Event Replicator Target Adapter

Using the Event Replicator Target Adapter

Version 3.4

March 2019

ADABAS & NATURAL

This document applies to Event Replicator Target Adapter Version 3.4 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: ART-AARTUSER-34-20220203

Table of Contents

Preface

This document describes how to use the Event Replicator Target Adapter to transform and apply replicated Adabas mainframe data to a relational database, such as DB2, MySQL, Oracle, SQL Server, Sybase, or Teradata.

Note: Event Replicator Target Adapter only supports replicating data to one relational database at a time.

It covers the following topics:

Starting the Event Replicator Target Adapter	Describes how to start the Event Replicator Target Adapter in Windows and UNIX environments.
Activating Event Replicator Target Adapter Processing	Describes how to activate Event Replicator Target Adapter processing.
Deactivating Event Replicator Target Adapter Processing	Describes how to deactivate Event Replicator Target Adapter processing.
Shutting Down the Event Replicator Target Adapter	Describes how to shut down the Event Replicator Target Adapter in Windows and UNIX environments.
Requesting Initial-State Data	Describes how to request that initial-state data be applied to your RDBMS using the Adabas Event Replicator Subsystem.
Replicating Data to JMS Queues or Topics	Describes how to use the Event Replicator Target Adapter to replicate data to JMS queues and topics that are managed by webMethods Broker JNDI providers.
Replicating Data to Adabas	Describes how to use the Event Replicator Target Adapter to replicate data to Adabas.
Replicating Data to Terracotta Caches	Describes how to use the Event Replicator Target Adapter to replicate data to Terracotta in memory caches.
Replicating Data to TerracottaDB	Describes how to use the Event Replicator Target Adapter to replicate data to TerracottaDB.
Using the DB2 CONNECT IMPORT Command to Load Initial-State Data	Describes how to use the DB2 CONNECT IMPORT command to load initial-state data into a DB2 database.
Using the Microsoft SQL Server Bulk Copy (bcp) Utility to Load Initial-State Data	Describes how to use the Microsoft SQL Server bulk copy (bcp) utility to load initial-state data into a Microsoft SQL Server database.
*Using the Oracle SQL*Loader to Load Initial-State Data*	Describes how to request that initial-state data be applied to your Oracle database using the Oracle SQL*Loader.
Using the PostgreSQL psql Utility to Load Initial-State Data	Describes how to use the PostgreSQL psql utility to process and load initial-state data into a PostgreSQL database.
Using the Teradata MultiLoad (MLoad) Utility to Load Initial-State Data	Describes how to use the Teradata MultiLoad (MLoad) utility to load initial-state data into a Teradata database.

Requesting RDBMS Data Refreshes	Describes how to request RDBMS data refreshes.
Dropping RDBMS Tables	Describes how to drop RDBMS tables.
Replicating ADALOD Data to the RDBMS Tables	Describes how to replicate ADALOD data to your RDBMS tables.
Managing Event Replicator Target Adapter Log Files and Console Messages	Describes how to manage and review the Event Replicator Target Adapter log files and console messages.
Controlling Event Replicator for Adabas Destination Definitions	Describes batch utilities provided in the Event Replicator Target Adapter that you can use to activate, deactivate, open and close Event Replicator for Adabas destination definitions.
Samples	Provides a sample of Event Replicator Target Adapter use.

1 About this Documentation

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

In addition, you can also access the cloud product documentation via **https://www.softwareag.cloud**. Navigate to the desired product and then, depending on your solution, go to "Developer Center", "User Center" or "Documentation".

Product Training

You can find helpful product training material on our Learning Portal at **https://knowledge.softwareag.com**.

Tech Community

You can collaborate with Software AG experts on our Tech Community website at **https://tech-community.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software AG news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://hub.docker.com/publishers/softwareag** and discover additional Software AG resources.

Product Support

Support for Software AG products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

2 Starting the Event Replicator Target Adapter

This chapter describes how to start Event Replicator Target Adapter in Windows and in UNIX as well as how to start it from the Administration tool.

Note: The Event Replicator Target Adapter is automatically stopped and restarted every time an Event Replicator Target Adapter definition is saved or deleted using the Administration tool. For this reason, we recommend that you shut down the Event Replicator Target Adapter prior to making changes to the definitions and restart the Event Replicator Target Adapter when all definition changes are complete.

Starting Event Replicator Target Adapter from the Administration Tool

Note: At least one source and one target definition must be configured before you can start the Event Replicator Target Adapter using the Administration tool.

To start Event Replicator Target Adapter from the Administration tool:

1 Start up the Administration tool, as described in *Starting the Administration Tool*, in the *Event Replicator Target Adapter Administration Guide*.

The Administration tool window appears.

2 Click on the **Start Target Adapter** button () on the Administration tool window.

Or:

On the **Run** menu, select **Start Target Adapter**.

The Event Replicator Target Adapter starts up.

Restarting the Event Replicator Target Adapter

The Event Replicator Target Adapter can be restarted from the Administration tool.

To restart Event Replicator Target Adapter from the Administration tool once it is already started:

- Click on the **Restart Target Adapter** button () on the Administration tool window.

  Or:

  On the **Run** menu, select **Restart Target Adapter**.

  The Event Replicator Target Adapter stops and restarts.

Starting Event Replicator Target Adapter in Windows

To start Event Replicator Target Adapter using the *startupart* batch file in Windows environments:

1 Navigate to the following directory: *<installation-dir>/Event Replicator Target Adapter/Vv.r.s/bin*, where *<installation-dir>* is the name of the directory in which the Event Replicator Target Adapter program files were installed and *v.r.s* represents the current version, release, and system modification level of the Event Replicator Target Adapter.

Or:

From the Windows Start menu, select **All Programs**, then select **Software AG Event Replicator Target Adapter** *v.r.s* (where *v.r.s* represents the current version, release, and system modification level of the Event Replicator Target Adapter), and finally select **Event Replicator Target Adapter Command Prompt**. A command prompt window appears, already placed in the correct directory.

2 Run the *startupart.bat* file.

The Event Replicator Target Adapter starts up.

Even if there are no target, source, or target database option definitions defined for the Event Replicator Target Adapter, it will still start. If no definitions are defined, no processing will occur and no replication data can be received, but the Event Replicator Target Adapter will start.

Starting Event Replicator Target Adapter in UNIX

To start Event Replicator Target Adapter in UNIX:

1 Navigate to the following directory: *<installation-dir>\Event Replicator Target Adapter\Vvrs\bin*, where *<installation-dir>* is the name of the directory in which the Event Replicator Target Adapter program files were installed and *vrs* represents the current version, release, and system modification level of the Event Replicator Target Adapter.

2 Run the *startupart.sh* script at the UNIX prompt.

The Event Replicator Target Adapter starts up.

Even if there are no target, source, or target database option definitions defined for the Event Replicator Target Adapter, it will still start. If no definitions are defined, no processing will occur and no replication data can be received, but the Event Replicator Target Adapter will start.

3 Activating Event Replicator Target Adapter Processing

Once Event Replicator Target Adapter has been installed and your relational database has been set up properly, you can activate Event Replicator Target Adapter processing.

To activate Event Replicator Target Adapter processing:

1 Verify that you have proper authorization privileges to maintain the RDBMS tables.

Effective with Event Replicator Target Adapter 2.7, user authorization to maintain any new RDBMS tables via Event Replicator Target Adapter is inherited from the site's privilege settings for the database. Authorization is managed by the user's RDBMS privileges and not by Event Replicator Target Adapter. Event Replicator Target Adapter will no longer grant RDBMS privileges to the user. Therefore, if you want to use Event Replicator Target Adapter to maintain tables in an RDBMS, verify that your RDBMS authorization privileges are correct for the maintenance you want to perform.

2 Verify that appropriate target definitions and target database processing option definitions have been created in the Event Replicator Target Adapter Administration tool. One target definition should be created for each RDBMS target to which you want data replicated. A target database processing option definition is only necessary if the settings in the default target database processing option definition are not sufficient. For more information, read *Configuring Target Definitions for Event Replicator Target Adapter* and *Specifying Target Database Processing Option Definitions*, in the *Event Replicator Target Adapter Administration Guide*.

3 Verify that appropriate Event Replicator Server queue definitions have been set up for the messaging system (webMethods EntireX or WebSphere MQ) you are using. For more information about maintaining queue definitions, read the documentation in one of the following chapters:

- To maintain queue definitions using DDKARTE statements, read *IQUEUE Parameter* in *Event Replicator for Adabas Reference Guide*.
- To maintain queue definitions using the Adabas Event Replicator Subsystem, read *Maintaining Input Queue (destination) Definitions* in *Adabas Event Replicator Subsystem User's Guide*.

4 Verify that you have corresponding source definitions created in the Event Replicator Target Adapter Administration tool for every IQUEUE definition you need to use to replicate data from your mainframe Adabas to your relational database. For information on configuring source definitions in the Event Replicator Target Adapter Administration tool, read *Configuring Source Definitions for Event Replicator Target Adapter*, in the *Event Replicator Target Adapter Administration Guide*.

5 Verify that appropriate Event Replicator Server subscription definitions have been set up for data replication to your relational database. For more information about maintaining subscription definitions, read the documentation in one of the following chapters:

- To maintain subscription definitions using DDKARTE statements, read *SUBSCRIPTION Parameter* in *Event Replicator for Adabas Reference Guide*.
- To maintain subscription definitions using the Adabas Event Replicator Subsystem, read *Maintaining Subscription Definitions* in *Adabas Event Replicator Subsystem User's Guide*.

6 Verify that a global format buffer and field table (GFFT) have been *generated* for the subscription definition. To use Event Replicator Target Adapter, you must use generated global format buffers. To generate a global format buffer and field table (GFFT) using the Adabas Event Replicator Subsystem, read *Generating a GFB* in *Adabas Event Replicator Subsystem User's Guide*.

Note: If you want to use UTF-8 character encoding, you must verify that your GFB field lengths are increased as required to accommodate the character set referenced by the code page you select and the data requested in the GFB. You can increase these field lengths manually by editing the GFB itself or by editing the Predict file or data definition module (DDM) used when the GFB is generated.

7 Save the subscription definitions.

8 Edit the destination definitions used by the subscriptions you modified in the previous steps.

9 Verify that the destination class (DCLASS parameter) is specified as "SAGTARG" for each destination.

Note: DCLASS=SAGTARG cannot be specified for Adabas or file destinations (DTYPE=ADABAS or FILE). It is only valid for webMethods EntireX, WebSphere MQ, or NULL destinations. When DCLASS=SAGTARG is specified, the ADARUN RPLPARMS parameter must be set to "FILE" or "BOTH" to provide access to any field table (GFFT) definitions.

Optionally, specify keyword "NOSPRE" for the destination class parameter (DCLASSPARM parameter) if you do not want the subscription name to prefix the names of the tables produced by the Event Replicator Target Adapter. When "NOSPRE" is specified, the schema file name (Predict view name) alone is used for the table names; when "NOSPRE" is *not* specified, the subscription name prefixes the Predict view name in the table names.

For more information about specifying the destination class in a destination definition, read the documentation in one of the following chapters:

- To modify the destination definition using DDKARTE statements, read *DESTINATION Parameter* in *Event Replicator for Adabas Reference Guide*.
- To modify the destination definition using the Adabas Event Replicator Subsystem, read *Maintaining Destination Definitions* in *Adabas Event Replicator Subsystem User's Guide*.

10 Save the destination definitions.

11 Stop and restart the Event Replicator Server or refresh the destination and subscription definitions, as described in *RPLREFRESH Command*, in *Event Replicator for Adabas Administration and Operations Guide*.

12 Activate the destination definitions and the subscription definitions you modified in the previous steps. For more information, read one or more of the following chapters:

- To activate destination and subscription definitions using Adabas Online System (AOS), read *Managing Replication Definitions from AOS* in *Adabas Event Replicator Subsystem User's Guide*.
- To activate destination definitions using an Event Replicator Target Adapter batch utility, read ***Controlling Event Replicator for Adabas Destination Definitions***, elsewhere in this chapter.

If the subscription definitions are already activated, deactivate them first and then reactivate them to pick up the changes made in these steps.

13 Verify that the message systems referenced by your Event Replicator Server queue definitions and your Event Replicator Target Adapter source definitions are started or that their associated source definitions in the Administration tool have **Autostart** selected (checked).

14 If it is not already started, start the Event Replicator Target Adapter.

Event Replicator Target Adapter processing is activated and replicated Adabas data will be transformed and applied to your RDBMS.

Note: The Event Replicator Target Adapter will not process updates for a field that is the primary key for an RDBMS table or for any fields that comprise a composite key for the table.

4 Deactivating Event Replicator Target Adapter Processing

To deactivate Event Replicator Target Adapter processing:

- Deactivate the subscription definitions that manage the replicated data you are transforming and applying to your relational database. To deactivate subscription definitions using Adabas Online System (AOS), read *Managing Replication Definitions from AOS* in *Adabas Event Replicator Subsystem User's Guide*.

  Event Replicator Target Adapter processing is deactivated.

5 Shutting Down the Event Replicator Target Adapter

This chapter describes how to shut down the Event Replicator Target Adapter in Windows and UNIX environments as how to stop it from the Administration tool.

Note: The Event Replicator Target Adapter is automatically stopped and restarted every time an Event Replicator Target Adapter definition is saved or deleted using the Administration tool. For this reason, we recommend that you shut down the Event Replicator Target Adapter prior to making changes to the definitions and restart the Event Replicator Target Adapter when all definition changes are complete.

Stopping Event Replicator Target Adapter from the Administration Tool

To shut down Event Replicator Target Adapter from the Administration tool:

1 Start up the Administration tool, as described in *Starting the Administration Tool*, in the *Event Replicator Target Adapter Administration Guide*.

The Administration tool window appears.

2 Click on the **Stop Target Adapter** button ( ) on the Administration tool window.

Or:

On the **Run** menu, select **Stop Target Adapter**.

The Event Replicator Target Adapter shuts down.

Shutting Down Event Replicator Target Adapter in Windows

To shut down Event Replicator Target Adapter using the *shutdownart* batch file in Windows:

1 Navigate to the following directory: *<installation-dir>/Event Replicator Target Adapter/Vv.r.s/bin*, where `<installation-dir>` is the name of the directory in which the Event Replicator Target Adapter program files were installed and `v.r.s` represents the current version, release, and system modification level of the Event Replicator Target Adapter.

2 Run the *shutdownart.bat* file.

The Event Replicator Target Adapter shuts down.

Shutting Down Event Replicator Target Adapter in UNIX

To shut down Event Replicator Target Adapter in UNIX environments:

1 Navigate to the following directory: *<installation-dir>\Event Replicator Target Adapter\Vvrs\bin*, where *<installation-dir>*is the name of the directory in which the Event Replicator Target Adapter program files were installed and *vrs* represents the current version, release, and system modification level of the Event Replicator Target Adapter.

2 Enter `shutdownart.sh` at the UNIX prompt.

The Event Replicator Target Adapter shuts down.

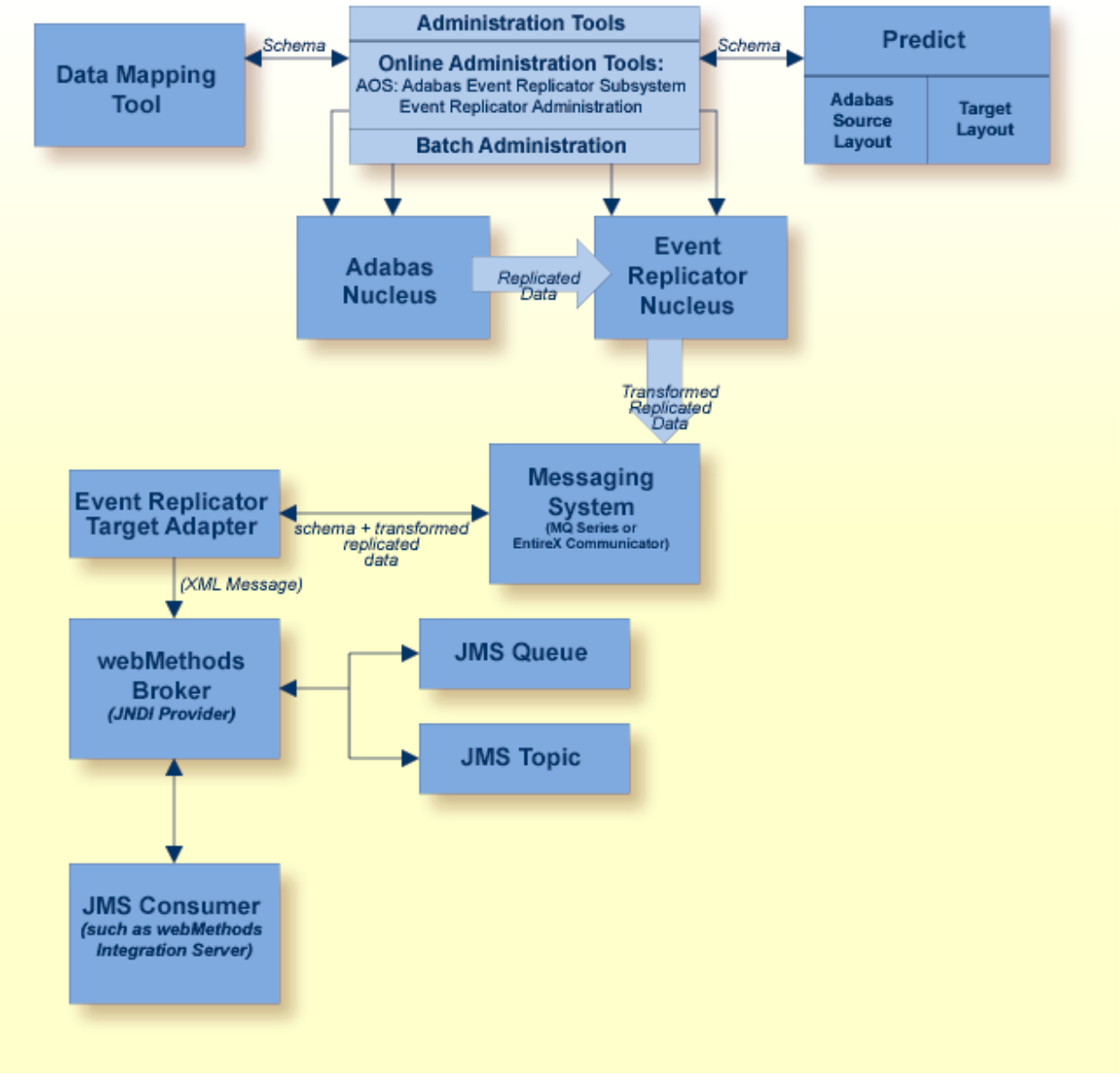
6 Requesting Initial-State Data

Using the Adabas Event Replicator Subsystem, you can manually submit requests to the Event Replicator Target Adapter that initiate an initial-state request to populate the relational database tables. For information on using the Adabas Event Replicator Subsystem to do this, read *Populating the RDBMS Tables* in *Adabas Event Replicator Subsystem User's Guide*.

7 Replicating Data to JMS Queues or Topics

You can use the Event Replicator Target Adapter to replicate data to JMS queues and topics that are managed by webMethods Broker JNDI providers. Replicate the data to a JMS topic if you want to use publish-subscribe messaging; replicate the data to a JMS queue if you want to use point-to-point messaging. Once the data is published on a topic or queue, message consumers, such as webMethods Integration Server, can retrieve or consume it.

The following diagram depicts the interaction between the Event Replicator Target Adapter and the JNDI providers managed by webMethods Broker.



This chapter describes how to set up this JMS support in the Event Replicator Target Adapter and the high-level steps that must be taken in webMethods Broker.

Requirements

webMethods Broker and My webMethods Version 7.1 or later must be installed. In addition, JMS support must be established within webMethods Broker. For complete information, read *EntireX Message Service Provider (MSP)*, available in the webMethods EntireX documentation on Software AG's **Empower** web site.

Implementation Steps

To set up replication of Adabas data to a JMS topic or queue, complete the following steps:

1 Set up JMS support in webMethods Broker specific for use by the Event Replicator Target Adapter. Be sure to perform the following steps in webMethods Broker:

- Establish a JNDI (Java Naming and Directory Interface) provider for use by the Event Replicator Target Adapter. The root location of the JNDI is called the *initial context*. The JNDI serves as a template for the JMS administered objects (connection factories and destinations), storing their property settings in specified locations relative to the initial context. These settings are applied when the JMS connection factories and destinations are created and used when the JMS application runs.
- Add topic and queue connection factories that configure the connections between JMS programs and webMethods Broker Server.
- Bind and create topic and queue destinations in the JNDI.
- As necessary, update client group access permissions to ensure they have permission to publish and subscribe to the document types used in any durable subscriptions for the JNDI's topics.
- As necessary, create durable subscriptions, linking specific document types to a topic. This is necessary for subscriptions whose data is persisted if the subscriber is disconnected.

For complete information, read *EntireX Message Service Provider (MSP)*, available in the webMethods EntireX documentation on Software AG's **Empower** web site.

2 Collect the following information about the JMS support in webMethods Broker:

- The URL of the initial context of the JNDI provider established for use by the Event Replicator Target Adapter.
- The lookup name of the JMS topic or queue destination (as defined to webMethods Broker) that should be used by the Event Replicator Target Adapter.
- The name of the JMS topic or queue connection factory (as defined to webMethods Broker) that should be used by the Event Replicator Target Adapter.

3 Using the Event Replicator Target Adapter Administration tool, add a JMS target definition and save it. For more information, read *Maintaining JMS Target Definitions*, in the *Event Replicator Target Adapter Administration Guide*.

4 Start the Event Replicator Target Adapter (if it is not already started), as described in ***Starting the Event Replicator Target Adapter***, elsewhere in this guide.

5 Using the Adabas Event Replicator Subsystem (SYSRPTR), edit the Event Replicator subscriptions you want used to transform and submit data to the Event Replicator Target Adapter. For each subscription, verify that:

- A global format buffer (GFB) and field table (GFFT) are generated for at least one SFILE definition in the subscription. The GFB and GFFT must be generated either by Event Replicator for Adabas using Predict file definitions or using the Data Mapping Tool.
- At least one destination definition used by the subscription specifies a value of "SAGTARG" for the **Destination Class** field (DCLASS parameter).

  Note: DCLASS=SAGTARG cannot be specified for Adabas or file destinations (DTYPE=ADABAS or FILE). It is only valid for webMethods EntireX, WebSphere MQ, or NULL destinations. When DCLASS=SAGTARG is specified, the ADARUN RPLPARMS parameter must be set to "FILE" or "BOTH" to provide access to any field table (GFFT) definitions.

Be sure to save the subscription definitions and any destination definitions you modified.

6 Activate the Event Replicator subscription and destination definitions. To activate subscription and destination definitions using Adabas Online System (AOS), read *Managing Replication Definitions from AOS*, in the *Event Replicator for Adabas Administration and Operations Guide*.

XML Message Structure for JMS

This section describes the structure of the XML messages containing replicated data that the Event Replicator Target Adapter publishes on JMS topics and queues. This structure is defined by an XML schema which is included with your Event Replicator Target Adapter.

This section is organized into the following topics:

XML Schema

The schema defining the JMS message structure is provided with the Event Replicator Target Adapter. It is called *jms.xsd* and can be found in the following locations:

Platform	Directory
Windows Vista	**C:\Program Files\Software AG\Event Replicator Target Adapter\v***v.r.s***\bin**
Other Windows platforms	**C:\Program Files\Software AG\Event Replicator Target Adapter\v***v.r.s***\bin**
UNIX	**$SAG/art/v***vrs***/bin**

XML Elements

The following XML elements are used in the XML messages sent to JMS topics and queues.

Element Name	Hierarchy Level	Attributes	Subelements	Description
`<artdocument/>`	1	version	`<method/>` `<tableName/>` `<fields/>`	Identifies the message as an Event Replicator Target Adapter message containing replicated data from an Adabas database.
`<field/>`	3	length name pre type value	—	Identifies an individual field (column) in the table that is affected by the data or table change and provides the details about that change. The attributes that appear in the `<field/>` element vary, based on the setting of the `<method/>` element.
`<fields/>`	2	—	`<field/>`	Surrounds the set of fields affected by the replicated data or table change.
`<method/>`	2	—	—	Identifies the kind of change applied to the database. Valid values are: ■ "create": Indicates that a new table is being created. ■ "delete": Indicates that a field (column) in a table is being deleted. ■ "drop": Indicates that a table is being dropped. ■ "insert": Indicates that a new field (column) is being inserted into a table. ■ "update": Indicates that the value of a field (column) is being updated.

Element Name	Hierarchy Level	Attributes	Subelements	Description
<tableName/>	2	—	—	Identifies the name of the table affected by the replicated data or table change.

XML Attributes

The following XML attributes are used in the XML messages sent to JMS topics and queues.

Attribute Name	Element Used In	Prerequisite	Description
length	<field/>	<method/> element must be "create"	Identifies the length of a field being created.
name	<field/>	always required	The name of the field (column). This is the name passed to the Event Replicator Target Adapter in the generated GFB and GFFT used for Event Replicator Target Adapter processing. This name is assigned to a database field when the Adabas database is initially set up. It can be renamed using the Data Mapping Tool before the Data Mapping Tool is used to generate the GFB and associated GFFT.
pre	<field/>	<method/> element must be "create"	Identifies the precision of a field being created. Alphanumeric fields have a precision of "0" (zero).
type	<field/>	<method/> element must be "create"	Identifies the data type of a field being created.
value	<field/>	<method/> element must be "insert", "update", or "delete"	The value of the field being inserted or updated.
version	<artdocument/>	—	The version of the Event Replicator Target Adapter schema used to generate this XML message.

XML Message Examples

This section includes examples of all possible types of messages submitted by the Event Replicator Target Adapter to JMS queues and topics.

- Field (Column) Deleted

Table Created

The following is a sample XML message produced when a table is created. Note that the `<method/>` element is set to "create.".

```
<artdocument version="1">
   <method>create</method>
   <tableName>empldb31fnr5</tableName>
   <fields>
      <field name="ISN" type="integer" length="30" pre="0" />
      <field name="PERSONNEL-ID" type="string" length="8" pre="0" />
      <field name="FIRST-NAME" type="string" length="20" pre="0" />
      <field name="MIDDLE-I" type="string" length="1" pre="0" />
      <field name="NAME" type="string" length="20" pre="0" />
      <field name="MIDDLE-NAME" type="string" length="20" pre="0" />
      <field name="MAR-STAT" type="string" length="1" pre="0" />
      <field name="SEX" type="string" length="1" pre="0" />
      <field name="BIRTH" type="date" length="10" pre="0" />
      <field name="CITY" type="string" length="20" pre="0" />
      <field name="ZIP" type="string" length="10" pre="0" />
      <field name="POST-CODE" type="string" length="10" pre="0" />
      <field name="COUNTRY" type="string" length="3" pre="0" />
      <field name="AREA-CODE" type="string" length="6" pre="0" />
      <field name="PHONE" type="string" length="15" pre="0" />
      <field name="DEPT" type="string" length="6" pre="0" />
      <field name="JOB-TITLE" type="string" length="25" pre="0" />
      <field name="LEAVE-DUE" type="decimal" length="2" pre="0" />
      <field name="LEAVE-TAKEN" type="decimal" length="2" pre="0" />
   </fields>
</artdocument>
```

Table Dropped

The following is a sample XML message produced when a table is dropped. Note that the `<method/>` element is set to "drop".

```
<artdocument version="1">
   <method>drop</method>
   <tableName>empldb31fnr5</tableName>
</artdocument>
```

Field (Column) Inserted

The following is a sample XML message produced when a field (column) is inserted into a table. Note that the <method/> element is set to "insert".

```
<artdocument version="1">
   <method>insert</method>
   <tableName>empldb31fnr5_INCOME_BONUS</tableName>
   <fields>
      <field name="ISN_INCOME_INDEX" value="1_1" />
      <field name="BONUS" value="23000" />
      <field name="BONUS_IX2" value="1" />
   </fields>
</artdocument>
```

Field (Column) Value Updated

The following is a sample XML message produced when a field (column) value is updated in a table. Note that the <method/> element is set to "update".

```
<artdocument version="1">
   <method>update</method>
   <tableName>empldb31fnr5_INCOME_BONUS</tableName>
   <fields>
      <field name="ISN_INCOME_INDEX" value="1_1" />
      <field name="BONUS" value="230" />
      <field name="BONUS_IX2" value="1" />
   </fields>
</artdocument>
```

Field (Column) Deleted

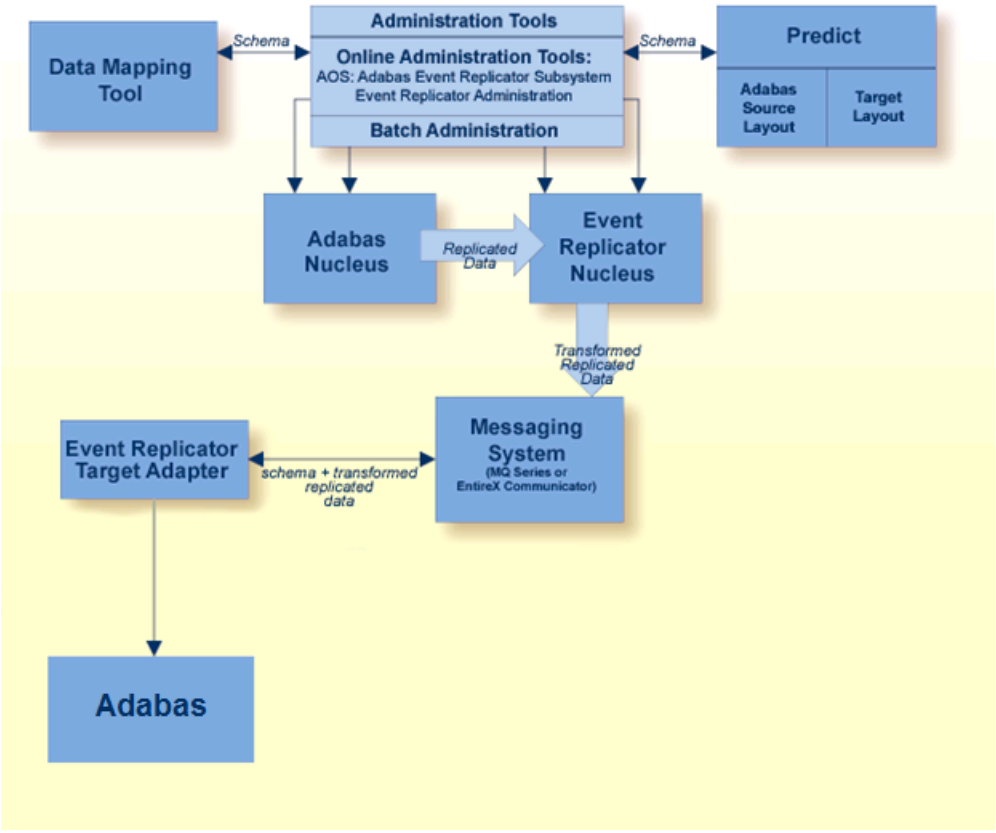
The following is a sample XML message produced when a field (column) is deleted from a table. Note that the <method/> element is set to "delete".

```
<artdocument version="1">
   <method>delete</method>
   <tableName>empldb31fnr5_INCOME_BONUS</tableName>
   <fields>
      <field name="ISN_INCOME_INDEX" value="1_1" />
      <field name="BONUS" value="23000" />
      <field name="BONUS_IX2" value="1" />
   </fields>
</artdocument>
```

8 Replicating Data to Adabas

You can use the Event Replicator Target Adapter to replicate data to Adabas.

The following diagram depicts the interaction between the Event Replicator Target Adapter and Adabas.



This chapter describes how to set up the Adabas target in the Event Replicator Target Adapter. It covers the following topics:

Requirements

- The Event Replicator Target Adapter uses the Adabas Client for Java internally to write the data to Adabas. See the requirements for Adabas Client for Java.
- Direct replication using Adabas file short names requires Event Replicator for Adabas Version 3.7 SP1 with zap AZ371002 applied on the mainframe.

Adabas-specific Notes

Adabas Client for Java and the Map File

The Event Replicator Target Adapter uses Adabas Client for Java to process the data. For details about the concept, especially instructions on creating and configuring a Map File in the target database, see the section *Creating Maps* in the *Adabas Client for Java* documentation.

Depending on the user's desired end result and configuration, there are times when a map file is required and times when it is not.

A Map File and target Map *are not* required when:

- The Adabas source and target file numbers are the same.
- Event Replicator for Adabas Version 3.7 SP1 is installed on the mainframe with zap AZ371002 applied.
- The Event Replicator Destination is defined with `DCLASSPARM=NOSPRE,ADA`.
- The user wants the Event Replicator Target Adapter to create the target file and is not concerned if all extended field attributes are replicated when the target file is created. By default, only UQ and DE attributes are sent as defined in the GFFT, with all fields defined at the target as Null Suppressed. As a workaround to this behavior, the user can preload the target file with an FDT that fully defines the extended field attributes. The file number must still match the source database file number.

When no Map File is defined in the Adabas target definition in Event Replicator Target Adapter Administration, none will be used or created during Initial State processing. However, the requirements outlined above must be met or the Target Adapter will throw an exception and stop processing.

A Map File and target Map *are* required when:

- The Adabas source and target file numbers are different.
- The mainframe replicator configuration does not meet the requirements outlined above (e.g. short names are not being sent in the schema and replicated data).
- The user desires the target file definition with specific field attributes exactly matching the source (e.g. using the same predefined FDT for file and target).

When a Map File is defined in the Adabas Target definition, a Map is always used. It can be:

- Predefined using the Data Designer in Adabas Client for Java by preloading the target file with an FDT that matches the source file FDT or using a systrans'd DDM. See *Creating Maps* in the *Adabas Client for Java* documentation for details. In this case, the name of the target file and the

name of the map must match and be the same as the file name of the source. The target file number need not be the same as the source file number. or

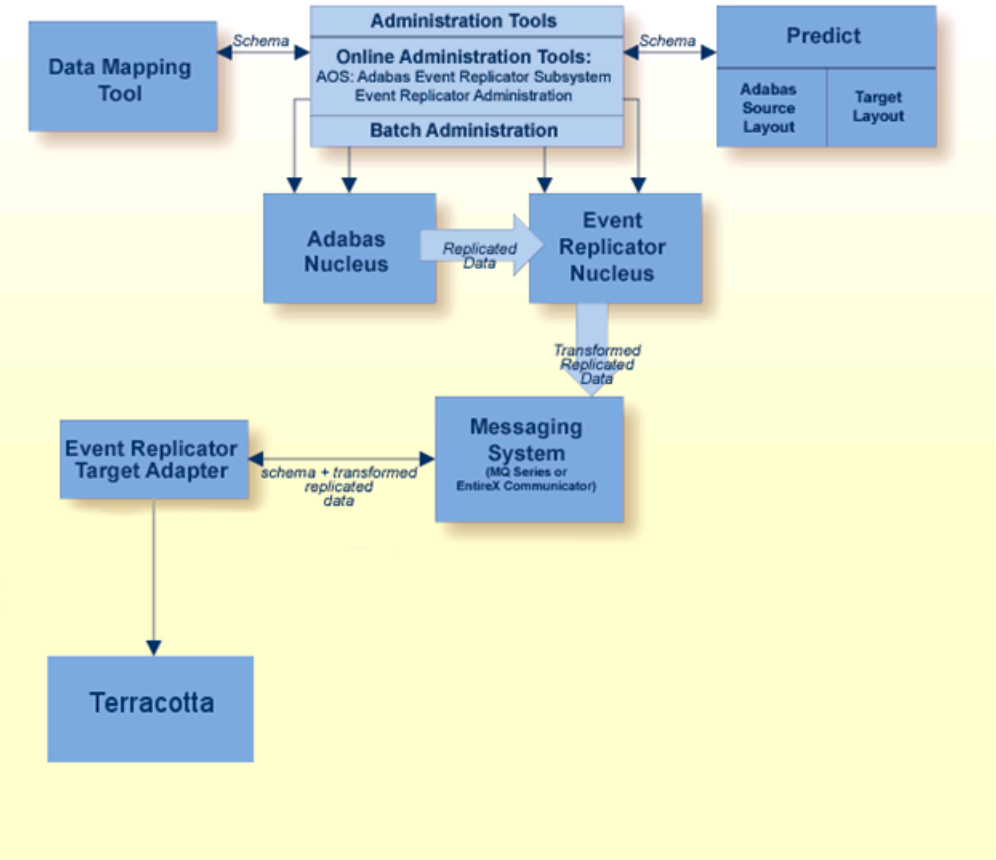
- Created during the Initial State processing, thereby mapping field Long Names from the GFFT definition used by the Event Replicator for Adabas to Adabas short names in the target file "on the fly". If the target database file number that matches the source is not available, the next available file number in the target database will be used.

Note: If the Event Replicator for Adabas is not running Version 3.7 SP1 with AZ371002 applied and with `DCLASSPARM=ADA` defined, the field short names on the target side are assigned by the Target Adapter and will not necessarily agree with those from the target.

9 Replicating Data to Terracotta Caches

You can use the Event Replicator Target Adapter to replicate data to Terracotta in memory caches.

The following diagram depicts the interaction between the Event Replicator Target Adapter and Terracotta.



This document describes how to set up this Terracotta support in the Event Replicator Target Adapter. It covers the following topics:

Requirements

Terracotta Version 4.3 must be installed in your enterprise. For complete information, read *Terracotta*, available in the "IBO and Big Data" documentation on Software AG's **Empower** web site. The necessary components to connect to the Terracotta target are provided with the delivery of Event Replicator Target Adapter, except for the licenses described under Implementation Steps below.

Implementation Steps

Note: In the steps described below, the value of <*Root*> is the upper level directory where your Software AG products are installed. On Windows, the default is `C:\SoftwareAG` and on Linux/UNIX systems the default is `/opt/softwareag`. You may have installed your Software AG products to a different root, in which case you would replace <*Root*> with this location.

Licenses

Event Replicator Target Adapter must be licensed to communicate with the Terracotta target. This is an additional license to the one you used when installing v. In addition, if you are connecting to/using the enterprise features of the Terracotta cache server, you must also provide a copy of that license to the v environment.

To license Event Replicator Target Adapter for connection to the Terracotta server:

- Obtain a license for the Terracotta target from your Software AG representative. This license will likely be provided to you in the form of an .xml file with a name like <*numericprefix*>*_arttc.xml*.
  - Rename this file to *arttc.xml*.
  - Manually copy this file to <*Root*>/common/conf.

You will be able to connect to the Terracotta server without this license file, but your ability to use Terracotta features will be limited to 30 minutes of use in a single Event Replicator Target Adapter session.

The use the Terracotta enterprise features:

To use the features of the enterprise versions of the Terracotta products a Terracotta license is required. This is the same license you were provided in order to use the enterprise features of your Terracotta server installation. A copy of this license must be manually placed in a resource location available to Event Replicator Target Adapter:

- **Recommended**: simply copy the file to:

  Windows: <*Root*>\EventReplicatorTargetAdapter\art\data\webapps\sqlrep\WEB-INF\classes.

  UNIX: <*Root*>/EventReplicatorTargetAdapter/art/TargetAdapter/webapps/sqlrep/WEB-INF/classes.

  This is a resource location already “well known” to Event Replicator Target Adapter and the application will load the license from this resource location.

- **Optional** (not recommended as it involves manually editing an installed file):

  Set the path to the license in the CATALINA_OPTS section of the ERTA.bat (Windows)/ERTA.sh (Unix) file. This file will be found in:

  Windows: <*Root*>\EventReplicatorTargetAdapter\art\program\bin

  UNIX: <*Root*>/EventReplicatorTargetAdapter/art/bin.

  Add the property:

  ```
  -Dcom.tc.productkey.path=/path/to/terracotta-license.key
  ```

  …at the end of the line that sets CATALINA_OPTS.

  For example:

  Windows: set CATALINA_OPTS=-Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.port=%ART_JMX_PORT% -Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.management.jmxremote.authenticate=false -Dcom.tc.productkey.path=/path/to/terracotta-license.key

  UNIX: CATALINA_OPTS="-Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.port=${ART_JMX_PORT} -Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.management.jmxremote.authenticate=false" -Dcom.tc.productkey.path=/path/to/terracotta-license.key

For other possibilities on specifying the location of this file, check the Terracotta documentation.

Terracotta-specific Notes

Cache Configuration File

The Ehcache cache configuration file (*ehcache.xml*) contains the configuration for the CacheManager that is used by the client. It contains all information about the environment, parameters, caches, etc. At a minimum, the *ehcache.xml* file used by Event Replicator Target Adapter should contain the following:

```
<?xml version="1.0" encoding="UTF-8"?>

<ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="ehcache.xsd" updateCheck="false"
 monitoring="autodetect" dynamicConfig="true">

 <!-- Mandatory Default Cache configuration. These settings will be applied
  to caches created programmtically using CacheManager.add(String cacheName).
  The defaultCache has an implicit name "default" which is a reserved cache
  name. -->
```

```
 <defaultCache eternal="true"
  overflowToDisk="false" maxEntriesLocalHeap="1">
  <terracotta/>
 </defaultCache>

 <terracottaConfig url="terracottahost:port" rejoin="true"/> This line is only ↩
needed for interaction with the enterprise version of Terracotta.
</ehcache>
```

For more information about the Cache Configuration File, read *Terracotta*, available in the "IBO and Big Data" documentation on Software AG's **Empower** web site.

Cache Definition

Although not necessary, cache definitions can be defined in the Cache Configuration File (*ehcache.xml*). The path to this file is specified using Event Replicator Target Adapter Administration in the your target configuration.

The main reason to use a cache definition in *echache.xml* is to enable a client application to search by a particular key. Event Replicator Target Adapter does not require your cache be defined in this file; however, specifying the cache definition in *echache.xml* does let you control which Adabas descriptor fields are to be marked in the Terracotta cache as keys (searchable fields). If no cache definition exists in *echache.xml*, the cache will be dynamically created when a CREATE statement (schema) is sent from Event Replicator for Mainframe and all fields specified in the Adabas DDM as descriptors will be marked as searchable in the Terracotta cache. For more information see Searchable Attributes below.

AdabasObject

A Terracotta cache is in general a Key-Value store for Java objects. To store Adabas records in such a Key-Value store the record must be transformed to a Java object (value). The key for this object is the key that is provided by the Event Replication. Unless you specified a custom Primary Key in the replication GFB/GFFT, the ISN by default will be used as key. The Java object that is stored in the cache is an instance of AdabasObject, which is a simple Java class. To retrieve the fields from this objects the method Object `evaluateValue(String fieldName)` must be used, where fieldname is the long name for this field as defined in the GFFT (e.g. NAME, Address, etc.). The result type depends on the type of the Adabas field:

Adabas Structure	Adabas Object	Example
Field	Java type	String, int, float,
Multiple field	List of Java type	List <String>, List<int>
Group	Adabas Object	AdabasObject
Periodic group	List of Adabas Object	List<AdabasObject>

Note: Groups are not supported by the Event Replication Target Adapter.

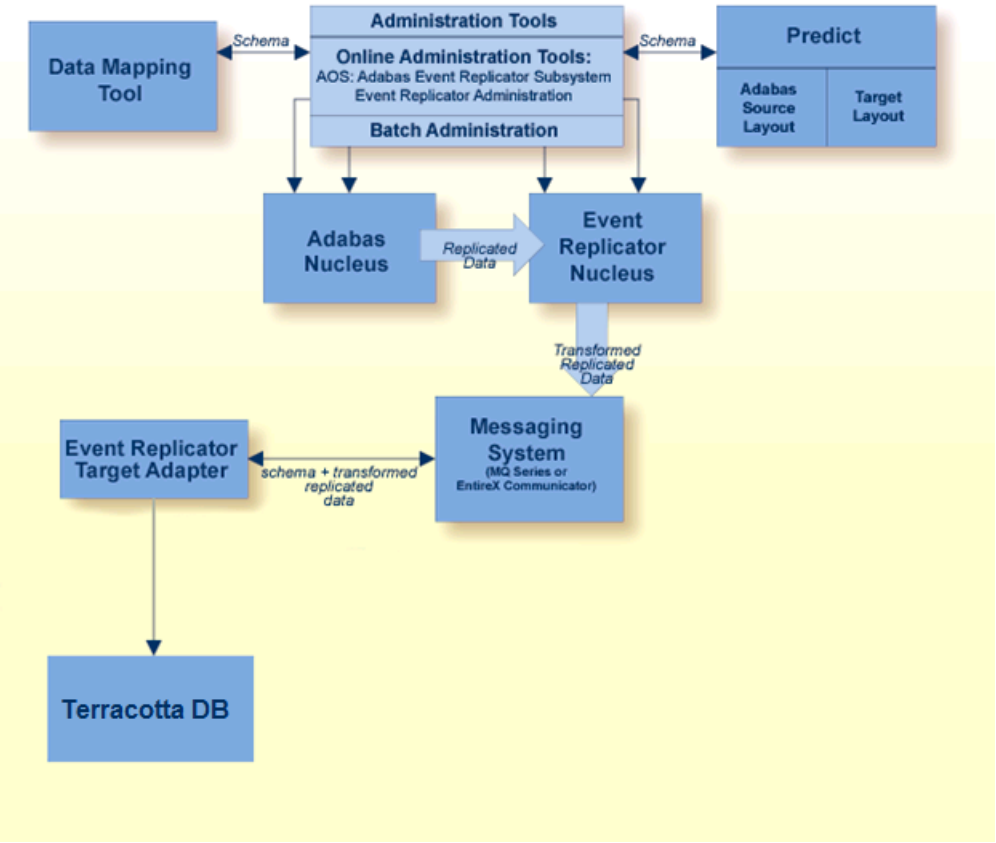
Searchable Attributes

The data in the Terracotta cache is accessible via the key. To use the Terracotta search capabilities the attributes must be marked as searchable. By default, when a cache is created with receipt of a schema from Event Replicator Target Adapter, any Adabas field marked as a Descriptor in the GFFT is created as a searchable key in the Terracotta cache. For information on how to use Searchable Attributes to query the cache, please refer to the *Terracotta* documentation in **Empower**.

10 Replicating Data to Terracotta DB

You can use the Event Replicator Target Adapter to replicate data to Terracotta DB.

The following diagram depicts the interaction between the Event Replicator Target Adapter and Terracotta DB.



This chapter describes how to set up the Terracotta DB target in the Event Replicator Target Adapter. It covers the following topics:

Requirements

Terracotta DB Version 10.1 or higher must be installed in your enterprise. For complete information, read *TCStore API Developer Guide*, available in the *Terracotta DB* documentation on Software AG's **Empower** web site. The necessary components to connect to the Terracotta DB target are provided with the delivery of Event Replicator Target Adapter. Replication to Terracotta DB requires a license.

Terracotta-specific Notes

Sub-Datasets

Terracotta DB is very similar to a relational database. This means that Adabas multiple fields (MU) and periodic groups (PE) are represented as their own sub-datasets, similar to the creation of child tables in relational databases.

11 Using the DB2 CONNECT IMPORT Command to Load Initial-State Data

You can use the DB2 CONNECT IMPORT command to process and load initial-state data into aDB2 database. Only ASCII codepages are supported.

Note: This support is provided only in Windows environments at this time.

Prerequisites

Before the DB2 CONNECT IMPORT command can be used to load initial-state data to a DB2 database, via the Event Replicator Target Adapter, DB2 CONNECT for Windows must be installed. In addition, using the Configuration Assistant from DB2 CONNECT, make sure you configure the alias that should be used as a database name when importing the initial-state data occurs.

Limitations

The following limitations exist in this version:

- This support is provided only in Windows environments at this time.
- Only ASCII codepages are supported.

Enabling DB2 Database Initial-State Data Loading

To enable DB2 database initial-state data loading, complete the following steps:

1 Once all prerequisites have been met, use the Administration tool to configure a target definition for your DB2 database. For more information, read *Configuring Target Definitions for Event Replicator Target Adapter* in the *Event Replicator Target Adapter Administration Guide*.

2 Using the Administration tool, create a specific target processing option definition for your DB2 database. For more information about setting up target processing option definitions, read *Specifying Target Database Processing Option Definitions*, in the *Event Replicator Target Adapter Administration Guide*.

In the target processing option definition for your DB2 database:

Note: No value need be specified in the **Environment File** field.

- Select the **Use Loader** check box (check it).
- Enter the alias name of the DB2 database in the **Oracle Service, Teradata Tdpip, Microsoft SQL Schema Name or DB2 Alias** field.

- Provide a value for the **Path to Loader Executable** field, identifying the full path of the DB2 executable, *db2cmd.exe*.
- Specify any of the following options, as required for your installation, in the Options field.

  When specified, these options must be specified in the order shown in the table and they must be separated by a space. The Event Replicator Target Adapter does not validate these command options; it just passes them to the DB2 CONNECT IMPORT command for processing. For complete information about these options, refer to your *DB2 Command Reference*.

Option	Description
ALLOW WRITE ACCESS	This option causes the Event Replicator Target Adapter to run the import process in online mode. Specify this option only when you want to update your Adabas database at the same time as running an initial-state. It will avoid the database table locks required by the IMPORT command. Otherwise, this option is only required when you import data to a table with a nickname (in which case the COMMITCOUNT option must also be supported with an integer setting, not the literal "AUTOMATIC"). When this option is specified, an intent exclusive (IX) lock on the target table is acquired when the first row is inserted. This allows concurrent readers and writers to access table data. The import operation will periodically commit inserted data to prevent lock escalation to a table lock and to avoid running out of active log space. These commits will be performed even if the COMMITCOUNT option is not specified for the run. During each commit, the import process will lose its IX table lock, and will attempt to reacquire it after the commit.
COMMITCOUNT { *n* | AUTOMATIC }	This option controls when commits of imported records occur. Valid values are an integer representing the number of imported records or the literal "AUTOMATIC". To reduce network overhead, Event Replicator Target Adapter import processes use NOT ATOMIC compound SQL to insert the data, with 100 SQL statements attempted with each import. If your SQL transaction log is not sufficiently large, the import operation will fail. It must be large enough to accommodate the number of rows specified by the COMMITCOUNT option or the number of rows in the data file if COMMITCOUNT is not specified. We therefore recommend that the COMMITCOUNT option be specified to avoid transaction log overflows. When a number (*n*) is specified, the import process performs a commit after that many records have been imported. When compound inserts are used, a user-specified commit frequency of *n* is rounded up to the first integer multiple of the compound count value. When the literal "AUTOMATIC" is specified, the import process internally determines when a commit should be performed. The utility will commit the imported data for one of two reasons: ■ To avoid running out of active log space; or

Option	Description
	■ To avoid a lock escalation from row level to table level. If the ALLOW WRITE ACCESS option is specified, and the COMMITCOUNT option is not specified, the import process will perform commits as if COMMITCOUNT AUTOMATIC had been specified.
WARNINGCOUNT *n*	This option controls when the import operation stops because of warnings. Valid values are integers beginning with zero (0). Set this parameter if no warnings are expected, but you want to verify that the correct file and table are being used. If the import file or the target table is specified incorrectly, the import utility will generate a warning for each row that it attempts to import, which will cause the import to fail. If a value of zero is specified for this option or if this option is not specified, the import operation will continue regardless of the number of warnings issued.
NOTIMEOUT	This option specifies that the import utility will not timeout while waiting for locks.

Save the target processing option definition with the same name as the target definition you created in Step 1.

3 When all processing options are specified, start initial-state processing, directing the output to the Event Replicator Target Adapter via a GFFT you created (using the Data Mapping Tool or the Adabas Event Replicator Subsystem) for the DB2 database. For more information on this, read ***Requesting Initial-State Data***, elsewhere in this guide.. The Event Replicator Target Adapter will collect the initial-state records and load them into your DB2 database.

Processing Output

The output from this processing includes:

- Various DB2 CONNECT IMPORT files used for its processing;
- DB2 CONNECT IMPORT log files from the processing; and
- DAT files containing the initial-state data received by the Event Replicator Target Adapter prior to being imported.

All output files are written to the *\logs* subdirectory of your Event Replicator Target Adapter installation

- Processing Files

Processing Files

The Event Replicator Target Adapter's load processing for DB2 creates two processing files per table loaded. The following processing files are stored in the *\logs* subdirectory of your Event Replicator Target Adapter installation:

- A single *xxxx_yyyymmdd_hhmmss.BAT* file for all DB2 CONNECT processing (where *xxxx* is the base table name, *yyyymmdd* is the date, and *hhmmss* is the time: This .BAT file is used to run the load utility.
- One or more *xxxx_yyyymmdd_hhmmss.DEL* files (where *xxxx* is the table name, *yyyymmdd* and *hhmmss* are the timestamp identifying when the file was written, and DEL indicates the file is a delimited ASCII file). These files contain the data to be loaded to the database tables. These files can take a good deal of space, so make sure that there is enough space to accommodate them and be sure to manually delete them when initial-state processing has completed successfully.

Log Files

Two kinds of log files are produced by Event Replicator Target Adapter import processing for DB2: a primary log file and multiple processing log files. All log files are stored in the *\logs* subdirectory of your Event Replicator Target Adapter installation.

1. The DB2 load processing primary log files make references to the processing log files. The primary log files have names in the format *xxxx_yymmdd_hhmmsss.out*, where *xxxx* is the table name, *yymmdd* is the date of the log file, and *hhmmsss* is the time (to the microsecond) when the log file was last updated.

   We recommend that you review these primary log files carefully to ensure that the import process had no problems restoring primary and foreign keys, indexes, or the unique index of tables. If import processing fails for some reason, we recommend reviewing these primary *.out* log files first when resolving the problem, proceeding then to the processing log files, as necessary.
2. Multiple processing log files are created, one for each import process executed by the Event Replicator Target Adapter. These processing log files have names in the format *xxxx_yyyymmdd_hhmmss.MSG*, where *xxxx* is the table name, *yyyymmdd* and *hhmmss* are the timestamp identifying when the file was written, and MSG indicates the file is a log file.

12 Using the Microsoft SQL Server Bulk Copy (bcp) Utility to Load Initial-State Data

You can use the bcp utility to process and load initial-state data into a Microsoft SQL Server database. The bcp utility can only be used with the Event Replicator Target Adapter to load initial-state data; it cannot be used for transactional data.

Note: The bcp utility is supported in Windows environments only. In addition, composite keys are not supported when using the bcp utility with Event Replicator Target Adapter.

Prerequisites

The following prerequisites must be met before the bcp utility can be used to load initial-state data to a Microsoft SQL Server database, via the Event Replicator Target Adapter:

- The Microsoft SQL Server utilities supporting the bcp utility must be installed on the same machine as the Event Replicator Target Adapter. The bcp utility version 9 or greater must be installed.
- Be sure there is enough space to accommodate the *.BCPDAT files that will be created by this processing.

Limitations

You can use the bcp utility to load initial-state data only if:

- Composite keys are *not* used in the data.
- You are running Event Replicator Target Adapter, Microsoft SQL Server, and the bcp utility in Windows environments.

Enabling bcp Utility Initial-State Processing

To enable bcp utility initial-state processing, complete the following steps:

1. Using the Administration tool, configure a target definition for your Microsoft SQL Server database. For more information, read *Configuring Target Definitions for Event Replicator Target Adapter* in the *Event Replicator Target Adapter Administration Guide*.
2. Using the Administration tool, create a specific target processing option definition for your Microsoft SQL Server database.

- In the Loader options area, check the box **Use Loader** and provide a value for the **Path to Loader Executable** field, identifying the full path of the Microsoft SQL Server bcp utility executable. For example:

```
C:\Program Files\Microsoft SQL Server\90\Tools\Binn\bcp.exe
```

- Optionally, in the Loader options area of the target processing option definition panel, specify the MS SQL Server schema name if your schema name is different from the MS SQL Server default schema name. Prior to the release of MS SQL Server 2005, all tables created by MS SQL Server used the user login name as the default schema name; in MS SQL Server 2005 and later versions, Microsoft uses "dbo" as the default schema name (which is different from the user login name). If you have used any MS SQL Server schema name that is not the same as the MS SQL Server default schema name, specify your schema name in the **Oracle Service or Teradata Tdpip** field.
- Specify the following options, as required for your installation, in the **Options** field.

Option	Description
-b *batchsize*	Use the -b (batch size) option in this field to specify the number of rows per batch of imported data. Each batch is imported and logged as a separate transaction that imports the whole batch before being committed. By default, all rows in the data file are imported as one batch. To distribute the rows among multiple batches, use the -b option to specify a batch size that is smaller than the number of rows in the data file. If the transaction for any batch fails, only insertions from the current batch are rolled back. Batches already imported by committed transaction are unaffected by a later failure. Do not use this option in conjunction with the -h ("ROWS_PER_BATCH=bb") option.

Note: The **Environment File**, and **Use Stream** fields in the Loader options area have no effect on the bcp utility, so they should not be specified.

Save the target processing option definition with the same name as the target definition you created in Step 1.

3 When all processing options are specified, start initial-state processing, directing the output to the Event Replicator Target Adapter via a GFFT you created (using the Data Mapping Tool or the Adabas Event Replicator Subsystem) for the Microsoft SQL Server database. For more information on this, read *Requesting Initial-State Data*, elsewhere in this guide.. The Event Replicator Target Adapter will collect the initial-state records and send them to the bcp utility to be loaded to your database.

Processing Output

The output from this processing includes:

- Various bcp utility processing files (*.FMT* files)
- bcp utility log files from the processing (*.BCPERR* and *.BCPOUT* files)
- Binary data files (*.BCPDAT* files) containing the initial-state data received by the Event Replicator Target Adapter prior to being processed by the bcp utility, and possibly *.DBGDAT* files (if the debug logging option is turned on)
- An output file (*.out* file) containing output messages from the final loader process run.

All output files are written to the *\logs* subdirectory of your Event Replicator Target Adapter installation

Processing Files

The Event Replicator Target Adapter's bcp utility processing creates three processing files per table loaded. The following processing files are stored in the *\logs* subdirectory of your Event Replicator Target Adapter installation:

- A single *xxxx_yyyymmdd_hhmmss.BAT* file for all bcp utility processing (where *xxxx* is the base table name, *yyyymmdd* is the date, and *hhmmss* is the time: This .BAT file is used to run the bcp utility. It generates the format file (*.FMT) for each table and loads the binary *.BCPDAT file into the database.
- One `xxxx_yyyymmdd_hhmmss.FMT` file for each table (where *xxxx* is the table name): This is the format file for the bcp utility processing.
- One or more *xxxx_yyyymmdd_hhmmss.BCPDAT* file for each table (where *xxxx* is the table name, *yyyymmdd* is the date, and *hhmmss* is the time): These files contain the data to be loaded to the database tables by the bcp utility. These files are in compressed binary format, but can still take a good deal of space, so make sure that there is enough space to accommodate them and be sure to manually delete them when initial-state processing has completed successfully.
- If the file debug logging option is turned on, the resulting debugging information is stored in a debug file with a name in the format *xxxx_yyyymmdd_hhmmss.DBGDAT*, where *xxxx* is the table name, *yyyymmdd* is the date of the debug file, and *hhmmss* is the time the debug file was last updated. Multiple debug files may be produced for a single table.

Log Files

Several kinds of log files are produced by Event Replicator Target Adapter bcp utility processing: a primary log file and multiple processing log files. All log files are stored in the *\logs* subdirectory of your Event Replicator Target Adapter installation.

1. The bcp utility primary log file makes references to the processing log files. The primary log file has a name in the format *xxxx_yyyymmdd_hhmmss_nnn.out*, where *xxxx* is the base table name, *yyyymmdd* is the date of the log file, and *hhmmss_nnn* is the time (hours, minutes, seconds and milliseconds) the log file was last updated. A single primary log file is created for all bcp utility processing. If bcp utility processing fails for some reason, we recommend reviewing these primary *.out log files first when resolving the problem, proceeding then to the processing log files, as necessary.
2. Multiple processing log files are created, one for each bcp process executed by the Event Replicator Target Adapter. These processing log files have names in the format *xxxx_yyyymmdd_hhmmss.BCPOUT*, where *xxxx* is the table name, *yyyymmdd* is the processing date (year, month, and day) and *hhmmss* represents the processing time (hours, minutes, and seconds).
3. Any errors encountered during bcp utility processing will be written to error files with names in the format *xxxx_yyyymmdd_hhmmss.BCPERR*, where *xxxx* is the table name, *yyyymmdd* is the processing date (year, month, and day) and *hhmmss* represents the processing time (hours, minutes, and seconds). A single*.BCPERR file is created for each table.

13 Using the Oracle SQL*Loader to Load Initial-State Data

You can use the Oracle SQL*Loader to process and load initial-state data into an Oracle database. This process is supported in two modes: file mode and stream mode:

- In file mode, initial-state data is received by the Event Replicator Target Adapter and stored in a DAT file prior to being sent to the Oracle SQL*Loader for processing. For more information about the DAT files created as part of this processing, read ***Processing Output***, elsewhere in this chapter
- In stream mode, initial-state data is received by the Event Replicator Target Adapter and stored in memory prior to being sent to the Oracle SQL*Loader for processing. No DAT files are created.

Note: The Oracle SQL*Loader can only be used with the Event Replicator Target Adapter to load initial-state data; it cannot be used for transactional data.

Prerequisites

The following prerequisites must be met before the Oracle SQL*Loader can be used to load initial-state data to an Oracle database, via the Event Replicator Target Adapter.

- The Oracle utilities supporting the Oracle SQL*Loader must be installed on the same machine as the Event Replicator Target Adapter.
- The Oracle Net Manager must be installed and configured if Oracle is running on a different machine than the Event Replicator Target Adapter.
- If file mode is requested, be sure there is enough space to accommodate the DAT files that will be created by this processing.

Enabling Oracle SQL*Loader Initial-State Processing

To enable Oracle SQL*Loader initial-state processing:

1 Using the Administration tool, configure a target definition for your Oracle database. For more information, read *Configuring Target Definitions for Event Replicator Target Adapter*, in the *Event Replicator Target Adapter Administration Guide*.

2 Using the Administration tool, create a specific target processing option definition for your Oracle database. For this definition you must:

   - Select the **Use Loader** check box (check it).
   - Provide a value for the **Path** field, identifying the full path of the Oracle SQL*Loader executable.

- Specify the `rows` option in the **Options** field, specifying the number of rows you want to read from the data file before a data save is performed by the SQL*Loader.
- Save the target processing option definition with the same name as the target definition you created in Step 1.

You should also evaluate and optionally specify values for the following fields in this target processing option definition:

- **Service**, which specifies the Oracle database service name that should be used with the Oracle SQL*Loader.
- **Environment File**, which specifies the file name containing the Oracle environment variables that should be used with the Oracle SQL*Loader.
- **Use Stream**, which indicates whether processing should be performed in stream mode or not.

For more information about these options, read *Specifying Target Database Processing Option Definitions*, in the *Event Replicator Target Adapter Administration Guide*.

3 When all processing options are specified, start initial-state processing, directing the output to the Event Replicator Target Adapter via a GFFT you created (using the Data Mapping Tool or the Adabas Event Replicator Subsystem) for the Oracle database.

For more information on this, read ***Requesting Initial-State Data***, elsewhere in this guide.

The Event Replicator Target Adapter will collect the initial-state records and send them to the Oracle SQL*Loader to be loaded to your Oracle database.

Note: Some errors might prevent the Oracle SQL*Loader from restoring primary and foreign keys, indexes, and the unique index of tables. We recommend that you examine the Oracle SQL*Loader log files to determine if such an error occurred. For more information, read ***Processing Output***, elsewhere in this section.

Processing Output

The output from this processing includes:

- Various Oracle SQL*Loader files used for its processing
- Oracle SQL*Loader log files from the processing
- DAT files containing the initial-state data received by the Event Replicator Target Adapter prior to being processed by the Oracle SQL*Loader.

All output files are written to the *\logs* subdirectory of your Event Replicator Target Adapter installation

Log Files

The Oracle SQL*Loader log files are stored in files that have names in the format *xxxx_yymmdd_hhmmss*.log, where *xxxx* is the table name, *yymmdd* is the date of the log file, and *hhmmss* is the time the log file was last written to.

We recommend that you review this log file carefully to ensure that the Oracle SQL*Loader had no problems restoring primary and foreign keys, indexes, or the unique index of tables.

DAT Files

DAT files are only created by the Event Replicator Target Adapter if you have selected Oracle SQL*Loader processing in file mode. If you select stream mode, these files are not created. These files can take a good deal of space, so make sure that there is enough space to accommodate them and be sure to manually delete them when initial-state processing has completed successfully.

The format of the DAT file names is *xxxx_yymmdd_hhmmss*.dat, where *xxxx* is the table name, *yymmdd* is the date of the DAT file, and *hhmmss* is the time the DAT file was last written to.

14 Using the PostgreSQL psql Utility to Load Initial-State Data

You can use the PostgreSQL psql utility to process and load initial-state data into a PostgreSQL database.

Note: The PostgreSQL psql utility can only be used with the Event Replicator Target Adapter to load initial-state data; it cannot be used for transactional data.

This document covers the following topics:

Prerequisites

The following prerequisite must be met before the PostgreSQL psql utility can be used to load initial-state data to a PostgreSQL database, via the Event Replicator Target Adapter.

- The PostgreSQL utility "psql" (psql.exe on Windows, psql on Linux) must be installed on the same machine as the Event Replicator Target Adapter.

Enabling PostgreSQL SQL Initial-State Processing using psql

To enable PostgreSQL initial-state processing using psql:

1. Using the Administration tool, configure a target definition for your PostgreSQL database. For more information, read *Configuring Target Definitions for Event Replicator Target Adapter* in the *Event Replicator Target Adapter Administration Guide*.
2. Using the Administration tool, create a specific target processing option definition for your PostgreSQL database. For this definition you must:
   - Select the **Use Loader** check box (check it).
   - Provide a value for the Path field, identifying the full path of the PostgreSQL *psql* executable.
   - Save the target processing option definition **with the same name** as the target definition you created in Step 1.

     For more information about these options, read *Specifying Target Database Processing Option Definitions*, in the *Event Replicator Target Adapter Administration Guide*.
   - When all processing options are specified, start initial-state processing, directing the output to the Event Replicator Target Adapter via a GFFT you created (using the Data Mapping Tool or the Adabas Event Replicator Subsystem) for the PostgreSQL database.

     For more information on this, read ***Requesting Initial-State Data***, elsewhere in this guide..

     The Event Replicator Target Adapter will collect the initial-state records and send them to the psql utiltiy to be loaded to your PostgreSQL database.

Processing Output

The output from this processing includes:

- Output files containing the output from the psql utility.
- psql utility log files.
- Data files containing the initial-state data received by the Event Replicator Target Adapter prior to being processed by the PostgreSQL psql utility.

All output files are written to the \logs subdirectory of your Event Replicator Target Adapter installation

This section covers the following topics:

Log Files

The PostgreSQL psql log files are stored in files that have names in the format *xxxx_yymmdd_hhmmss*.log, where *xxxx* is the table name, *yymmdd* is the date of the log file, and *hhmmss* is the time the log file was last written to. A similarly names file with the extention .out contain the actual output of the psql utility.

In the event of errors received during processing of initial-state data using the psql utility, please examine these files for hints as to the cause of the error.

Processing Files

A single processing file for each table with the extension .sql is created by the Event Replicator Target Adapter to feed into the psql utiltiy. This files can take a good deal of space, so make sure that there is enough space to accommodate it and be sure to manually delete it when initial-state processing has completed successfully. This file contains not only the data loaded into the table(s), but also the actual psql commands passed to the utility.

The format of the processing file name is *xxxx_yymmdd_hhmmss*.sql, where *xxxx* is the table name, *yymmdd* is the date of the file, and *hhmmss* is the time the sql data file was last written to.

15 Using the Teradata MultiLoad (MLoad) Utility to Load Initial-State Data

You can use the Teradata MultiLoad (MLoad) utility to process and load initial-state data into a Teradata database. The MLoad utility can only be used with the Event Replicator Target Adapter to load initial-state data; it cannot be used for transactional data. It can load UTF8 data.

Note: The MLoad utility is supported in Windows environments only. In addition, composite keys are not supported when using the MLoad utility with Event Replicator Target Adapter.

This chapter covers the following topics:

Prerequisites

The following prerequisites must be met before the Teradata MLoad utility can be used to load initial-state data to a Teradata database, via the Event Replicator Target Adapter:

- The Teradata utilities supporting the MLoad utility must be installed on the same machine as the Event Replicator Target Adapter.
- Be sure there is enough space to accommodate the DAT files that will be created by this processing.
- After installing the Teradata MLoad utility, it must be configured. According to Teradata MLoad utility configuration rules, you should add entries in the Windows HOSTS file that map the TDPIDs (Teradata Director Program Identifier) with the Teradata databases and their hosts. When the MLoad utility is invoked for a particular target, the Event Replicator Target Adapter will relay the TDPID provided in the Loader section of the Target Database Options entry definition (in Event Replicator Target Adapter Administration) into the Teradata .login command. For example:

```
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#

127.0.0.1       localhost
n.n.n.n         tdpid tdpidcop1
```

The following substitutions are used in this example:

Term	Description
n.n.n.n	The specific TCP/IP address for the named TDPID where the Teradata database expected to interact with the MLoad utility is running.
tdpid	The Teradata Director Program Identifier (TDPID); usually the first eight characters of the Teradata database name or the Teradata service name. Teradata database names can be up to 30 characters long. However, MLoad restrictions for Teradata releases up to Version 12 require that TDPIDs be eight (8) characters or less.

Term	Description
	Event Replicator Target Adapter current logic dynamically determines the TDPID using the first eight characters of the Teradata service name specified in the Loader options of the database option definition created using the Event Replicator Target Adapter Administration tool. If you do not supply a service name, the first eight characters of the Teradata database name is used. Whatever TDPID is used must also be present in the HOSTS file. Be sure that the specifications in the database option definition and the specification in the HOSTS file match.
tdpidcop1	Teradata requires that you specify a second parameter in the format *tdpid*cop1, where *tdpid* is the TDPID you specified above and "cop1" is a literal. For example, if you specify "A2345678" as your TDPID, you would specify "A2345678cop1" for this parameter.

Limitations

You can only use the MLoad utility to load initial-state data if:

- Composite keys are *not* used in the data.
- You are running Event Replicator Target Adapter, Teradata, and the MLoad utility in Windows environments.

Enabling Teradata MLoad Utility Initial-State Processing

To enable Teradata MLoad utility initial-state processing, complete the following steps:

1 Using the Administration tool, configure a target definition for your Teradata database. For more information, read *Configuring Target Definitions for Event Replicator Target Adapter* in the *Event Replicator Target Adapter Administration Guide*.

Note: If you intend to load UTF-8 data to your Teradata database, be sure you have applied the appropriate Event Replicator Target Adapter hot fix and that you have specified the "CHARSET=utf8" parameter in the data source URL. For more information about the hot fixes that support this, refer to the README file provided with Event Replicator Target Adapter.

2 Using the Administration tool, create a specific target processing option definition for your Teradata database. For this definition you must:

- Select the **Convert Hyphen** option (check it) on the target processing option definition panel.

- In the Loader options area of the target processing option definition panel, select the **Use Loader** check box (check it).
- In the Loader options area of the target processing option definition panel, specify the Teradata Director Program Identifier (TDPID) in the **Oracle Service or Teradata Tdpip** field. This is usually the first eight characters of the Teradata database name or the Teradata service name. Teradata database names can be up to 30 characters long. However, MLoad restrictions for Teradata releases up to Version 12 require that TDPIDs be eight (8) characters or less. Event Replicator Target Adapter current logic dynamically determines the TDPID using the first eight characters of the name specified in this field. If you do not supply a name, the first eight characters of the Teradata database name is used. Whatever TDPID is used must also be present in the HOSTS file.
- In the Loader options area, provide a value for the **Path to Loader Executable** field, identifying the full path of the Teradata MLoad utility executable. For example:

```
C:\Program Files\NCR\teradata client\bin\mload.exe
```

Save the target processing option definition with the same name as the target definition you created in Step 1.

3 When all processing options are specified, start initial-state processing, directing the output to the Event Replicator Target Adapter via a GFFT you created (using the Data Mapping Tool or the Adabas Event Replicator Subsystem) for the Teradata database.

For more information on this, read *Requesting Initial-State Data*, elsewhere in this guide..

The Event Replicator Target Adapter will collect the initial-state records and send them to the Teradata MLoad utility to be loaded to your Teradata database.

Note: Some errors might prevent the MLoad utility from restoring primary and foreign keys, indexes, and the unique index of tables. We recommend that you examine the MLoad utility log files to determine if such an error occurred. For more information, read *Processing Output*, next in this section.

Processing Output

The output from this processing includes:

- Various MLoad utility processing files
- MLoad utility log files from the processing
- Data files (*.mlddat* files) containing the initial-state data received by the Event Replicator Target Adapter prior to being processed by the MLoad utility.

All output files are written to the *\logs* subdirectory of your Event Replicator Target Adapter installation.

Processing Files

The Event Replicator Target Adapter's MLoad utility processing creates three processing files per table loaded. The following processing files are stored in the *\logs* subdirectory of your Event Replicator Target Adapter installation:

- *xxxx.bat* (where *xxxx* is the table name) file: This *.bat* file is used to run the MLoad utility.
- *xxxx.mldctl* (where *xxxx* is the table name) file: This is the control file for the MLoad utility processing.
- One or more *xxxx.mlddat* (where *xxxx* is the table name) files: These files contain the data to be loaded to the database tables by the MLoad utility. These files can take a good deal of space, so make sure that there is enough space to accommodate them and be sure to manually delete them when initial-state processing has completed successfully.

Log Files

Two kinds of log files are produced by Event Replicator Target Adapter MLoad utility processing: a primary log file and multiple processing log files. All log files are stored in the *\logs* subdirectory of your Event Replicator Target Adapter installation.

1. The Teradata MLoad utility primary log files make references to the processing log files. The primary log files have names in the format *xxxx_yymmdd_hhmmss.out*, where *xxxx* is the table name, *yymmdd* is the date of the log file, and *hhmmss* is the time the log file was last updated. We recommend that you review these primary log files carefully to ensure that the MLoad utility had no problems restoring primary and foreign keys, indexes, or the unique index of tables. If MLoad utility processing fails for some reason, we recommend reviewing these primary *.out* log files first when resolving the problem, proceeding then to the processing log files, as necessary.
2. Multiple processing log files are created, one for each MLoad process executed by the Event Replicator Target Adapter. These processing log files have names in the format *Multiloadprocess-yyyy-mm-dd-hh-mm-ss-nnnn.log*, where *yyyy-mm-dd* is the processing date (year, month, and day) and *hh-mm-ss-nnnn* represents the processing time (hours, minutes, seconds, and milliseconds).

16 Requesting RDBMS Data Refreshes

Using the Adabas Event Replicator Subsystem , you can manually submit requests to the Event Replicator Target Adapter that delete the data in all relational database tables associated with the request, but leave the tables in place. This effectively clears the tables. For more information on using the Adabas Event Replicator Subsystem to do this, read *Clearing (Refreshing) the RDBMS Table Data* in *Adabas Event Replicator Subsystem User's Guide*.

17 Dropping RDBMS Tables

Using the Adabas Event Replicator Subsystem, you can manually submit requests to the Event Replicator Target Adapter that delete the data and remove the tables associated with the request from the relational database. For more information on using the Adabas Event Replicator Subsystem to do this, read *Deleting (Dropping) RDBMS Tables* in *Adabas Event Replicator Subsystem User's Guide*.

18 Replicating ADALOD Data to the RDBMS Tables

You can use the LOAD and UPDATE functions of the Adabas ADALOD utility to add and update data in the RDBMS tables. In both cases, you must specify RPLLOAD=YES when running the utility to ensure that the loaded data is automatically replicated.

The processing of ADALOD data for Event Replicator Target Adapter is the same as the processing of regularly replicated data for Event Replicator Target Adapter. In other words, the data that is loaded is processed by subscriptions; if a subscription is activated, includes a generated global format buffer field table (GFFT) for the Adabas file being loaded or updated, and includes an activated destination with a destination class type of "SAGTARG", the data is replicated and sent as an XML message to the Event Replicator Target Adapter for processing.

Before you decide to add data to the RDBMS tables using the ADALOD utility, consider the following potential problems:

- All records processed by ADALOD and sent to the Event Replicator Target Adapter with action types of "Insert" are treated by the Event Replicator Target Adapter as if they were new records, unless the RESEND flag associated with the incoming data from Event Replicator for Adabas has been set to YES. If the RESEND flag is set to YES, checks are performed to determine whether a record already exists with the same primary key. If it does, the existing row is deleted and a new one is inserted for the new incoming data.

  Note: If, during Insert processing, the RESEND flag is set to NO and a record with the same primary key as an existing record is found in the RDBMS tables, Event Replicator Target Adapter processing will fail.

- Performance problems can occur due to the massive numbers of records that might be sent if you use the ADALOD utility and the Event Replicator Target Adapter to add records to the RDBMS tables. A single subscription activated for Event Replicator Target Adapter processing can update multiple RDBMS tables simultaneously (depending on the number of pertinent SFILE definitions and the number of different generated global format buffer field tables (GFFTs)

that exist in the subscription). If multiple subscriptions are activated for Event Replicator Target Adapter processing, then, the amount of traffic on your system could be quite high.

To limit the number of ADALOD records replicated through the Event Replicator Target Adapter, you can deactivate some of the subscriptions, use transaction filter definitions, or elect not to drop an RDBMS table prior to the ADALOD utility run (in this case, all insert transaction will fail if the primary key already exists for that table).

19 Managing Event Replicator Target Adapter Log Files and Console Messages

Log File Location

All Event Replicator Target Adapter log files are stored in the *\logs* subdirectory of your Event Replicator Target Adapter installation. The Event Replicator Target Adapter log file name is *sqlrep.log*, which contains all current log messages. If you have a problem, please keep this log file.

In addition, the following additional log files may be found at this location. If you have a problem, please keep these log files as well:

- Statistics about the Event Replicator Target Adapter session are stored in log files with names in the format: `ArtResource_yyyymmdd_hhmmss.log`, where the date and time are used to make unique log files for each session.
- Metrics up to the last transaction processed during the session are stored in log files with names in the format: `ArtMetrics_yyyymmdd_hhmmss.log`, where the date and time are used to make unique log files for each session.
- If you requested that webMethods EntireX or WebSphere MQ tracing occur when you configured the source definitions for Event Replicator Target Adapter, the *etb_trace.log* file (for webMethods EntireX) or the *mqs_trace.log* file (for WebSphere MQ) will be created at this location.
- If you requested that all incoming Event Replicator messages be captured for debugging purposes when you configured the source definitions for Event Replicator Target Adapter, an *etb_capture.log* file (for webMethods EntireX) or an *mqs_capture.log* file (for WebSphere MQ) will be created at this location.

Reviewing the sqlrep.log File

On Windows systems, the *sqlrep.log* file can be reviewed using a text editor. On UNIX systems, it can be reviewed by running the *sqlreplog.sh* script. You can control the level of log messages written to the *sqlrep.log* file; read ***Setting Log Levels***, elsewhere in this section, for more information.

sqlrep.log File Retention

Each time you start up the Event Replicator Target Adapter, a new *sqlrep.log* file is created. The previously-existing log file will be renamed *sqlrep.log.1*. All previous log files are also retained and renamed, so that the lowest numbered file is the most recent historical log file (the *sqlrep.log* file with no number is always the current log file). In other words, if an *sqlrep.log* and an *sqlrep.log.1* file already exist when the Event Replicator Target Adapter is started, the following sequence of events occurs:

1. File *sqlrep.log.1* is renamed to *sqlrep.log.2*.

2. File *sqlrep.log* is renamed to *sqlrep.log.1*.
3. A new *sqlrep.log* file is created for the new Event Replicator Target Adapter session.

When an *sqlrep.log* file reaches 10MB in size, it is renamed *sqlrep.log.1*, and all previous log files are retained and renamed as well, as described above.

A maximum of 99,999 log files can be retained. When this maximum is reached and a new *sqlrep.log* file is created, the *sqlrep.log.99999* file is overwritten with data from the *sqlrep.log.99998* log file.

Windows and UNIX Event Logs

When the Event Replicator Target Adapter is installed, all messages that display to the console are automatically written to the Windows or UNIX system event logs (as appropriate). You can control the level of log messages written to the Windows and UNIX event logs; read ***Setting Log Levels***, elsewhere in this section, for more information.

Setting Log Levels

You can set the log level of messages written to the *sqlrep.log* log file, the Windows and UNIX event logs, and to the console. Three log levels are available (listed in order of settings that result in the most logging to those resulting in the least logging):

Log Level	Description
DEBUG	Only debugging, informational, warning, error, and fatal messages are logged or displayed. **Note:** Setting this log level can impact performance.
INFO	Only informational, warning, error, and fatal messages are logged or displayed.
ERROR	Only error and fatal messages are logged or displayed.

To change the log level of messages sent to the console:

1 Edit the *log4j.properties* file. This file is located in the *\webapps\sqlrep\WEB-INF\classes* directory wherever you installed Event Replicator Target Adapter.

2 Locate the lines in the *log4j.properties* that read:

```
log4j.category.com.softwareag.ada=loglevel,appender-name1,appender-namen
.
.
.
log4j.appender.CONSOLEAppender.Threshold=loglevel
```

Note: These lines will not be next to each other in the file.

These lines in the file control the level of Event Replicator Target Adapter log messages sent to the console. By default, the log level is set to ERROR.

3 Change the log level as needed. Valid log level values are ERROR, INFO, and DEBUG, as described earlier in this section. For example, the following settings would display informational and error messages on the console.

```
log4j.category.com.softwareag.ada=DEBUG, CONSOLEAppender
.
.
.
log4j.appender.CONSOLEAppender.Threshold=INFO
```

Note: Setting the log level to DEBUG can impact performance.

To change the log level of messages written to the *sqlrep.log* file:

1 Edit the *log4j.properties* file. This file is located in the *\webapps\sqlrep\WEB-INF\classes* directory wherever you installed Event Replicator Target Adapter.

2 Locate the lines in the *log4j.properties* that read:

```
log4j.category.com.softwareag.ada=loglevel,appender-name1,appender-namen
.
.
.
log4j.appender.FILEAppender.Threshold=loglevel
```

Note: These lines will not be next to each other in the file.

These lines in the file control the level of Event Replicator Target Adapter log messages written to the *sqlrep.log* file. By default, the log level is set to ERROR.

3 Change the log level as needed. Valid log level values are ERROR, INFO, and DEBUG, as described earlier in this section. For example, the following settings would write all error, informational, and debugging Event Replicator Target Adapter messages to the *sqlrep.log* file.

```
log4j.category.com.softwareag.ada=DEBUG, FILEAppender
.
.
.
log4j.appender.FILEAppender.Threshold=DEBUG
```

Note: Setting the log level to DEBUG can impact performance.

To change the log level of messages written to the Windows or UNIX system event logs:

1 Edit the *log4j.properties* file. This file is located in the *\webapps\sqlrep\WEB-INF\classes* directory wherever you installed Event Replicator Target Adapter.

2 In Windows environments, locate the lines in the *log4j.properties* that read:

```
log4j.category.com.softwareag.ada=loglevel, appender-name1,appender-namen
.
.
.
log4j.appender.NTAppender.Threshold=loglevel
```

Note: These lines will not be next to each other in the file.

These lines in the file control the level of Event Replicator Target Adapter log messages written to the Windows event log. By default, the log level is set to ERROR.

In UNIX environments, locate the line in the *log4j.properties* that reads:

```
log4j.appender.UNIXAppender.Threshold=loglevel
```

This line in the file controls the level of Event Replicator Target Adapter log messages written to the UNIX system event log. By default, the log level is set to ERROR.

3 Change the log level as needed. Valid log level values are ERROR, INFO, and DEBUG, as described earlier in this section. For example, the following setting would write all error, informational, and debugging Event Replicator Target Adapter messages to the Windows event log.

```
log4j.category.com.softwareag.ada=DEBUG, NTAppender
.
.
.
log4j.appender.NTAppender.Threshold=DEBUG
```

Note: Setting the log level to DEBUG can impact performance.

4 This step pertains to UNIX systems only. If you are working on a Windows system, you can ignore this step.

In UNIX environments, an additional step must be taken that has system-wide implications. The UNIX *syslog.conf* file controls the level of logging for the entire UNIX system, so any log level settings set by the Event Replicator Target Adapter for the UNIX system event log (in the previous step) may be overridden by the *syslog.conf* settings. To fully change the log level settings, you must have your UNIX system administrator update the *syslog.conf* file as follows and restart the daemon:

To set the log level to:	Update *syslog.conf* to include an entry for:
DEBUG	`user.debug`
INFO	`user.info`
ERROR	`user.err`

The *syslog.conf* entries should direct these message levels to appropriate UNIX system directories. For complete information on updating the *syslog.conf* file, refer to your UNIX documentation.

20 Controlling Event Replicator for Adabas Destination Definitions

Event Replicator Target Adapter provides two commands (batch utilities) you can use to activate, deactivate, open, or close a destination definition in the Event Replicator for Adabas.

These commands are located in the *bin* directory, wherever the Event Replicator Target Adapter source code and program files are installed.

- Use the `etbdest` command to control your webMethods EntireX destination definitions.
- Use the `mqsdest` command to control your WebSphere MQ destination definitions.

From the Target Adapter Command Prompt, enter the appropriate command, using one of the following syntaxes:

```
etbdest -bbrokerID -sclass/server/service -ooperation -ddestination-name -ttracelevel ↩
(-mscm -uuserid -kusertoken -ppassword)
```

or

```
mqsdest  -hhost -pport -mqueuemgr -cchannel -qqueuename -ooperation -ddestinationname ↩
-ttracelevel
```

The parameters in this syntax can be specified in any order because each parameter is identified by a parameter identifier. Note that there are no spaces between the parameter identifier and the value of the parameter, although a single space separates each parameter. Each parameter identifier is described in the following table:

Parameter Identifier	Specify
-b	The webMethods EntireX Broker ID.
-c	The case-sensitive WebSphere MQ channel.
-d	The name of the Event Replicator for Adabas destination you want to control.
-h	The WebSphere MQ host name.
-k	User Token if required for Single Conversation Mode communication with Broker.
-m	(etbdest) Mode. Follow with 'scm' (no quotes) if the input queue is configured for Single Conversation Mode.
-m	(mqsdest) The case-sensitive WebSphere MQ queue manager name.
-o	The controlling operation you want to issue for the destination. Valid operation are OPND (open the destination) and CLSD (close the destination).
-p	(etbdest) Password if Broker is running with security.
-p	The WebSphere MQ port number.
-q	The case-sensitive WebSphere MQ queue name.
-s	The case-sensitive class, server, and service name (separated by slashes) of the webMethods EntireX Broker.
-t	The trace level that should be used for the attempt. Valid values range from "0" (no trace) through "5". The default is "0".
-u	Broker userid if required for Single Conversation Mode.

If errors occur when you run *etbdest* or *mqsdest*, rerun them using a trace level setting of "3". Then submit the trace log to your webMethods EntireX or WebSphere MQ administrators to identify the problem. For more information about Event Replicator Target Adapter log files, read ***Managing Event Replicator Target Adapter Log Files and Console Messages***, in the *Event Replicator Target Adapter User Guide*.

21 Samples

Generated GFFT and Schema Sample

The sample in this section shows a generated field table (GFFT) and its corresponding XML schema.

Generated Field Table (GFFT)

The field table is created when you generate a global format buffer for a subscription in Event Replicator for Adabas. It is generated from Predict file definitions. To generate a global format buffer and field table (GFFT) using the Adabas Event Replicator Subsystem, read *Generating a GFB* in *Adabas Event Replicator Subsystem User's Guide*.

```
I T L DB Name                              F       Leng S D Remark
- - - -- -------------- top -------------- - ---------- - - -----------

    1 AA PERSONNEL-ID                      A          8   D
  G 1 AB FULL-NAME
    2 AC FIRST-NAME                        A         20 N
    2 AD MIDDLE-I                          A          1 N
    2 AE NAME                              A         20   D
    1 AF MAR-STAT                          A          1 F
  *      M=MARRIED
  *      S=SINGLE
  *      D=DIVORCED
  *      W=WIDOWED
    1 AG SEX                               A          1 F
  G 1 A1 FULL-ADDRESS
    2 AJ CITY                              A         20 N D
    2 AK ZIP                               A         10 N
    2 AL COUNTRY                           A          3 N
  G 1 A2 TELEPHONE
    2 AN AREA-CODE                         A          6 N
    2 AM PHONE                             A         15 N
    1 AO DEPT                              A          6   D
    1 AP JOB-TITLE                         A         25 N D
  G 1 A3 LEAVE-DATA
    2 AU LEAVE-DUE                         N        2.0
    2 AV LEAVE-TAKEN                       N        2.0 N
```

Schema and Replicated Data

The XML schema for the data is created from the generated field table (GFFT) for the subscription that processed the replicated data. It is provided in the Create operation in the sample below. Insert, Update, and Delete operations for the replicated data are also shown in the sample. The Create operation will create the SQL table in the RDBMS, the Insert operation will insert replicated data into the created SQL table, the Update operation will update replicated data in the SQL table, and the Delete operation will delete replicated data in the SQL table. The name of the table created is determined by combining the name of the subscription (PAYROLL in this sample) with the name of the schema file (EMPLOYEES in this sample): PAYROLL_EMPLOYEES. The names of the columns in the table are taken from the long names in the field table.

Note: If no primary key specified, the ISN will be used as the primary key. The primary key must be defined as an Adabas Unique Descriptor.

```
EXAMPLE "Create":

<TransactionPart Sub='SUB1' Origin='I' Prefix='N' Part='1' ↩
Id='C0CF513C08C1F20400000000' En='4091'>
 <Event Row='1' File='EMPLOYEES-FILE' Op='Create' Time='2007/06/27-15:20:42.444665' ↩
FNR='10'>
  <Field name='PERSONNEL-ID' ei='1' fmt='string' prc='0' len='8' key='uk'/>
  <Field name='FIRST-NAME' ei='2' fmt='string' prc='0' len='20'/>
  <Field name='NAME' ei='3' fmt='string' prc='0' len='20' key='ky'/>
  <Field name='MIDDLE-NAME' ei='4' fmt='string' prc='0' len='20'/>
  <Field name='MAR-STAT' ei='5' fmt='string' prc='0' len='1'/>
  <Field name='SEX' ei='6' fmt='string' prc='0' len='1'/>
  <Field name='BIRTH' ei='8' fmt='date' prc='0' len='10' key='ky'/>
  <Field name='ADDRESS-LINE' ei='9' fmt='string' prc='0' len='20' att='mu'/>
  <Field name='CITY' ei='10' fmt='string' prc='0' len='20' key='ky'/>
  <Field name='POST-CODE' ei='11' fmt='string' prc='0' len='10'/>
  <Field name='COUNTRY' ei='12' fmt='string' prc='0' len='3'/>
  <Field name='AREA-CODE' ei='13' fmt='string' prc='0' len='6'/>
  <Field name='PHONE' ei='14' fmt='string' prc='0' len='15'/>
  <Field name='DEPT' ei='15' fmt='string' prc='0' len='6' key='ky'/>
  <Field name='JOB-TITLE' ei='16' fmt='string' prc='0' len='25' key='ky'/>
  <Field name='CURR-CODE' ei='18' fmt='string' prc='0' len='3' att='pe' ↩
gname='INCOME'/>
  <Field name='SALARY' ei='19' fmt='decimal' prc='0' len='10' att='pe' ↩
gname='INCOME'/>
  <Field name='BONUS' ei='20' fmt='decimal' prc='0' len='10' att='mu' gname='INCOME'/>

  <Field name='LEAVE-DUE' ei='21' fmt='decimal' prc='0' len='2'/>
  <Field name='LEAVE-TAKEN' ei='22' fmt='decimal' prc='0' len='2'/>
  <Field name='LEAVE-START' ei='24' fmt='decimal' prc='0' len='8' att='pe' ↩
gname='LEAVE-BOOKED'/>
  <Field name='LEAVE-END' ei='25' fmt='decimal' prc='0' len='8' att='pe' ↩
gname='LEAVE-BOOKED'/>
  <Field name='LANG' ei='26' fmt='string' prc='0' len='3' att='mu'/>
    <Alias name='INCOME' ei='17'/>
```

```
    <Alias name='LEAVE-BOOKED' ei='23'/>
 </Event>
 <TransactionEnd RowCount='1' DBID='44444' Sent='2007/06/27-15:20:58.782005' ↩
Committed='2007/06/27-15:20:58.434591' Version='1.0'/>
</TransactionPart>

EXAMPLE "Insert":

<TransactionPart Sub='SUB1' Origin='I' Prefix='N' Part='1' ↩
Id='C0D5D7140B13D3C900000000' En='4091'>
 <E Row='1' File='EMPLOYEES-FILE' ISN='1' Op='Insert' FNR='10'>
  <F ei='1' ai='50005800' key='uk'/>
  <F ei='2' ai='SIMONE'/>
  <F ei='3' ai='ADAM' key='ky'/>
  <F ei='5' ai='M'/>
  <F ei='6' ai='F'/>
  <F ei='8' ai='1952/01/30' key='ky'/>
  <F ei='9' ai='26 AVENUE RHIN ET DA' att='mu' ix='1'/>
  <F ei='10' ai='JOIGNY' key='ky'/>
  <F ei='11' ai='89300'/>
  <F ei='12' ai='F'/>
  <F ei='13' ai='1033'/>
  <F ei='14' ai='44864858'/>
  <F ei='15' ai='VENT59' key='ky'/>
  <F ei='16' ai='CHEF DE SERVICE' key='ky'/>
  <F ei='18' ai='FRA' att='pe' ix='1' gei='17'/>
  <F ei='19' ai='159980' att='pe' ix='1' gei='17'/>
  <F ei='20' ai='23000' ix='1' ix2='1' att='mu' gei='17'/>
  <F ei='21' ai='19'/>
  <F ei='22' ai='5'/>
  <F ei='24' ai='19990801' att='pe' ix='1' gei='23'/>
  <F ei='25' ai='19990831' att='pe' ix='1' gei='23'/>
  <F ei='26' ai='FRE' att='mu' ix='1'/>
  <F ei='26' ai='ENG' att='mu' ix='2'/>
 </E>
 <TransactionEnd RowCount='10' DBID='44444' Sent='2007/07/02-19:51:44.051797' ↩
Committed='2007/07/02-19:51:43.712573' Version='1.0'/>
</TransactionPart>

EXAMPLE "Update":

<TransactionPart Sub='SUB1' Origin='I' Prefix='N' Part='1' ↩
Id='C0D5D7140B13D3C900000000' En='4091'>
 <E Row='1' File='EMPLOYEES-FILE' ISN='10' Op='Update' FNR='10'>
  <F ei='1' bi='50004000' ai='11111111' key='uk'/>
  <F ei='20' bi='8880' ai='0000' ix='1' att='mu' ix2='1' gei='17'/>
  <F ei='18' bi='USD' ai='TWN' att='pe' ix='31' gei='17'/>
  <F ei='26' bi='ENG' ai='CHN' att='mu' ix='2'/>
 </E>
 <TransactionEnd RowCount='1' DBID='44444' Sent='2007/07/08-19:51:44.051797' ↩
Committed='2007/07/08-19:51:43.712573' Version='1.0'/>
</TransactionPart>
```

```
EXAMPLE "Delete":

<TransactionPart Sub='SUB1' Origin='I' Prefix='N' Part='1' ↩
Id='C0D5D7140B13D3C900000000' En='4091'>
 <E Row='1' File='EMPLOYEES-FILE' ISN='10' Op='Delete' FNR='10'>
 </E>
 <TransactionEnd RowCount='1' DBID='44444' Sent='2007/07/09-19:51:44.051797' ↩
Committed='2007/07/09-19:51:43.712573' Version='1.0'/>
</TransactionPart>
```

Index

M

O

R

S

U

W