WebSphere. Monitor

Version 6.0.2

File Event Emitter

Contents

1 Introduction **3**

2 Design overview **4**
2.1 Event framework 4
2.2 File Event Emitter 5

3 Configuring and deploying the Emitter application **7**
3.1 Create an IBM Cloudscape™ database for the Monitor Server Scheduler 7
3.2 Create a data source for the File Emitter 8
3.3 Create a scheduler for the DB2 Emitter
3.4 Deploy the File Emitter enterprise application using the Administrative Console 8
3.5 Create the inbound, fault and archive folders to be used for event management 8
3.6. Restart the Monitor Server 8

4 Verifying the configuration and deployment of the Emitter application **9**

5 Importing and working with the source code projects **10**
5.1 Projects overview 11

6 Design details — creating a new event formatter to handle additional application data types **12**
6.1 FileEmitterFormatter.properties file 12
6.2 EventFormatter implementation 12

7 Working with CBEs and CEI — key concepts **14**
7.1 Creating a new or obtaining an existing EventFactory 14
7.2 Creating the new CommonBaseEvent 14
7.3 Setting mandatory fields 14
7.4 Creating an ExtendedDataElement and its children 14
7.5 Obtaining the EmitterFactory 15
7.6 Obtaining the Emitter from the EmitterFactory 15
7.7 Sending an Event 15

Figures

Figure 1 Event Emitter Framework 4
Figure 2 File Emitter 6

1 Introduction

The Sample File Event Emitter is a sample program written in Java™ that demonstrates how an enterprise information system (EIS) resource storing data pertaining to the state of a business can be instrumented to contribute to the overall monitoring of the activities of a business.

The main goal of the Sample File Event Emitter is to introduce the use of the libraries and application programming interfaces (APIs) provided by the Common Event Infrastructure (CEI) to generate and emit business events in the form of Common Base Events (CBEs). Common Base Events are the data packaging and format used by the IBM WebSphere® Business Monitor (Monitor) Server to propagate business events.

The Sample File Event Emitter is made available in two forms of packaging. One is the binary archive, which contains the precompiled Java 2 Platform, Enterprise Edition (J2EE) enterprise application, ready to be deployed to the Monitor Server. It also includes the accompanying documentation. The second is the source archive, which also includes the documentation and, additionally, contains the source code projects that can be imported into the IBM WebSphere Integration Developer (WID) integrated development environment (IDE) for browsing through the source code and making custom changes to the emitter and its configuration options.

The Sample File Event Emitter was developed and tested using the following environment:

- *Microsoft® Windows® XP Professional SP2*
- *IBM DB2® Universal Database™ (UDB) Enterprise Server Edition v8.1.13.193*
- *WebSphere Business Monitor (Monitor) Server v6.0.2*
- *WebSphere Integration Developer (WID) v6.0.2*

Note: The configuration and deployment information in this document describes the Sample File Event Emitter being deployed to the Monitor Server itself (that is, to the same application server on which Monitor runs). The emitter can also be deployed to a separate application server and configured to emit its events to the Monitor Server's CEI Server (see section 3, "Configuring and deploying the Emitter application", for more information on this).

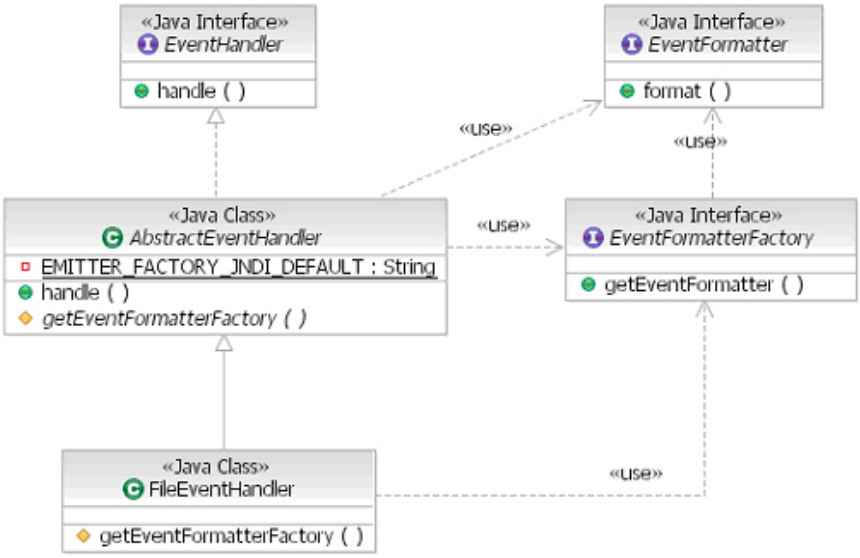
2 Design Overview

2.1 Event framework

The Sample DB2 Event Emitter, along with the other sample event emitters made available, is implemented around a common, simple emitter framework. Figure 1 depicts a class diagram of this framework. When an event in the enterprise back-end system is detected, the following flow is regulated by this common framework:

1. *Retrieve, from the EmitterFormatterFactory, an EventFormatter specific for the type of data being processed.*
2. *Invoke the EventFormatter to convert the input data to a CBE object.*
3. *Retrieve an emitter from an EmitterFactory to be used to send the event to the CEI Server.*
4. *Emit the CBE to the CEI Server.*

Figure 1
Event Emitter Framework



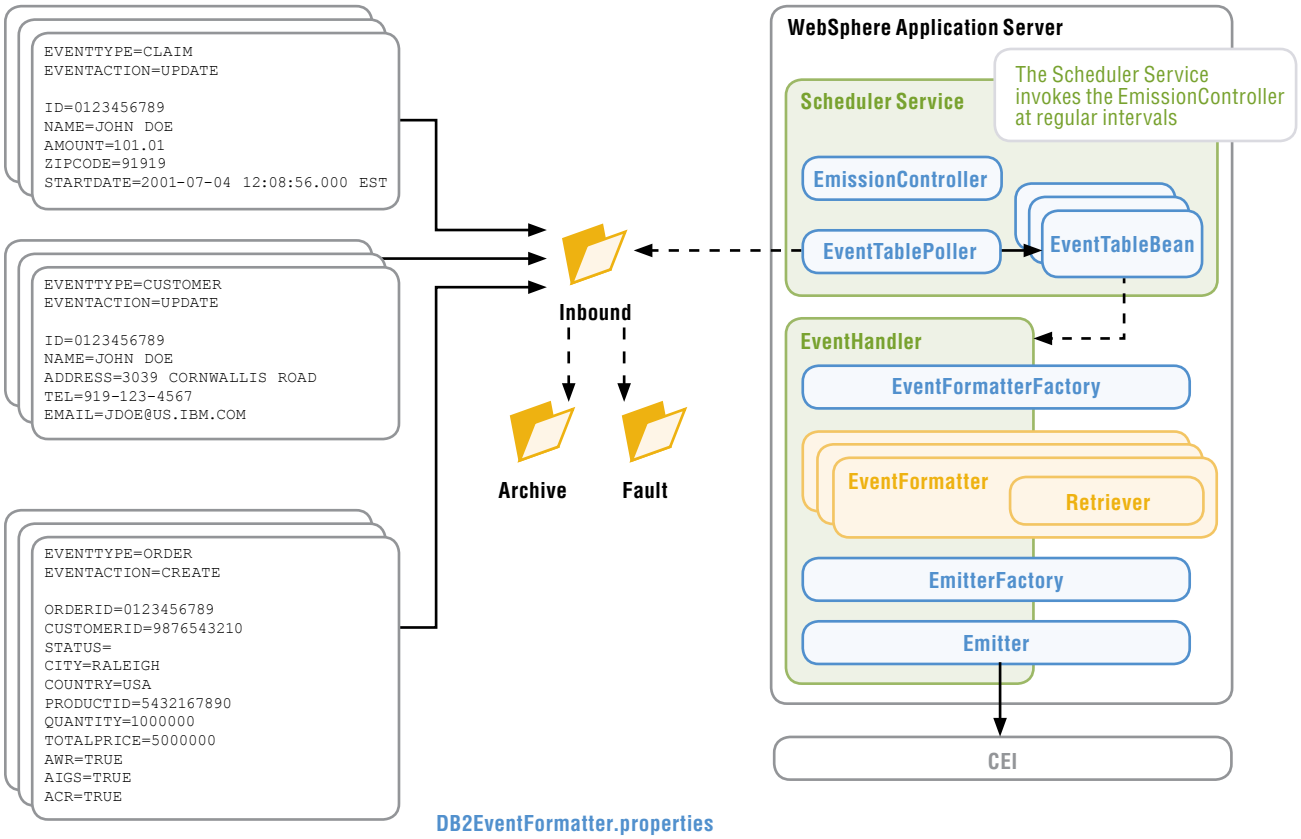
2.2 File Event Emitter

The Sample File Event Emitter is a J2EE enterprise application, which also implements the WebSphere TaskHandler interface, allowing it to handle invocations from a WebSphere Application Server Scheduler.The overall event retrieval and event emission flow is explained below (see Figure 2 for a graphical depiction of this flow):

1. *A file encapsulating information about a create, update, or delete event is placed in the inbound directory. This file contains information about what data type the event pertains to, the actual event type (that is, create, delete, or update) and the attributes and their associated values.*
2. *The Scheduler service in the Monitor Server invokes the EmissionController at a specified interval. The EmissionController calls EventFolderPoller. If any files are found in the inbound directory, EventFolderPoller populates and returns a collection of EventFileBean objects. If the collection of EventFileBean objects is not empty, EmissionController invokes the handle( ) method of EventHandler for each EventFileBean object to handle the emission steps.*
3. *EventHandler calls the EventFormatterFactory, which returns an EventFormatter object according to the application data type specified in the event file. The EventFormatter object transforms the EventFileBean object into a CommonBaseEvent object. Finally, the CEIEmitter sends the CommonBaseEvent object to CEI Event Server.*

The Sample File Event Emitter is designed so that it can be extended to handle processing of additional application data types. This is done by creating new classes that implement the EventFormatter interface (in the EmitterFW project) and "registering" application data type to event formatter associations by updating the FileEventFormatter.properties (in the FileEmitterImpl project). For more-extensive details on this see Section 6: Design details—creating a new event formatter to handle additional application data types.

Figure 2
File Emitter



DB2EventFormatter.properties

```
CUSTOMER=com.ibm.wbimonitor.samples.db2emitter.formatter.CUSTOMERFormatterImpl
ORDER=com.ibm.wbimonitor.samples.db2emitter.formatter.ORDERFormatterImpl
CLAIM=com.ibm.wbimonitor.samples.db2emitter.formatter.CLAIMFormatterImpl
```

3 Configuring and deploying the Emitter application

This section describes how to configure the Monitor Server to host the Sample File Event Emitter application. The following section, "Importing and working with the source code," describes how to import the source code into the WID IDE and introduce modifications, if needed.

The following default are values specified in the File Emitter's EJB deployment descriptor. These can be customized by editing the source projects in WID and exporting a new enterprise archive (EAR) file.

```
<env-entry-name>inboundEventsDirectory</env-entry-name>
<env-entry-value>C:\FileEmitter\inbound</env-entry-value>

<env-entry-name>archivedEventsDirectory</env-entry-name>
<env-entry-value>C:\FileEmitter\archive</env-entry-value>

<env-entry-name>faultDirectory</env-entry-name>
<env-entry-value>C:\FileEmitter\fault</env-entry-value>

<env-entry-name>emitterFactoryJNDI</env-entry-name>
<env-entry-value>iiop://localhost:2809/com/ibm/events/configura-
tion/emitter/Default</env-entry-value>

<env-entry-name>schedulerJNDI</env-entry-name>
<env-entry-value>sched/FolderPoller</env-entry-value>
```

Notes:

- The default emitter factory specified in the "emitterFactoryJNDI" environment entry refers to "localhost" because the emitter is being deployed to the same application server as the Monitor Server. If the emitter is deployed to its own application server, ensure that the hostname is that of the Monitor Server (where the CEI Server resides and to where the events will be emitted).
- The default emitter factory specified in the "emitterFactoryJNDI" environment entry uses port 2809. Check the Monitor Server's BOOTSTRAP_ADDRESS setting to ensure that is the correct port to use.

If any of these environment entries need to be modified, import the source projects into WID and make the necessary modifications to the EJB deployment descriptor. (See Section 5: Importing and working with the source code projects for instructions on how to import the source projects).

3.1 Create an IBM Cloudscape™ database for the Monitor Server Scheduler:

1. *In the ${WBI_INSTALL_ROOT}/cloudscape/bin/embedded directory, open the cview.bat file.*
2. *Select **File→ New → Database**.*
3. *In the Name field, type* `${WBI _ INSTALL _ ROOT}/cloudscape/databases/SKDLR`.
4. *Click **OK** to create the new database and then exit from the cview.bat file.*

3.2 Create a data source for the Monitor Server Scheduler:

1. *Open the Administrative Console.*
2. *Open the **Resources→ JDBC Providers** page, and set the scope to **Server**.*
3. *By default, there should be a Java Database Connectivity (JDBC) provider named **Cloudscape JDBC Provider (XA)**. Click it then select **Data sources** under the Additional Properties section.*
4. *Click **New** to create a new data source with the following properties:*

```
Name: SKDLR Datasource
Java Naming and Directory Interface (JNDI) name: jdbc/skdlr
Disable "Use this Data Source in container managed persistence
(CMP)"
Description: JDBC Datasource for SKDLR Database
Database name: ${WBI _ INSTALL _ ROOT}/cloudscape/databases/SKDLR
```

5. *Click **OK** and **Save**.*

3.3. Create a scheduler for the File Emitter:

1. *Open the Administrative Console.*
2. *Go to **Resources→ Schedulers** and set the scope to **Server**.*
3. *Click **New** to create a new scheduler with the following properties:*

```
Name: FolderPoller
JNDI name: sched/FolderPoller (This is the default name specified
as an environment entry in the EJB deployment descriptor.)
Description: Scheduler for File Sample Emitter's Inbound Folder
Poller
Data source JNDI name: jdbc/skdlr
Table prefix: FILEEMTR _
Poll interval: 30
Work managers: DefaultWorkManager
```

4. *Click **OK** and **Save**.*
5. *On the Schedulers page, select the newly created FolderPoller scheduler and click **Create tables**.*

3.4 Deploy the File Emitter enterprise application using the Administrative Console.

Accept all default values and save the configuration when deployment is completed.

3.5 Create the inbound, fault and archive folders to be used for event management.

The default folder locations that are specified as environment entries in the EJB deployment descriptor are: C:\FileEmitter\inbound, C:\FileEmitter\fault, and C:\FileEmitter\archive.

3.6. Restart the Monitor Server.

4 Verifying the configuration and deployment of the Emitter application

1. *Ensure that the Monitor Server is started.*
2. *Create a file with the following contents (see Listing 1) and save it into the inbound events directory:*

```
EVENTTYPE=CUSTOMER
EVENTACTION=CREATE

ID=0123456789
NAME=JOHN DOE
ADDRESS=3039 CORNWALLIS ROAD
TEL=919-123-4567
EMAIL=JDOE@US.IBM.COM
```

3. *Open a command prompt and change the directory to* `<MONITOR _ PROFILE _ DIR>/bin` *(for example, C:\IBM\WebSphere\ProcServer\profiles\wbmonitor\bin).*
4. *Run the script eventquery.jacl.*

   For example: wsadmin -f ..\event\bin\eventquery.jacl -group "All events"
   All the CBEs categorized under that group are displayed. Among them should be one or more representing the file containing the event information that was saved in the inbounds event folder.

5 Importing and working with the source code projects

The source code for the Sample File Event Emitter is available in the source archive package in the form of WID projects. The following projects need to be imported into the workspace: EmitterFW, CEIEmitter, FileEmitterEAR, FileEmitterEJB, and FileEmitterImpl.

To import a project perform the following steps:

1. *Open **WebSphere Integration Developer**.*
2. *From the menu bar, select **File→ Import**.*
3. *In the Import dialog box, select **Existing Project into Workspace...** then click **Next**.*
4. *Select the project to be imported into the workspace then click **Finish**.*

Perform these four steps for each of the five projects listed previously.

You can expect build errors to be raised until the classpath reference to the events-client.jar library in the CEIEmitter project is corrected. The events-client.jar library includes classes used for creating CBEs and emitting them.

Perform these additional steps:

5. *Right-click the CEIEmitter project and select **Properties**.*
6. *Go to **Java Build Path→ Libraries**.*
7. *Select the **events-client.jar** entry then click **Edit...***
8. *Browse to the folder in your file system where the events-client.jar file is located (for example, <WPS_INSTALL_DIR>/CEI/client).*

5.1 Projects overview

The Sample File Event Emitter source code is split out across five logical projects as follows:

- *EmitterFW: The common emitter framework that is used by all the sample event emitters made available.*
- *CEIEmitter: The CEI emitter code. This is also common code share by all the sample event emitters*
- *FileEmitterEAR: The enterprise application package that is deployed to the server.*
- *FileEmitterEJB: File Emitter specific EJB code.*
- *FileEmitterImpl: File Emitter specific code. This project contains implementation code for retrieving records from the database and formatting events into CBEs. Additional formatters can be implemented and added here to extend the number of application data types that the emitter can process. (See the next section, "Design details – creating a new event formatter to handle additional application data types", for more information on formatters.)*

6 Design details—creating a new event formatter to handle additional application data types

The Sample File Event Emitter is designed so that it can be extended to handle processing of additional application data types by creating new classes that implement the EventFormatter interface (in the EmitterFW project), and by "registering" the event formatter associated with an application data type by updating the FileEventFormatter.properties (in the FileEmitterImpl project).

The inbound events folder is periodically polled by the EmissionController, using the EventFolderPoller, for the presence of new event files. When new files are present in the inbound events folder, the file emitter will obtain the appropriate formatter and retriever based on the data type recorded and based on the mapping specified in the FileEventFormatter.properties file.

6.1 FileEmitterFormatter.properties file

The FileEmitterFormatter.properties file is a typical properties resource file containing entries in the form of "name equals value". Listing 2 shows the one used by this sample.

Listing 2
FileEventFormatter.properties mapping file

```
#
# This properties file defines the formatter to use for a given data
type.
# The accepted format is:
#    EventType=EventFormatter
# Where the EventFormatter is provided as the fully qualified Java
class.
#
CUSTOMER=com.ibm.wbimonitor.samples.fileemitter.formatter.
CUSTOMERFormatterImpl
ORDER=com.ibm.wbimonitor.samples.fileemitter.formatter.
ORDERFormatterImpl
CLAIM=com.ibm.wbimonitor.samples.fileemitter.formatter.
CLAIMFormatterImpl
```

To introduce support for a new application data type, a new entry mapping the event name to the fully qualified event formatter implementation class needs to be added to this file.

6.2 EventFormatter implementation

After an EventFormatter is obtained (based on the mapping in the FileEmitterFormatter.properties file), it will be invoked to handle the transformation of the event from its "native format" to the CBE format that is expected by the Monitor Server.

A new EventFormatter will need to implement the EventFormatter interface defined in the com.ibm.wbimonitor.samples.emitterframework package in the EmitterFW project.

```
public CommonBaseEvent format( Object o ) throws EventFormatFailed
Exception;
```

Inspect the EventFormatter classes implemented by this sample emitter for an idea on how this format( ) interface can be implemented. They can be found in the com.ibm.wbimonitor.samples.db2emitter.formatter package in the FileEmitterImpl project.

7 Working with CBEs and CEI—key concepts

This section outlines the key concepts related to CEI and CBEs used in the Sample File Event Emitter.

7.1. Creating a new or obtaining an existing EventFactory

1. *Creating a new one from scratch (with and without a ContentHandler):*

```
EventFactory eventFactory =
    (EventFactory) EventFactoryFactory.createEventFactory();
EventFactory eventFactory =
    (EventFactory) EventFactoryFactory.createEventFactory
    (ContentHandler);
```

2. *Obtaining an existing one through JNDI (inherits ContentHandler, if one exists):*

```
Context context = new InitialContext();
EventFactory eventFactory = (EventFactory) context.lookup("com/
ibm/events/EventFactory");
```

7.2 Creating the new CommonBaseEvent:

```
CommonBaseEvent event = eventFactory.createCommonBaseEvent
("ActivityEvent");
```

7.3 Setting mandatory fields:

```
event.setVersion( "1.0.1" );
event.setCreationTimeAsLong( System.currentTimeMillis( ) );
event.setGlobalInstanceId( eventFactory.createGlobalInstanceId( )
);

ComponentIdentification componentId = eventFactory.createCompo-
nentIdentification( );
componentId.setApplication( "DB2" );  // Additional setters avail-
able
event.setSourceComponentId( componentId );

Situation situation = eventFactory.createSituation( );
situation.setStopSituation( "EXTERNAL", "STOP _ COMPLETED",
"SUCCESSFUL" ); event.setSituation( situation );
```

7.4 Creating an ExtendedDataElement and its children:

```
ExtendedDataElement activityEventData =
    event.addExtendedDataElementWithNoValue
    ( "ActivityEventData" );
activityEventData.addChild( "activityName", activityName );
activityEventData.addChild( "eventType", "completed" );
activityEventData.addChild( "activityDisplayState", "Completed" );
activityEventData.addChildWithDateAsLongValue( "startTime", efb.
getLastModifiedDate( ) );
activityEventData.addChildWithDateAsLongValue( "endTime", efb.
getLastModifiedDate( ) );
```

7.5 Obtaining the EmitterFactory:

```
import javax.naming.*
import com.ibm.events.*
Context context = new InitialContext();
EmitterFactory emitterFactory =
    (EmitterFactory) context.lookup("com/ibm/events/configuration/
    emitter/Default");
```

7.6 Obtaining the Emitter from the EmitterFactory:

```
Emitter emitter = emitterFactory.getEmitter();
```

7.7 Sending an Event:

```
emitter.sendEvent((CommonsBaseEvent)event);
```

IBM Corporation
Software Group
Route 100
Somers, NY 10589
U.S.A.

Produced in the United States of America
12-06