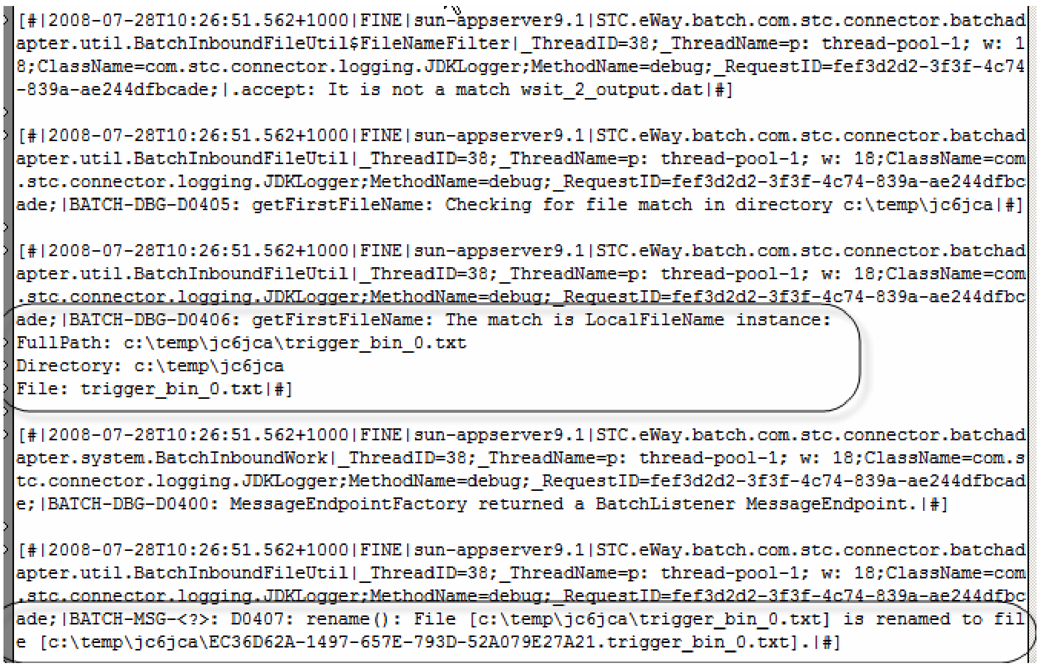
Specify log levels for individual server modules. Corresponding logg
cases is INFO. The available settings are:

Figure 5-10 Locating logging configuration

STC.eWay.batch	FINEST

6 Exercise the solution

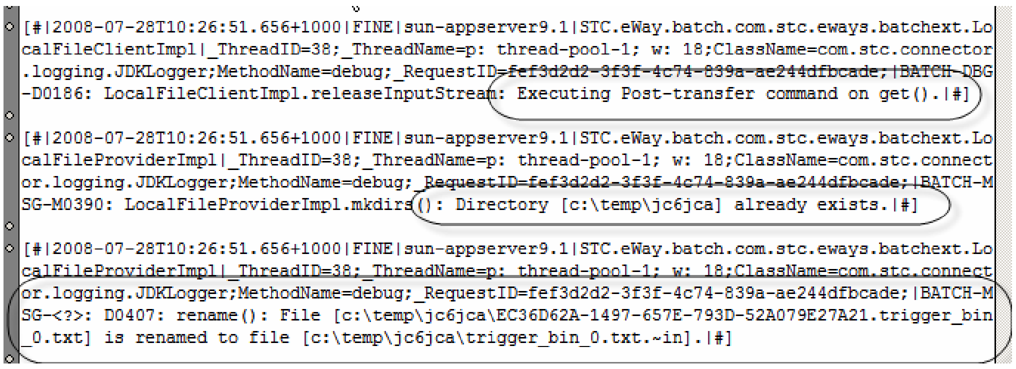
Let's create a file named "trigger_bin_0.txt.~in" in "c:/temp/jc6jca". The directory comes from the Batch Inbound configuration. The name is that which we specified in the Batch Inbound configuration but with the ".~in" appended. This is to allow us time to add some content to the file before the Batch Inbound JCA Adapter gets a chance to process it. Let's add some content and rename the file by removing the ".~in" suffix. Batch Inbound will find the file on its next poll, rename it and trigger the MDB. Figure 6-1 show the log extract to that effect.



```
[#|2008-07-28T10:26:51.562+1000|FINE|sun-appserver9.1|STC.eWay.batch.com.stc.connector.batchad
apter.util.BatchInboundFileUtil$FileNameFilter|_ThreadID=38;_ThreadName=p: thread-pool-1; w: 1
8;ClassName=com.stc.connector.logging.JDKLogger;MethodName=debug;_RequestID=fef3d2d2-3f3f-4c74
-839a-ae244dfbcade;|.accept: It is not a match wsit_2_output.dat|#]

[#|2008-07-28T10:26:51.562+1000|FINE|sun-appserver9.1|STC.eWay.batch.com.stc.connector.batchad
apter.util.BatchInboundFileUtil|_ThreadID=38;_ThreadName=p: thread-pool-1; w: 18;ClassName=com
.stc.connector.logging.JDKLogger;MethodName=debug;_RequestID=fef3d2d2-3f3f-4c74-839a-ae244dfbc
ade;|BATCH-DBG-D0405: getFirstFileName: Checking for file match in directory c:\temp\jc6jca|#]

[#|2008-07-28T10:26:51.562+1000|FINE|sun-appserver9.1|STC.eWay.batch.com.stc.connector.batchad
apter.util.BatchInboundFileUtil|_ThreadID=38;_ThreadName=p: thread-pool-1; w: 18;ClassName=com
.stc.connector.logging.JDKLogger;MethodName=debug;_RequestID=fef3d2d2-3f3f-4c74-839a-ae244dfbc
ade;|BATCH-DBG-D0406: getFirstFileName: The match is LocalFileName instance:
FullPath: c:\temp\jc6jca\trigger_bin_0.txt
Directory: c:\temp\jc6jca
File: trigger_bin_0.txt|#]

[#|2008-07-28T10:26:51.562+1000|FINE|sun-appserver9.1|STC.eWay.batch.com.stc.connector.batchad
apter.system.BatchInboundWork|_ThreadID=38;_ThreadName=p: thread-pool-1; w: 18;ClassName=com.s
tc.connector.logging.JDKLogger;MethodName=debug;_RequestID=fef3d2d2-3f3f-4c74-839a-ae244dfbcad
e;|BATCH-DBG-D0400: MessageEndpointFactory returned a BatchListener MessageEndpoint.|#]

[#|2008-07-28T10:26:51.562+1000|FINE|sun-appserver9.1|STC.eWay.batch.com.stc.connector.batchad
apter.util.BatchInboundFileUtil|_ThreadID=38;_ThreadName=p: thread-pool-1; w: 18;ClassName=com
.stc.connector.logging.JDKLogger;MethodName=debug;_RequestID=fef3d2d2-3f3f-4c74-839a-ae244dfbc
ade;|BATCH-MSG-<?>: D0407: rename(): File [c:\temp\jc6jca\trigger_bin_0.txt] is renamed to fil
e [c:\temp\jc6jca\EC36D62A-1497-657E-793D-52A079E27A21.trigger_bin_0.txt].|#]
```

Figure 6-1 Batch Inbound found the file and renamed it

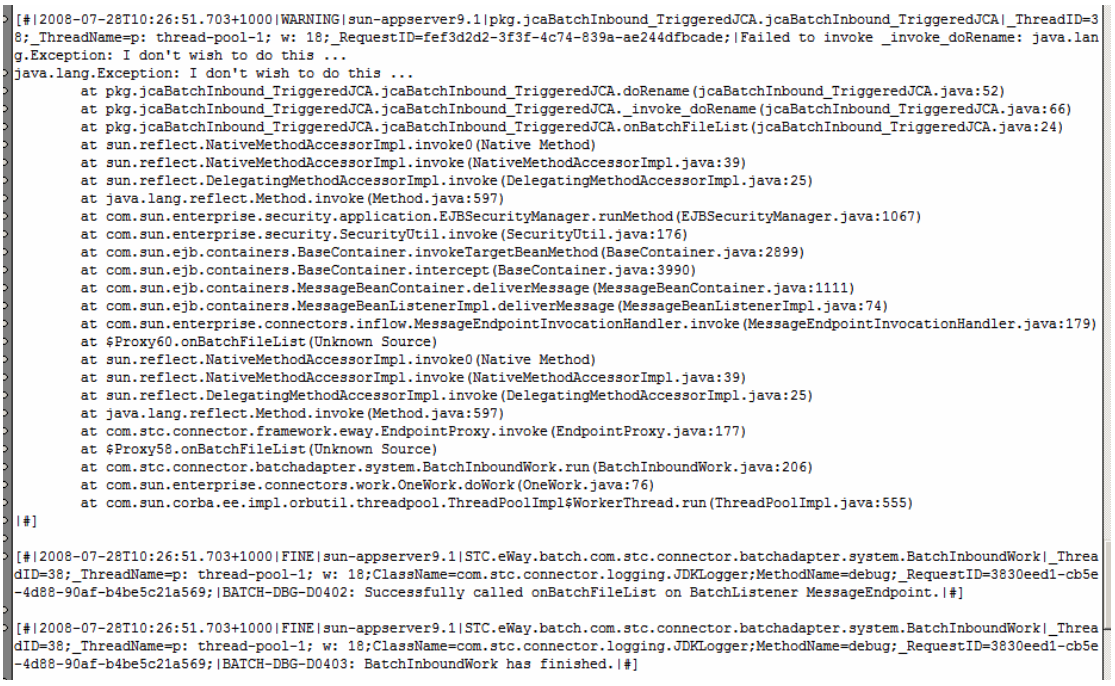
Looking at the log further reveals trace of post-processing activities in which the file is renamed. Figure 6-2 shows this.



```
[#|2008-07-28T10:26:51.656+1000|FINE|sun-appserver9.1|STC.eWay.batch.com.stc.eways.batchext.Lo
calFileClientImpl|_ThreadID=38;_ThreadName=p: thread-pool-1; w: 18;ClassName=com.stc.connector
.logging.JDKLogger;MethodName=debug;_RequestID=fef3d2d2-3f3f-4c74-839a-ae244dfbcade;|BATCH-DBG
-D0186: LocalFileClientImpl.releaseInputStream: Executing Post-transfer command on get().|#]

[#|2008-07-28T10:26:51.656+1000|FINE|sun-appserver9.1|STC.eWay.batch.com.stc.eways.batchext.Lo
calFileProviderImpl|_ThreadID=38;_ThreadName=p: thread-pool-1; w: 18;ClassName=com.stc.connect
or.logging.JDKLogger;MethodName=debug;_RequestID=fef3d2d2-3f3f-4c74-839a-ae244dfbcade;|BATCH-M
SG-M0390: LocalFileProviderImpl.mkdirs(): Directory [c:\temp\jc6jca] already exists.|#]

[#|2008-07-28T10:26:51.656+1000|FINE|sun-appserver9.1|STC.eWay.batch.com.stc.eways.batchext.Lo
calFileProviderImpl|_ThreadID=38;_ThreadName=p: thread-pool-1; w: 18;ClassName=com.stc.connect
or.logging.JDKLogger;MethodName=debug;_RequestID=fef3d2d2-3f3f-4c74-839a-ae244dfbcade;|BATCH-M
SG-<?>: D0407: rename(): File [c:\temp\jc6jca\EC36D62A-1497-657E-793D-52A079E27A21.trigger_bin
_0.txt] is renamed to file [c:\temp\jc6jca\trigger_bin_0.txt.~in].|#]
```

Figure 6-2 Post-processing activities

Finally, the log shows the exception being thrown, Figure 6-3.



```
[#|2008-07-28T10:26:51.703+1000|WARNING|sun-appserver9.1|pkg.jcaBatchInbound_TriggeredJCA.jcaBatchInbound_TriggeredJCA|_ThreadID=3
8;_ThreadName=p: thread-pool-1; w: 18;_RequestID=fef3d2d2-3f3f-4c74-839a-ae244dfbcade;|Failed to invoke _invoke_doRename: java.lan
g.Exception: I don't wish to do this ...
java.lang.Exception: I don't wish to do this ...
	at pkg.jcaBatchInbound_TriggeredJCA.jcaBatchInbound_TriggeredJCA.doRename(jcaBatchInbound_TriggeredJCA.java:52)
	at pkg.jcaBatchInbound_TriggeredJCA.jcaBatchInbound_TriggeredJCA._invoke_doRename(jcaBatchInbound_TriggeredJCA.java:66)
	at pkg.jcaBatchInbound_TriggeredJCA.jcaBatchInbound_TriggeredJCA.onBatchFileList(jcaBatchInbound_TriggeredJCA.java:24)
	at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
	at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
	at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
	at java.lang.reflect.Method.invoke(Method.java:597)
	at com.sun.enterprise.security.application.EJBSecurityManager.runMethod(EJBSecurityManager.java:1067)
	at com.sun.enterprise.security.SecurityUtil.invoke(SecurityUtil.java:176)
	at com.sun.ejb.containers.BaseContainer.invokeTargetBeanMethod(BaseContainer.java:2899)
	at com.sun.ejb.containers.BaseContainer.intercept(BaseContainer.java:3990)
	at com.sun.ejb.containers.MessageBeanContainer.deliverMessage(MessageBeanContainer.java:1111)
	at com.sun.ejb.containers.MessageBeanListenerImpl.deliverMessage(MessageBeanListenerImpl.java:74)
	at com.sun.enterprise.connectors.inflow.MessageEndpointInvocationHandler.invoke(MessageEndpointInvocationHandler.java:179)
	at $Proxy60.onBatchFileList(Unknown Source)
	at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
	at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
	at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
	at java.lang.reflect.Method.invoke(Method.java:597)
	at com.stc.connector.framework.eway.EndpointProxy.invoke(EndpointProxy.java:177)
	at $Proxy58.onBatchFileList(Unknown Source)
	at com.stc.connector.batchadapter.system.BatchInboundWork.run(BatchInboundWork.java:206)
	at com.sun.enterprise.connectors.work.OneWork.doWork(OneWork.java:76)
	at com.sun.corba.ee.impl.orbutil.threadpool.ThreadPoolImpl$WorkerThread.run(ThreadPoolImpl.java:555)
|#]

[#|2008-07-28T10:26:51.703+1000|FINE|sun-appserver9.1|STC.eWay.batch.com.stc.connector.batchadapter.system.BatchInboundWork|_Threa
dID=38;_ThreadName=p: thread-pool-1; w: 18;ClassName=com.stc.connector.logging.JDKLogger;MethodName=debug;_RequestID=3830eed1-cb5e
-4d88-90af-b4be5c21a569;|BATCH-DBG-D0402: Successfully called onBatchFileList on BatchListener MessageEndpoint.|#]

[#|2008-07-28T10:26:51.703+1000|FINE|sun-appserver9.1|STC.eWay.batch.com.stc.connector.batchadapter.system.BatchInboundWork|_Threa
dID=38;_ThreadName=p: thread-pool-1; w: 18;ClassName=com.stc.connector.logging.JDKLogger;MethodName=debug;_RequestID=3830eed1-cb5e
-4d88-90af-b4be5c21a569;|BATCH-DBG-D0403: BatchInboundWork has finished.|#]
```

Figure 6-3 Trace of the exception

Note that the file was renamed explicitly in the MDB but it was not renamed back to the original name on exception, as one would expect if the Batch Inbound was a transactional resource capable of rollback. Had the Batch Local File Adapter not been used to rename the file, the GUID-based file name, to which the Batch Inbound renamed the file when it found it, would have remained. In EAI speak, the message would have been consumed and, because the exception did not cause rollback, it would have been lost.

7 Conclusion

One lesson from this example is to design integration solutions so as to minimize message loss. In this case the MDB triggered by the Batch Inbound JCA Adapter should have passed the name of the file to a component that could deal with exceptions and could successfully roll back a transaction on exception. Instead of manipulating the file (renaming it) the MDB could send the file name and directory details to a JMS Queue. A component downstream would process the file. Exceptions in that component would cause JMS rollback and redevliery handling would be able to take care of inability to process the file without message loss.