JBoss ESB 4.2.1 GA

Message Store

JBESB-MS-10/31/07

Legal Notices

The information contained in this documentation is subject to change without notice.

JBoss Inc. makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. JBoss Inc. shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Java™ and J2EE is a U.S. trademark of Sun Microsystems, Inc. Microsoft® and Windows NT® are registered trademarks of Microsoft Corporation. Oracle® is a registered U.S. trademark and Oracle9™, Oracle9 Server™ Oracle9 Enterprise Edition™ are trademarks of Oracle Corporation. Unix is used here as a generic term covering all versions of the UNIX® operating system. UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

Software Version

JBoss ESB 4.2.1 GA

Restricted Rights Legend

Use, duplication, or disclosure is subject to restrictions as set forth in contract subdivision (c)(1)(ii) of the Rights in Technical Data and Computer Software clause 52.227-FAR14.

Contents

Table of Contents

Contents..... ..iv

About This Guide..... 5

What This Guide Contains.....5
Audience..... 5
Prerequisites..... 5
Organization..... 5
Documentation Conventions.....5
Additional Documentation.....7
Contacting Us..... 7

The Message Store?..... 9

Introduction..... 9
Message Store interface..... 9
Configuring the Message Store.....9

About This Guide

What This Guide Contains

The Message Store contains contain important information on changes to JBoss ESB 4.2.1 GA since the last release and information on any outstanding issues.

Audience

This guide is most relevant to engineers who are responsible for administering JBoss ESB 4.2.1 GA installations.

Prerequisites

None.

Organization

This guide contains the following chapters:

- **Chapter 1, What is the Message Store:** an overview of what the message store provides and how it is used in JBossESB.

Documentation Conventions

The following conventions are used in this guide:

Convention	Description
Italic	In paragraph text, italic identifies the titles of documents that are being referenced. When used in conjunction with the Code text described below, italics identify a variable that should be replaced by the user with an actual value.
Bold	Emphasizes items of particular importance.
`Code`	Text that represents programming code.
Function | Function	A path to a function or dialog box within an interface. For example, "Select File | Open." indicates that you should select the Open function from the File menu.
( ) and |	Parentheses enclose optional items in command syntax. The vertical bar separates syntax items in a list of choices. For example, any of the following three items can be entered in this syntax: `persistPolicy (Never | OnTimer | OnUpdate | NoMoreOftenThan)`
Note: **Caution**:	A note highlights important supplemental information. A caution highlights procedures or information that is necessary to avoid damage to equipment, damage to software, loss of data, or invalid test results.

Table 1 Formatting Conventions

Additional Documentation

In addition to this guide, the following guides are available in the JBoss ESB 4.2.1 GA documentation set:

1. **JBoss ESB 4.2.1 GA** *Trailblazer Guide*: Provides guidance for using the trailblazer example.

2. **JBoss ESB 4.2.1 GA** *Getting Started Guide*: Provides a quick start reference to configuring and using the ESB.

3. **JBoss ESB 4.2.1 GA** *Programmers Guide*: How to use JBossESB.

4. **JBoss ESB 4.2.1 GA** *Release Notes*: Information on the differences between this release and previous releases.

5. **JBoss ESB 4.2.1 GA** *Administration Guide*: How to manage the ESB.

Contacting Us

Questions or comments about JBoss ESB 4.2.1 GA should be directed to our support team.

Chapter 1

The Message Store?

Introduction

The message store mechanism in JBossESB is designed with audit tracking purposes in mind. As with other ESB services, it is a pluggable service, which allows for you, the developer to plug in your own persistence mechanism should you have special needs. The implementation supplied with JBossESB is a database persistence mechanism. If you require say, a file persistence mechanism, then it's just a matter of you writing your own service to do this, and override the default behaviour with a configuration change.

One thing to point out with the Message Store – this is a base implementation. We will be working with the community and partners to drive the feature functionality set of the message store to support advanced audit and management requirements. This is meant to be a starting point.

> Note: In JBossESB 4.2 the Message Store is also used for storing messages that need to be redelivered in the event of failures. See the Programmers Guide around the ServiceInvoker for further details.

Message Store interface

The `org.jboss.soa.esb.services.persistence.MessageStore` interface is defined as follows:

```
public interface MessageStore
{
   public MessageURIGenerator getMessageURIGenerator();
   public URI addMessage (Message message, String classification) throws MessageStoreException;
   public Message getMessage (URI uid) throws MessageStoreException;
   public void setUndelivered(URI uid) throws MessageStoreException;
   public void setDelivered(URI uid) throws MessageStoreException;
   public Map<URI, Message> getUndeliveredMessages(String classification) throws
MessageStoreException;
   public Map<URI, Message> getAllMessages(String classification) throws MessageStoreException;
   public Message getMessage (URI uid, String classification) throws MessageStoreException;
   public int removeMessage (URI uid, String classification) throws MessageStoreException;
}
```

The `MessageStore` is responsible for reading and writing `Messages` upon request. Each `Message` must be uniquely identified within the context of the store and each `MessageStore` implementation uses a `URI` to accomplish this identification. This URI is used as the "key" for that message in the database.

> Note: `MessageStore` implementations may use different formats for their `URIs`.

`Messages` can be stored within the store based upon `classification` using `addMessage`. If the `classification` is not defined then it is up to the

implementation of the `MessageStore` how it will store the `Message`. Furthermore, the `classification` is only a hint: implementations are free to ignore this field if necessary.

> Note: It is implementation dependent as to whether or not the `MessageStore` imposes any kind of concurrency control on individual `Messages`. As such, you should use the `removeMessage` operation with care.

Because the current `MessageStore` interface is designed to support both audit trail and redelivery scenarios, you should not use the `setUndelivered`/`setDelivered` and associated operations unless they are applicable!

The default implementation of the MessageStore is provided by the `org.jboss.internal.soa.esb.persistence.format.db.DBMessageStoreImpl` class. The methods in this implementation make the required DB connections (using a pooled Database Manager DBConnectionManager).

To override the `MessageStore` implementation you should look at the **MessageActionGuide** and the `MessagePersister Action`.

Configuring the Message Store

To configure your Message Store, you can change and override the default service implementation through the following settings found in the `jbossesb-properties.xml`:

```
<properties name="dbstore">

        <!--  connection manager type -->
          <property name="org.jboss.soa.esb.persistence.db.conn.manager"
        value="org.jboss.internal.soa.esb.persistence.manager.StandaloneConnectionManager"/>
          <!--  property name="org.jboss.soa.esb.persistence.db.conn.manager"
value="org.jboss.internal.soa.esb.persistence.manager.J2eeConnectionManager"/ -->

          <!-- this property is only used if using the j2ee connection manager -->
          <property name="org.jboss.soa.esb.persistence.db.datasource.name"
        value="java:/JBossesbDS"/>

                <!-- standalone connection pooling settings -->
                <!--  mysql
                <property name="org.jboss.soa.esb.persistence.db.connection.url"
        value="jdbc:mysql://localhost/jbossesb"/>
                <property name="org.jboss.soa.esb.persistence.db.jdbc.driver"
        value="com.mysql.jdbc.Driver"/>
                <property name="org.jboss.soa.esb.persistence.db.user"
        value="kstam"/>
                 -->
                <!--  postgres
                <property name="org.jboss.soa.esb.persistence.db.connection.url"
        value="jdbc:postgresql://localhost/jbossesb"/>
                <property name="org.jboss.soa.esb.persistence.db.jdbc.driver"
        value="org.postgresql.Driver"/>
                <property name="org.jboss.soa.esb.persistence.db.user"
        value="postgres"/>
                <property name="org.jboss.soa.esb.persistence.db.pwd"
        value="postgres"/>
                -->
```

```
            <!-- hsqldb -->
            <property name="org.jboss.soa.esb.persistence.db.connection.url"
        value="jdbc:hsqldb:hsql://localhost:9001/jbossesb"/>
            <property name="org.jboss.soa.esb.persistence.db.jdbc.driver"
        value="org.hsqldb.jdbcDriver"/>
            <property name="org.jboss.soa.esb.persistence.db.user"
        value="sa"/>
            <property name="org.jboss.soa.esb.persistence.db.pwd"
        value=""/>

            <property name="org.jboss.soa.esb.persistence.db.pool.initial.size"       value="2"/>
            <property name="org.jboss.soa.esb.persistence.db.pool.min.size"
        value="2"/>
            <property name="org.jboss.soa.esb.persistence.db.pool.max.size"
        value="5"/>
            <!--table managed by pool to test for valid connections - created by pool automatically --
>
            <property name="org.jboss.soa.esb.persistence.db.pool.test.table"
        value="pooltest"/>
            <property name="org.jboss.soa.esb.persistence.db.pool.timeout.millis"  value="5000"/>

  </properties>
```

The section in the property file called "dbstore" has all the settings required by the database implementation of the message store. The standard settings, like URL, db user, password, pool sizes can all be modified here.

The scripts for the required database schema, are again, very simple. They can be found under lib/jbossesb.esb/message-store-sql/<db_type>/create_database.sql of your JBossESB installation.

The structure of the table can be seen from the sample SQL:

```
CREATE TABLE message
(
 uuid varchar(128) NOT NULL,
 type varchar(128) NOT NULL,
 message text(4000) NOT NULL,
 delivered varchar(10) NOT NULL,
 classification varchar(10),
 PRIMARY KEY  (`uuid`)
);
```

the uuid column is used to store a unique key for this message, in the format of a standard URI. A key for a message would look like:

```
urn:jboss:esb:message:UID: + UUID.randomUUID()
```

This logic uses the new UUID random number generator in jdk 1.5.the type will be the type of the stored message. JBossESB ships with JBOSS_XML and JAVA_SERIALIZEDcurrently.

The "message" column will contain the actual message content.

The supplied database message store implementation works by invoking a connection manager to your configured database. Supplied with Jboss ESB is a standalone connection manager, and another for using a JNDI datasource.

To configure the database connection manager, you need to provide the connection manager implementation in the *jbossesb-properties.xml*. The properties that you would need to change are:

```
<!--  connection manager type -->
<property name="org.jboss.soa.esb.persistence.db.conn.manager"
value="org.jboss.internal.soa.esb.persistence.format.db.StandaloneCo
nnectionManager"/>
<!--  property name="org.jboss.soa.esb.persistence.db.conn.manager"
value="org.jboss.soa.esb.persistence.manager.J2eeConnectionManager"/
-->
<!-- this property is only used if using the j2ee connection manager
-->
<property name="org.jboss.soa.esb.persistence.db.datasource.name"
value="java:/JBossesbDS"/>
```

The two supplied connection managers for managing the database pool are

```
org.jboss.soa.esb.persistence.manager.J2eeConnectionManager
org.jboss.soa.esb.persistence.manager.StandaloneConnectionManager
```

The Standalone manager uses C3PO to manage the connection pooling logic, and the J2eeConnectionManager uses a datasource to manage it's connection pool. This is intended for use when deploying your ESB endpoints inside a container such as Jboss AS or Tomcat, etc. You can plug in your own connection pool manager by implementing the interface:

```
org.jboss.internal.soa.esb.persistence.manager.ConnectionManager
```

Once you have implemented this interface, you update the properties file with your new class, and the connection manager factory will now use your implementation.