Event Replicator for Adabas

Concepts

Version 3.5.3

March 2018

ADABAS & NATURAL

This document applies to Event Replicator for Adabas Version 3.5.3 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: ARF-INTRO-353-20180619

Table of Contents

1 Concepts and Facilities

Software AG's Event Replicator for Adabas allows specific Adabas files to be monitored for data modifications. Whenever any record modification (delete, store, or update) occurs in one of the monitored files, the Event Replicator for Adabas extracts each modified record and delivers it to one or more target applications through a messaging system (such as webMethods EntireX or IBM WebSphere MQ). The set of replicated files are defined in one or more subscriptions.

Note: The term *MQSeries* is sometimes used in this documentation when referring to the product now known as *WebSphere MQ*.

This chapter provides an introduction to the concepts and use of the Event Replicator for Adabas:

Why Use the Event Replicator for Adabas?	Describes why the Event Replicator for Adabas is useful.
Event Replicator for Adabas Architecture	Describes the architecture of Event Replicator for Adabas.
Replication Processing	Describes the phases of replication processing provides details on replication processing in the Adabas database and the Event Replicator Server. This chapter also describes specific types of replication processing, such as processing initial-state data or submitting requests for data to the Event Replicator Server.
Event Replicator for Adabas Operation and Administration	Describes the general operation and administrative tasks associated with the Event Replicator for Adabas.
Getting Started	Describes how to get started installing and using Event Replicator for Adabas and where to go to get additional information.
Useful Documentation Links	Lists useful documentation links.

2 Why Use the Event Replicator for Adabas?

The Event Replicator for Adabas is an essential tool for organizations that need Adabas data modifications delivered to a target application while minimally impacting the normal processing of Adabas. The principle features of the Event Replicator for Adabas include:

- Near real-time replication
- Asynchronous replication
- Guaranteed consistency and sequence of the delivered replicated data
- Replication of committed updates only

With the Event Replicator for Adabas, whole Adabas files or a specific set of records can be replicated to the target location, as defined in one or more subscriptions. Data replication is asynchronous, which allows the Adabas database to operate normally while replication takes place. Only committed Adabas modifications are replicated for the predefined set of replicated files, at the transaction level.

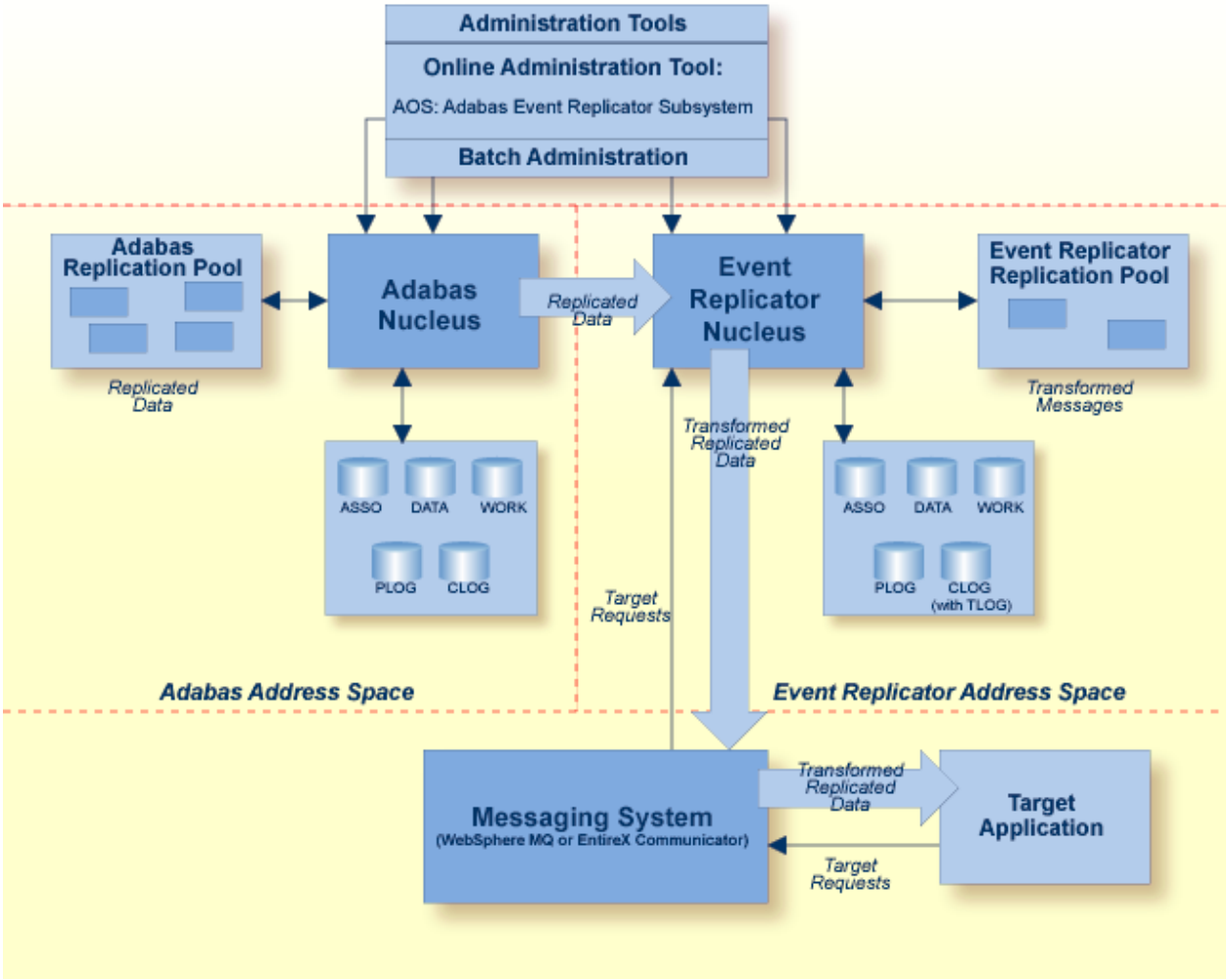
You can use Event Replicator for Adabas to:

- Provide real-time backups of your Adabas data. You can then guarantee data recoverability for your organization and the continuity of your business operations if a disaster should occur.
- Provide real-time information synchronization across multiple systems of data. For example, you might use the Event Replicator and its Event Replicator Target Adapter to provide real-time data from your mainframe Adabas database to a relational database, such as such as DB2, MySQL, Oracle, SQL Server, Sybase, or Teradata, where it can be used by other applications in your environment.
- Keep your data warehouses updated with current, real-time, data from your Adabas database, without impacting your production systems. This improves the productivity and effectivity of your business analysts by providing them with ready access to the most current data.
- Reduce your mainframe usage by replicating your real-time Adabas data to Adabas on open systems.

3 Event Replicator for Adabas Architecture

The Event Replicator for Adabas is not an application in the traditional sense. Rather, it is made up of several components that work together to replicate data to a target application without disrupting normal Adabas operations. A portion of the replication task occurs within the Adabas nucleus address space, while another portion of the task occurs within an address space known as the Event Replicator Server.

Pictured below is a diagram of the Event Replicator for Adabas architecture and the main components involved.



Administration Tools

The definitions and settings required for replication can be specified in any of the following ways:

- They can be specified as initialization parameters, which are read from the DDKARTE statements of the Event Replicator Server startup job. The parameters for this *batch administration* are described in *Event Replicator Initialization Parameters* in *Event Replicator for Adabas Reference Guide*.
- They can be specified in the Replicator system file which is read at Event Replicator Server startup. The definitions are maintained in the Replicator system file using the provided *online menu-driven interface*, Adabas Event Replicator Subsystem (a subsection of Adabas Online System). For more information, read *Using the Adabas Event Replicator Subsystem* in *Adabas Event Replicator Subsystem User's Guide*.

In addition, the Adabas Online System (AOS) can be used to maintain the definitions for the source Adabas database as well as the Event Replicator Server and it can be used to activate and deactivate replication resources. For more information on how it can be used to activate and deactivate replication resources, read *Managing Replication Definitions from AOS* in *Event Replicator for Adabas Administration and Operations Guide*.

Adabas Nucleus

The Adabas nucleus is a set of programs that control an Adabas database and coordinate all of its work.

For information on the setup that must be performed to replicate data from your Adabas nucleus, read ***Adabas Nucleus Replication Setup***, elsewhere in this guide. For information on the processing that occurs in an Adabas nucleus during replication, read ***Replication Processing***, paying specific attention to the section ***Adabas Nucleus Detail Processing***, elsewhere in this guide.

Adabas Replication Pool

The Adabas replication pool is a buffer area used to store the replication data processed by the Adabas nucleus. The Adabas replication pool is updated solely by the code within the Adabas nucleus address space. The size of this pool is set by the ADARUN `LRPL` parameter.

The Adabas replication pool may become full if:

- The ADARUN LRPL setting is too small to accommodate the replication data for parallel transactions.

- Adabas temporarily or persistently produces more replication data than the Event Replicator Server or the messaging system can process
- An outage of the Event Replicator Server or messaging system occurs.

When the Adabas replication pool becomes full, the replication system will deactivate replication for a file and a message is issued indicating which file was deactivated. When replication is inactive for a file, Adabas does not collect any further replicated updates for the file. Deactivated files may be explicitly reactivated via the ADADBS `REPLICATION` function once pool storage becomes available.

You can request warning messages when the replication pool usage exceeds a threshold that you specify using the ADARUN parameter `RPWARNPERCENT`. An initial message will be generated when this value, expressed as a percentage of replication pool usage, is exceeded. Additional messages will be generated when usage increases or decreases by the value you specify with ADARUN parameter `RPWARNINCREMENT`.

To avoid flooding the console with warnings, you can specify ADARUN parameters `RPWARNMESSAGELIMIT` and `RPWARNINTERVAL`. If more than `RPWARNMESSAGELIMIT` warnings are issued within the number of seconds specified by `RPWARNINTERVAL`, further warnings will be suppressed for the duration of the interval. When the interval expires, you will receive a message showing how many warning were suppressed, and warning messages can then resume.

Event Replicator Server Nucleus

The Event Replicator Server nucleus is a set of programs that control the Event Replicator Server and coordinate all of its work. This includes the components that parse and process all replicated data received from the Adabas nucleus and that receive and process inbound requests from the target application via the messaging system.

For information on the setup that must be performed to replicate data using a Event Replicator Server nucleus, read ***Event Replicator Server Nucleus Replication Setup***, elsewhere in this guide. For information on the processing that occurs in an Event Replicator Server nucleus during replication, read ***Replication Processing***, paying specific attention to the section ***Event Replicator Server Nucleus Processing***, elsewhere in this guide.

Event Replicator Server Replication Pool

The Event Replicator Server replication pool is a buffer area used by the replication code running in the Event Replicator Server address space. It is used to store decompressed and transformed replication data processed by the Event Replicator Server, along with other information. The pool is allocated in the address space of the Event Replicator Server. The size of this pool is set by the ADARUN parameter `LRPL`.

The Event Replicator replication pool may become full if:

- The ADARUN `LRPL` parameter setting is too small to accommodate the replication data being gathered from participating Adabas nuclei.
- The Event Replicator Server temporarily or persistently produces more replication data than the messaging system can process
- An outage of a participating Adabas nucleus or message system occurs.

When the Event Replicator replication pool becomes full, the replication system may deactivate its resources, including destinations, subscriptions, files, or databases. The replication system determines and deactivates the resource having the greatest impact on the Event Replicator replication pool usage.

- When a destination is deactivated, all subscriptions sent to that destination and to no other active destination will also be deactivated. A message is issued indicating which destination was deactivated.

  A destination may be explicitly reactivated once pool storage becomes available using the ADADBS REPTOR function or using the Event Replicator Server management screens in the Adabas Online System. For more information, read *Managing Replication Definitions from AOS* in *Event Replicator for Adabas Administration and Operations Guide*.
- When a subscription is deactivated, all files that occur in that subscription and in no other active subscriptions are also deactivated. A message is issued indicating which subscription is deactivated.

  Deactivation of a subscription does not cause deactivation of related destinations. Those destinations containing only subscriptions that have been deactivated simply receive no more data.

  A subscription may be explicitly reactivated once pool storage becomes available using the ADADBS REPTOR function or using the Event Replicator Server management screens in the Adabas Online System. For more information, read *Managing Replication Definitions from AOS* in *Event Replicator for Adabas Administration and Operations Guide*.
- When a file is deactivated, subscriptions containing the file remain active. Subscriptions that contain only files for which replication has been deactivated simply receive no more data. Sub-

scriptions that contain both deactivated files and active files will continue to deliver replication data to the target application, even though only partially replicated data will be delivered.

Once a file has been deactivated, the target application will be responsible for determining what to do with any partially replicated data. The application may decide to keep the partial data, and thus avoid the potentially costly refresh of large files for which replication has (or could have) remained active all the time. The target application has the choice to:

- Store the partially replicated data in the target database, but deactivate access to that part of the database until full replication has been reestablished.
- Store the partially replicated data in some temporary file and pick it up once full replication is being reestablished. Meanwhile, access to the replicated data can continue, as of the time when replication for the file was deactivated.
- Deactivate some or all subscriptions that contain the file for which replication has been deactivated.

Deactivation of a file in the Event Replicator Server causes deactivation of the file in the Adabas nucleus. A message is issued, indicating which file was deactivated.

A file may be explicitly reactivated once pool storage becomes available using the ADADBS REPTOR function or using the Event Replicator Server management screens in the Adabas Online System. For more information, read *Managing Replication Definitions from AOS*in the *Event Replicator for Adabas Administration and Operations Guide*.

- When a database is deactivated, all of its currently active replicated files are also deactivated. Messages are issued indicating which files were deactivated. See the information above on file deactivation for more information on actions that can be taken when a file is deactivated.

  A database may be explicitly reactivated once pool storage becomes available using the ADADBS REPTOR function or using the Event Replicator Server management screens in the Adabas Online System. For more information, read *Managing Replication Definitions from AOS*in the *Event Replicator for Adabas Administration and Operations Guide*.

You can request warning messages when the replication pool usage exceeds a threshold that you specify using the ADARUN parameter `RPWARNPERCENT`. An initial message will be generated when this value, expressed as a percentage of replication pool usage, is exceeded. Additional messages will be generated when usage increases or decreases by the value you specify with ADARUN parameter `RPWARNINCREMENT`.

You can reduce the risk of Event Replicator Server replication pool overflows by requesting that the Event Replicator Server temporarily write replication transactions received from the Adabas nucleus to the SLOG system file. The Event Replicator Server then processes them using a throttling mechanism so that only a limited amount of replication pool space is used at a time. This functionality is controlled by the setting of the optional `LOGINPUTTRANSACTION` initialization parameter for the Event Replicator Server. For more information, read *Reducing the Risk of Replication Pool Overflows*, in the *Event Replicator for Adabas Administration and Operations Guide* and *LOGINPUTTRANSACTION (Log Input Transaction) Parameter* in the *Event Replicator for Adabas Reference Guide*.

To avoid flooding the console with warnings, you can specify ADARUN parameters RPWARNMESSAGELIMIT and RPWARNINTERVAL. If more than RPWARNMESSAGELIMIT warnings are issued within the number of seconds specified by RPWARNINTERVAL, further warnings will be suppressed for the duration of the interval. When the interval expires, you will receive a message showing how many warning were suppressed, and warning messages can then resume.

The Messaging Systems

These intermediary systems process messages from the Event Replicator Server and send them to the target application. They also transmit requests from the target application to the Event Replicator Server.

The Event Replicator Server sends replicated data to a messaging system (for example, webMethods EntireX or WebSphere MQ) for delivery to the target application.

The target application may send initial-state requests, status requests, and prior-transaction (resend buffer) requests to the messaging system for delivery to the Event Replicator Server.

Note: The target application can be another Event Replicator Server accessed via an EntireX or WebSphere MQ destination and over an EntireX or WebSphere MQ queue. This arrangement might be useful if an Event Replicator Server cannot access a specific Adabas database or if the connection between them is very slow.

For information on integrating the message system with your target application, read *Integrating the Messaging System with the Target Application* in *Event Replicator for Adabas Administration and Operations Guide*.

Target Application

The target application is the outside application to which replicated data is sent.

Note: The target application can be another Event Replicator Server accessed via an EntireX or WebSphere MQ destination and over an EntireX or WebSphere MQ queue. This arrangement might be useful if an Event Replicator Server cannot access a specific Adabas database or if the connection between them is very slow.

Target Requests

Target requests are requests from the target application to the Event Replicator Server. Such requests include status requests, initial-state requests, and prior-transaction (resend buffer) requests.

4 Replication Processing

This chapter describes various aspects of Event Replicator for Adabas processing.

Normal Processing Phases

During normal processing, the following phases of processing occur:

Phase	Processing Description
Collection	In the Adabas nucleus, when an update occurs to a replicated file in an Adabas database, the compressed information related to the update is collected and stored in the Adabas nucleus replication pool. Once the update has been fully processed and committed by Adabas (the protection information is stored properly in the Work data set and the protection log), all of the updates in a transaction are queued to be sent to the Event Replicator Server.
Transfer	The compressed replication transaction in the Adabas replication pool is sent to the Event Replicator Server.
Input	The replication transaction is received by the Event Replicator Server and stored in the Event Replicator Server replication pool. At this point, the replication data exists in both the Adabas and Event Replicator Server replication pools. When the entire transaction is received, it is queued for the Assignment phase.
Assignment	The replication transaction is queued for processing by zero or more subscriptions.
Subscription	The replication transaction is processed by the subscriptions to which it has been assigned. During this phase, the compressed replication transaction is adjusted using the specifications in the subscription to create a decompressed replication transaction. It is also assigned to zero or more destinations by the subscription.
Output	The decompressed replication transaction is processed by each destination to which it was assigned. Each destination is processed in one of the following ways: ■ It sends the transaction to a messaging system, such as webMethods EntireX Broker or WebSphere MQ, which then transfers it to a target application for processing. ■ It writes the transaction data to a target Adabas database. ■ If the `DLOG` (Allow Logging) parameter is set to "YES" for a destination and if the destination is closed, the replication transaction is saved on the SLOG system file. For information about the SLOG system file, all of its uses, and suggestions for determining its size, read *Using the Subscription Logging Facility* and *Reducing the Risk of Replication Pool Overflows*, in the *Event Replicator for Adabas Administration and Operations Guide*. Once the replication transaction is fully processed by all its destinations, the transaction is queued for completion.
Completion	On the Event Replicator Server, the Adabas nucleus from which the replication transaction originated is notified that replication for the transaction is completed and the transaction is removed from the Event Replicator Server replication pool. When the Adabas nucleus receives the completion notification from the Event Replicator Server, all data for the complete transaction is removed from the Adabas nucleus replication pool.

Definitions Driving Replication

Once the Event Replicator for Adabas in installed, its replication processing is driven by definitions you specify. These definitions are described in the following table in order of importance to replication (required definitions are listed first).

Note: You can run Event Replicator for Adabas in verify (test) mode, by turning on verification in the VERIFYMODE replication definition. This is useful if you want to test the definitions you have specified before you start using Event Replicator for Adabas in production mode. For more information, read *Running in Verify Mode*, in *Event Replicator for Adabas Installation Guide*.

Definition Type	Defines	How many definitions are required?
destination	The destination of the replicated data. Destination definitions can be created for Adabas, File, webMethods EntireX, WebSphere MQ, and Null destinations. To maintain destination definitions using DDKARTE statements of the Event Replicator Server startup job, read *Destination Parameter* in *Event Replicator for Adabas Reference Guide*. To maintain destination definitions using the Adabas Event Replicator Subsystem, read *Maintaining Destination Definitions* in *Adabas Event Replicator Subsystem User's Guide*.	Required. At least one destination definition is required for data replication to occur. Create one definition for every Event Replicator for Adabas destination you intend to use.
subscription	A set of specifications to be applied to the replication of the data. These include (but are not limited to): ■ the identification of the Adabas files that should be replicated and how they should be replicated (SFILE definitions that should be processed as part of the subscription) ■ architecture key, output alpha and wide-character keys that should be used ■ the name of the resend buffer definition that should be used for replication, if any ■ various settings relating to the availability of the subscription in specific circumstances Subscription definitions identify SFILE definitions and resend buffer definitions that should be used. At least one SFILE definition is required. To maintain subscription definitions using DDKARTE statements of the Event Replicator Server startup job, read *SUBSCRIPTION Parameter* in *Event Replicator for Adabas Reference Guide*. To maintain subscription definitions using the Adabas	Required. At least one subscription definition is required for data replication to occur.

Definition Type	Defines	How many definitions are required?
	Event Replicator Subsystem, read *Maintaining Subscription Definitions* in *Adabas Event Replicator Subsystem User's Guide*.	
SFILE	An Adabas file to be replicated and the replication processing that should occur for that file. SFILE definitions are sometimes referred to as *subscription file definitions* and are referenced by subscription definitions. An SFILE definition identifies (among other things): ■ the Adabas database ID and file number that should be replicated ■ the transaction filter definitions that should be used to filter the data in the Adabas file during replication (if any) ■ the subscription user exit that should be processed during replication (if any) ■ whether insert, delete, and update transactions should be replicated ■ the file's alpha character encoding, if any ■ the GFB definitions that should be used for replication, if any, or the specific format buffer definitions that should be used instead. To maintain SFILE definitions using DDKARTE statements of the Event Replicator Server startup job, read *SUBSCRIPTION Parameter* in *Event Replicator for Adabas Reference Guide*. To maintain SFILE definitions using the Adabas Event Replicator Subsystem, read *Maintaining SFILE Definitions* in *Adabas Event Replicator Subsystem User's Guide*.	Required. At least one SFILE definition is required for data replication to occur.
initial-state	An initial-state request for data from the target application. Initial-state definitions identify the subscription, destination, and specific Adabas files to use in an Event Replicator for Adabas initial-state run. To maintain initial-state definitions using DDKARTE statements of the Event Replicator Server startup job, read *INITIALSTATE Parameter* in *Event Replicator for Adabas Reference Guide*. To maintain initial-state definitions using the Adabas Event Replicator Subsystem, read *Maintaining Initial-State Definitions* in *Adabas Event Replicator Subsystem User's Guide*.	Not required. If you want initial-state data produced in an Event Replicator for Adabas run, only one initial-state definition is required. Otherwise, no initial-state data definition is required.
IQUEUE	The input queue on which Event Replicator for Adabas should listen for requests from webMethods EntireX and WebSphere MQ targets. To maintain IQUEUE definitions using DDKARTE statements of the Event Replicator Server startup job, read *IQUEUE*	Not required. At least one IQUEUE definition is required for every EntireX Communicator or WebSphere MQ target you

Definition Type	Defines	How many definitions are required?
	Parameter in *Event Replicator for Adabas Reference Guide*. To maintain IQUEUE definitions using the Adabas Event Replicator Subsystem, read *Maintaining Input Queue (IQUEUE) Definitions* in *Adabas Event Replicator Subsystem User's Guide*.	intend to use. If webMethods EntireX or WebSphere MQ are not used, no IQUEUE definition is required.
GFB	A global format buffer (GFB) definition stored separately for use in SFILE definitions. You can specify GFBs manually or generate them using Predict file definitions. When you generate them, a field table is also generated. While a format buffer specification is required in a subscription's SFILE definition, a stored GFB definition does not need to be used. The SFILE definition could simply include the format buffer specifications it needs. To maintain GFB definitions using DDKARTE statements of the Event Replicator Server startup job, read *GFORMAT Parameter* in *Event Replicator for Adabas Reference Guide*. To maintain GFB definitions using the Adabas Event Replicator Subsystem, read *Maintaining GFB Definitions* in *Adabas Event Replicator Subsystem User's Guide*.	Not required. No GFB definition is required. If a global format buffer is needed, at least one GFB definition is required.
resend buffer	A resend buffer that can be used by any subscription to expedite the retransmission of a transaction. To maintain resend buffer definitions using DDKARTE statements of the Event Replicator Server startup job, read *RESENDBUFFER Parameter* in *Event Replicator for Adabas Reference Guide*. To maintain resend buffer definitions using the Adabas Event Replicator Subsystem, read *Maintaining Resend Buffer Definitions* in *Adabas Event Replicator Subsystem User's Guide*.	Not required. No resend buffer definition is required. If you elect to retransmit a transaction, at least one resend buffer definition is required.
transaction filter	A filter definition that can be used to filter the records used for replication based on the values of fields in those records. To maintain transaction filter definitions using DDKARTE statements of the Event Replicator Server startup job, read *FILTER Parameter* in *Event Replicator for Adabas Reference Guide*. To maintain transaction filter definitions using the Adabas Event Replicator Subsystem, read *Maintaining Transaction Filter Definitions* in *Adabas Event Replicator Subsystem User's Guide*.	Not required. No transaction filter definition is required. If you want to use a transaction filter to filter records used in replication, at least one transaction filter definition is required.

Detailed Adabas Nucleus Processing

During session start, the Adabas nucleus performs the following tasks:

- Allocates memory needed by the replication-related code.
- Determines which files need to be replicated.
- Establishes contact with each Event Replicator target ID associated with files to be replicated.

Once the Adabas nucleus is started, the following replication processing occurs during the **collection phase**:

1. For each user, the Adabas nucleus tracks and places information in the **Adabas replication pool** related to modifications to each record in each file selected for replication. To track modifications, the Event Replicator for Adabas captures the before and after images (compressed) of all modified data.
2. The nucleus accumulates replication data for an entire user transaction, as follows:
   - A primary key may or may not be associated with a replicated file. If a primary key is not associated with a replicated file, any before image associated with a record is the before image from data storage. If a primary key is associated with a replicated file and the key is modified, any before image associated with a record is only the compressed value of the primary key. The primary key can be specified using the ADALOD LOAD RPLKEY parameter.
   - Data for replication is collected at the time protection records are created for modifications to records within a replicated file. The following data is collected during a transaction that modifies records on one or more replicated files:
     - Data storage after image.
     - Data storage before image during a delete.
     - Data storage before image during an update (optionally, if this feature is activated for the file).

       For further information, see the ADALOD LOAD RPLDSBI parameter documentation and the ADADBS REPLICATION DSBI parameter documentation. Discussions of Adabas utility functions specific to Event Replicator for Adabas can be found in *Utilities Used with Replication* in *Event Replicator for Adabas Reference Guide*.
     - Primary key before image during an update or delete, when the optional primary key has been defined for the file and the key has been updated or deleted from the record.

     If a transaction is backed out, the nucleus discards all replication data collected for the transaction. No replication-related information is collected during an update command when no data is modified during the update command.

3. If any record is modified more than once during a transaction, the nucleus makes available to the outside destination only the final instance of the modification of the record. This is done by consolidating modifications to the same record within the same user transaction, as follows:
   - For the sake of performance, no data consolidation occurs at the point the modification related data is collected.
   - Data consolidation occurs in the nucleus address space.
   - Data consolidation occurs after a transaction is committed on WORK.
   - The following rules apply to the consolidation of modifications to the same record during one transaction:
     - An insert followed by an update is treated as an insert.
     - An update followed by another update is treated as one update.
     - An update followed by a delete is treated as one delete.
     - When a before image exists for a primary key and a before image exists from data storage, the before image for the primary key is used.
     - The first before image captured is used.

       Note: This rule is subject to the above rule regarding a before image of a primary key versus the before image from data storage.

     - The last after image captured is used.

   There is an exception to these data consolidation rules: a delete followed by an insert to the same record will be treated as two separate modifications.

   Important: The setting of the RPLSORT parameter can affect how modifications are consolidated. When RPLSORT=YES is specified, modifications are consolidated as described in this section. When RPLSORT=NO is specified, modifications are still consolidated, but their referential integrity is maintained. In other words, the chronological order of the updates is maintained for each ISN in a file. For more information, read *RPLSORT Parameter*, in *Event Replicator for Adabas Reference Guide*.

4. The nucleus notifies the Event Replicator Server when a transaction with replicated data has been committed.
5. The nucleus makes the replication data available to the Event Replicator Server.

When the Adabas nucleus receives the completion notification from the Event Replicator Server during the **completion phase**, all data for the complete transaction is removed from the Adabas nucleus replication pool.

Notes:

1. The presence of the Event Replicator Server has minimal impact on any Adabas session in terms of the performance (CPU consumption, throughput, response time) of an Adabas database, in comparison to it running without replication active.
2. In a clustered environment, the Event Replicator for Adabas supports replicated data coming from multiple Adabas nuclei that are active for the same database.

For information on the Adabas nucleus settings, see *Adabas Initialization (ADARUN) Parameters* and *Utilities Used with Replication* in *Event Replicator for Adabas Reference Guide*.

Detailed Event Replicator Server Nucleus Processing

The following description summarizes the processing performed by the Event Replicator Server:

1. During Event Replicator Server startup, the Event Replicator Server establishes contact with Adabas nuclei for the related database IDs. If an Adabas nucleus is *not* yet active, the nucleus contacts the Event Replicator Server during nucleus initialization.
2. During the **input, assignment, and subscription phases**, the Event Replicator Server processes the received modified data according to the subscriptions defined in the replication definition parameters.

   After processing the data, the Event Replicator Server may apply user-customizable logic to the replication process (for example, filtering, conversion, or transformation).
3. During the **output phase**, the Event Replicator Server delivers the replicated data to the messaging system destination for replication to the target application.

   Each replicated transaction delivered to a target is assigned a unique sequence number. This sequence number is generated for each unique subscription-destination combination of the replicated transaction. In other words, if the same replicated transaction is delivered to two different destinations by a subscription, that transaction may have two different sequence numbers (one for each destination).

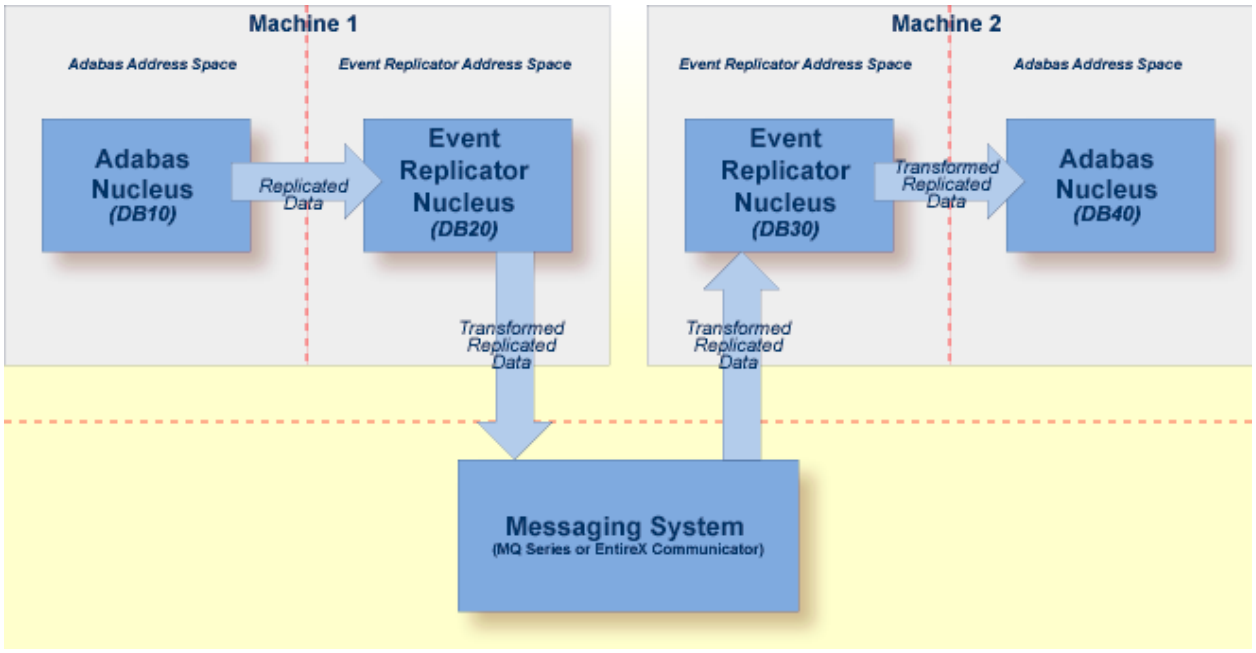
   Initial-state requests may be needed to resolve an ambiguous state incurred by the target application; the request can contain requests for a single record, a set of records, or an entire file.
4. During the **completion phase**, the Event Replicator Server notifies the appropriate Adabas nucleus that transaction replication is completed and then removes all information about the completed transaction from the Event Replicator Server replication pool.

An Event Replicator Server may process data from multiple databases. Replication data for one Adabas file must be processed by a single Event Replicator Server. No two Event Replicator Servers handle the same set of files from the same database.

Node-to-Node Support

Event Replicator for Adabas provides node-to-node support, in which one Event Replicator Server can send replicated data to a second Event Replicator Server. The second Event Replicator Server can then, in turn, send the replicated data onto other destinations.

Consider the example depicted in the following picture:



The following processing occurs in this example:

1. Source Adabas database 10 sends its data to Event Replicator Server 20.
2. Event Replicator Server 20 sends the replicated data to an webMethods EntireX or WebSphere MQ destination.
3. Event Replicator Server 30 reads the replicated data from the webMethods EntireX or WebSphere MQ input queue.
4. Event Replicator Server 30 processes the replicated and sends it to Adabas database 40.

The following definitions must exist for this processing to occur correctly:

- The definitions for Event Replicator Server 20 must include one or more subscriptions for Adabas database 10, which send the replicated data to a webMethods EntireX Broker or WebSphere MQ destination. The subscriptions may optionally also transform the replicated data.
- The definitions for Event Replicator Server 30 must include webMethods EntireX or WebSphere MQ input queue definitions that match the webMethods EntireX or WebSphere MQ destination

definitions in Event Replicator Server 20. The IQBUFLEN parameter value (input queue buffer length), specified in the input queue definitions for Event Replicator Server 30, must be greater than or equal to the length of the largest message sent by Event Replicator Server 20. The largest message sent by Event Replicator Server 20 is limited by the minimum of the following Event Replicator Server 20 settings: MAXOUTPUTSIZE parameter, DMAXOUTPUTSIZE parameter (if specified for the destination), or the message limit imposed by the messaging system.

Parameters IRMSGINTERVAL and IRMSGLIMIT control the number of input queue related messages printed. Consider the values set for these two parameters in Event Replicator Server 30.

The Event Replicator Server 30 definitions must also include at least one subscription for any data received from Adabas database 10 (in this case via the input queue). This subscription must send the replicated data to an Adabas destination definition for Adabas database 40.

The Event Replicator Server 30 definitions must also specify that, at startup, Event Replicator Server 30 will *not* automatically initiate a connection with Adabas database 10. In other words, the DBCONNECT parameter for database 10 must be set to "NO" in Event Replicator Server 30.

Replicating an Initial Version of Database Data

Normally, only changed data is replicated. However, this presumes that the target system has the same data that the source had before the change. If this is not the case, you need to get an initial version of the data to the target. This is accomplished using an *initial-state request*.

Initial-state requests must be supported by *initial-state definitions*. Each request must specify the name of the initial-state definition that should be used, as well as the database ID and file number to be processed. Initial-state definitions are specified in the Adabas Event Replicator Subsystem or by INITIALSTATE parameters in the Event Replicator Server startup job.

Initial-state requests are initiated either from the target application (client) to the Event Replicator Server or using the Adabas Event Replicator Subsystem. For information on how to submit an initial-state request from the target application to the Event Replicator Server, read *Event Replicator Client Requests* in *Event Replicator for Adabas Application Programmer's Reference Guide*. For information on how to submit an initial-state request from the Adabas Event Replicator Subsystem, read *Populating a Database With Initial-State Data* in *Adabas Event Replicator Subsystem User's Guide*.

Initial-state data can contain any subset of the data on the Adabas database, based on the specifications in the initial-state definition and parameters supplied in the initial-state request. Records can be selected for initial-state processing in one of the following manners:

- The complete file can be selected.
- Records are selected from the file based on an ISN list.
- Records are selected from the file based on specified selection criteria.

Note: Each replicated initial-state record contains the related data storage after image. No before image is replicated for an initial-state record.

During initial-state processing, the nucleus reads the selected records and passes them to the Event Replicator Server. The Event Replicator decompresses the records depending on the subscription format and sends the data to the assigned output destinations.

For more information, read about maintaining initial-state definitions using Adabas Event Replicator Subsystem. For information about the DDKARTE statements required for initial-state definitions, read about the INITIALSTATE parameter in *Event Replicator for Adabas Reference Guide*.

Submitting Requests for Data to the Event Replicator Server

Clients can send specific requests for data to the Event Replicator Server by sending messages to an Event Replicator input queue. The following requests can be made:

- Status inquiries
- Initial-state data requests
- Prior-transaction (resend buffer) data requests.

For more information about submitting these requests to the Event Replicator Server from the target application, read *Event Replicator Client Requests* in *Event Replicator for Adabas Application Programmer's Reference Guide*.

Messaging Event Replicator Server Destinations

The Adabas C5 command can be used to message Event Replicator Server destinations from your application. C5 commands are transmitted via the **messaging system**. For more information, read *C5 Command: Write User Data to Protection Log* in *Event Replicator for Adabas Reference Guide*.

Replicating Adabas Utility Functions

You can request that some utility functions performed against an Adabas database be replicated to another Adabas database. For example, if you change the field length of a field in an Adabas file, that change can be replicated to the target Adabas database. This eliminates the need for you to manually intervene to make the change in the target and eliminates the resulting errors if you do not.

Caution: In order for this utility replication to work properly, you must ensure that your source and target files are maintained in identical manners. If a utility function is performed against the source file and replicated to a target file that cannot accommodate the utility request, errors will result and replication to the target will fail.

This section covers the following topics:

Limitations

The following limitations exist in utility replication:

1. Utility functions cannot be replicated from an Adabas database to a target relational database (RDBMS) via the Event Replicator Target Adapter. If you need to refresh or drop tables in your RDBMS based on utility function activity performed against your Adabas database, you must use specific Adabas Event Replicator Subsystem functions once the Adabas utility has completed processing. For more information, read *Submitting Event Replicator Target Adapter Requests* (Adabas Event Replicator Subsystem) in the *Adabas Event Replicator Subsystem User's Guide*.
2. Not all utility functions can currently be replicated. In addition, some functions can only be replicated if they are initiated from Adabas Online System (AOS). The functions that are currently replicated include:
   - Adding new fields. The ADADBS NEWFIELD batch utility function can be replicated as well as the equivalent AOS and AMA functions.
   - Changing a field length. The ADADBS CHANGE batch utility function can be replicated as well as the equivalent AOS and AMA functions.
   - Deleting a file. The ADADBS DELETE batch utility function can be replicated as well as the equivalent AOS and AMA functions.
   - Refreshing (emptying) a file. The ADADBS REFRESH batch utility function can be replicated as well as the equivalent AOS and AMA functions.
   - Renaming a file. The ADADBS RENAME batch utility function can be replicated as well as the equivalent AOS and AMA functions.
   - Reusing ISNs. The ADADBS ISNREUSE batch utility function can be replicated as well as the equivalent AOS and AMA functions.
   - Reusing data storage blocks. The ADADBS DSREUSE batch utility function can be replicated as well as the equivalent AOS and AMA functions.
   - Modifying FCB parameters. The ADADBS MODFCB batch utility function can be replicated as well as the equivalent AOS and AMA functions.

- Releasing a descriptor. The ADADBS RELEASE batch utility function can be replicated as well as the equivalent AOS and AMA functions.
- Defining a file. The Adabas Online System functions can be replicated.
- Defining a new FDT. The Adabas Online System functions can be replicated.

Note: Event Replicator for Adabas supports the replication of data associated with an ADALOD LOAD or ADALOD UPDATE functions. For more information on this support, read *ADALOD LOAD Parameters* and *ADALOD UPDATE Parameters*, in *Event Replicator for Adabas Reference Guide*.

For complete information on these Adabas utility functions, refer to your online Adabas utilities or Adabas Online System documentation.

3. Password protection on the target file or database is not supported
4. Expanded files are not supported. Any segmentation of a file is hidden to the target. So, file deletion functions operate on the logical file.
5. The functions defining a new FDT, adding new fields, or defining a file, are replicated to all Event Replicator Servers known to the database nucleus. This is because the Event Replicator Server ID is not yet defined for the file (ADADBS REPLICATION function), so all Event Replicator Servers are informed. The functions are only replicated if the Event Replicator Server includes the file in a subscription with appropriate destination settings.

Required Parameters

Replicating Adabas utility functions is controlled by the destination definitions associated with your subscriptions. Using the DREPLICATEUTI parameter in a destination definition, you can control whether utility functions are replicated for that destination. In addition, for Adabas destinations, you can use the DAREPLICATEUTI parameter to control whether utility functions are replicated to a specific target database and file.

For more information about the DREPLICATEUTI and DAREPLICATEUTI parameters in destination definitions, read about the *Destination Parameters*, in *Event Replicator for Adabas Reference Guide*. For information on using the Adabas Event Replicator Subsystem to maintain destination definitions, read *Maintaining Destination Definitions Using the Adabas Event Replicator Subsystem* in *Adabas Event Replicator Subsystem User's Guide*.

Example

Suppose you have an Adabas database with a database ID of 241. In addition, suppose Event Replicator Server 65535 destination and subscription definitions look like this:

```
ADARPD DATABASE ID=240,DBCONNECT=NO
*
ADARPD DESTINATION NAME=ADA240
ADARPD  DREPLICATEUTI=YES
ADARPD  DTYPE=ADABAS
ADARPD  DAIFILE=151,DAIDBID=241,DATDBID=240,DATFILE=51
ADARPD   DAREPLICATEUTI=YES
*
ADARPD SUBSCRIPTION NAME=TICKER
ADARPD  SDESTINATION='ADA240'
ADARPD  SFILE=151,SFDBID=241,SFBAI='AA-ZZ.'
*
```

The TICKER subscription on Event Replicator Server 65535 is set up to replicate data from file 151 on database 241 using destination ADA240. However, suppose neither database file 151 or file 51 are loaded in their respective databases. The following processing can occur:

1. Define the new FDT and file 151 using Adabas Online System on database 241.
2. Since database 241 runs with REPLICATION=YES, the define operations for file 151 are sent to Event Replicator Server 65535.
3. The TICKER subscription sends changes from file 151 to destination ADA240 (database 240). Because destination ADA240 has DREPLICATEUTI=YES, it replicates the utility functions it receives in general. In addition, because DAREPLICATEUTI=YES for the target database 240, file 51, the define utility function is replicated to that specific target as file 51. In other words, new file 51 is defined using the definition specifications for file 151.

   Caution: If a file 51 already exists on database 240, this definition will fail and so will replication to this target. Take care when specifying target specifications for such operations.

4. You can now activate replication for file 151 (ADADBS REPLICATION FILE=151,ON,TARGET=65535). This will allow user transactions and utility functions from file 151 on database 241 to be replicated to Event Replicator Server 65535, which will, in turn, replicate them to file 51 on database 240.

Cross-Checking Subscription Definitions and Actual Replication

The Event Replicator Server includes functionality to check for inconsistencies between files specified in one or more subscriptions in the Event Replicator Server versus files replicated in Adabas. These inconsistencies may be caused by one of the following instances:

- A file may have replication turned on but not be referenced in a subscription in the related Event Replicator Server
- A file may be specified in a subscription in an Event Replicator Server and either not have replication turned on in Adabas or have replication turned on in Adabas with a different Event Replicator Server ID.

The cross-check function also displays a message listing files contained in an Adabas nucleus that are defined to the Event Replicator but which are currently inactive; replication for these files will not occur.

The Event Replicator Server replication cross-check function executes when an Adabas database first connects with the Event Replicator Server. It can also be invoked using the `RPLCHECK` operator command.

Ensuring Replication Data Availability

The *subscription logging facility* , also know as the *SLOG facility*, can be used to ensure that data replicated to specific destinations is not lost if problems occur on your destinations. In order for this to occur, the SLOG facility must be activated for those destinations. For complete information on using the SLOG facility, read *Using the Subscription Logging Facility* in *Event Replicator for Adabas Administration and Operations Guide*.

Notes:

1. Subscription logging is resource intensive and should only be used where absolutely necessary. As an alternative, you should consider using initial state processing to resynchronize replicated data in the event of a queue failure.
2. If a destination is unavailable for a significant amount of time, a large volume of data can be generated and written to the SLOG system file.
3. The single point of failure for the SLOG facility is that if the SLOG system file is not large enough to contain the data that must be logged, the SLOG facility fails and data will subsequently be lost.

Reducing the Risk of Event Replicator Server Replication Pool Overflows

To reduce the risk of an Event Replicator Server replication pool becoming full when a destination cannot handle the rate at which replication transactions are sent to it by the Event Replicator, you can now request that incoming compressed replication transactions be written to the SLOG system file, before they are queued to the **assignment phase**. This means that during the **input phase**, the compressed transactions are stored first in the Event Replicator replication pool, but then written to the SLOG system file, freeing up the space in the Adabas nucleus and Event Replicator Server replication pools.

For more information on using the SLOG system file in this manner, read *Reducing the Risk of Replication Pool Overflows*, in the *Event Replicator for Adabas Administration and Operations Guide*.

Transaction Logging

Event Replicator for Adabas transaction logging (TLOG) allows you to log transaction data and events occurring within the Event Replicator address space. This information can be used as an audit trail of data that has been processed by the Event Replicator Server and of state change events that occurred during Event Replicator Server operations. In addition, it can be used to assist in the diagnosis of problems when replication does not work as expected.

For complete information, read *Using Transaction Logging* in *Event Replicator for Adabas Administration and Operations Guide*.

Recovery

If Adabas terminates abnormally and restarts, it and the Event Replicator Server are usually able to recover any lost replication data and to deliver the normal stream of replication data to the target application.

This section covers the following topics:

- General Recovery Processing
- Data Loss Considerations
- Repeated Abends

- Cluster Database Considerations

General Recovery Processing

During normal processing, Adabas writes control information to its Work data set to keep track of which replicated transactions the Event Replicator Server has confirmed as successfully processed. If Adabas terminates abnormally and then performs the autorestart at the beginning of the next session, it uses the protection data and control information in the Work data set to rebuild the replication data that existed in its replication pool at the time of the failure.

After reconnecting to the Event Replicator Server, Adabas:

1. deletes rebuilt replication data that the Event Replicator Server successfully processed,
2. keeps rebuilt replication data that the Event Replicator Server fully received but has not yet successfully processed, and
3. resends rebuilt replication data that the Event Replicator Server has not yet fully received.

Adabas marks all replication data that it recovers from the Work data set as *possible resends* and sends it to the Event Replicator Server (because the Event Replicator Server did not indicate it already received the data). The target application may receive such replication data twice (the second time marked as *possible resend*) if the Event Replicator Server did not stay active throughout the Adabas outage, because, in this case, the next instance of the Event Replicator Server does not know which replication data the previous instance had already successfully processed.

Data Loss Considerations

If the Event Replicator Server has not been able to process the replication data sent by Adabas in a timely manner, it is possible that Adabas will overwrite replication-related protection data on the Work data set that has not yet been successfully processed by the Event Replicator Server. If, in such a situation, Adabas abends, it will not be able to rebuild all replication data that existed in its replication pool at the time of the failure. In this case, Adabas prints messages detailing which replicated files may be involved in the loss of replication data.

No replication data is lost in an Adabas failure if, prior to the failure, the Event Replicator Server processed the replication data in a timely manner so that no replication-related protection data that has not been successfully processed is overwritten on the Work data set. The amount of protection data that can be held on the Work data set before it is overwritten is determined by the ADARUN LP parameter setting of Adabas. Increasing the LP parameter setting provides for a greater safety margin against the overwrite of protection data related to unprocessed replication data.

No replication data is lost in an Adabas failure if the Event Replicator Server stayed active throughout the Adabas outage even though replication-related protection data that has not been successfully processed may have been overwritten on the Work data set. This is because the replication data that Adabas is unable to rebuild from the Work data set is still present in the Event

Replicator replication pool. After reconnecting to the Event Replicator Server, Adabas prints messages indicating that although the replication-related protection data that has not been successfully processed was overwritten on the Work data set, no replication data was actually lost.

Furthermore, in the case of the possible loss of replication data, the Event Replicator Server issues a status message to the target application indicating this condition for every subscription-destination related to any affected file.

Repeated Abends

If, after the session autorestart, Adabas abends *again* before the Event Replicator Server has received and processed all of the rebuilt replication data, Adabas will in the following second session autorestart again rebuild the relevant replication data from the information on its Work data set and, if necessary, resend it to the Event Replicator Server. This is possible as long as the protection data related to the as yet unprocessed replication data has not been overwritten on the Work data set.

If the session autorestart after an Adabas abend consistently fails for a replication-related reason, it is possible to restart Adabas with REPLICATION=NO. This makes Adabas perform the session autorestart without any attempt to recover replication data. It is an emergency measure to get Adabas back up, but disables replication processing. The replication-related parameters of the files that used to be replicated must be defined again and the original files and their replicas must be brought back in sync.

Cluster Database Considerations

In a cluster database, the cluster nucleus performing the recovery process makes the Work data sets of the other nuclei (logically) empty at the end of the recovery. Replication data originating from the other nuclei can no longer be rebuilt after successful recovery. To avoid the loss of this replication data when another failure occurs before the recovered replication data has been sent to and processed by the Event Replicator Server, the nucleus performing the recovery process writes all replication data originating from the other nuclei to its own Work data set. Then, even if the first nucleus fails before sending all recovered replication data to the Event Replicator Server, these replication data will be recovered again from where it was written in the first recovery process.

A cluster nucleus writing replication data to the Work data set during recovery incurs a greater risk of a Work data set overflow. If a Work data set overflow occurs in a cluster database and an existing peer nucleus also gets a Work data set overflow during the recovery process, use the following procedure to recover without the need to restore and regenerate the database:

1. Define (or use) a new cluster nucleus that has not been active at the time of the Work data set overflow. Define a large LP parameter for this new nucleus. An LP value equal to the sum of the LP values of all cluster nuclei that were active at the time of the first failure should be sufficient.

2. Start the new cluster nucleus with the large LP parameter to let it perform the recovery process. It will read the protection and replication data from the Work data sets of the failed peer nuclei and write new protection and replication data to its own Work data set. The latter one can be made as large as necessary to resolve the Work overflow condition.

This procedure of resolving a Work data set overflow applies to all cluster nuclei, with or without replication, but is especially important for cluster nuclei with replication, as they have a greater risk of Work overflow during recovery.

Interactions With Adabas Transaction Manager

Event Replicator for Adabas can be used with DTP=RM databases that have transactions coordinated by Adabas Transaction Manager. Replication takes place near real-time in the same way as it does for DTP=NO databases.

5 Event Replicator for Adabas Operation and Administration

The operation and administration features of Adabas allow you to activate replication, define files to be replicated, and configure other Adabas features related to the Event Replicator for Adabas. Event Replicator for Adabas administration also involves the configuration of both the Adabas nucleus and the Event Replicator Server components.

You can perform Event Replicator for Adabas administration tasks using standard Adabas utilities and database initialization parameters or the Adabas Online System (AOS) and its Adabas Event Replicator Subsystem. This chapter provides more information on Adabas and Event Replicator Server database administration tasks related to replication.

Adabas Nucleus Replication Setup

The following replication setup should be performed for an Adabas database you want replicated.

- If you are defining a database, be sure to specify the DEFINE REPTOR=NO parameter of the ADADEF utility to indicate that the database is not an Event Replicator Server. For more information, read *ADADEF DEFINE REPTOR Function* in *Event Replicator for Adabas Reference Guide*.

  You can modify this setting later using the MODIFY REPTOR parameter of the ADADEF utility. For more information, read *ADADEF MODIFY REPTOR Function* in *Event Replicator for Adabas Reference Guide*.
- The ADARUN REPLICATION=YES parameter must be specified for the database from which you want files replicated. For more information, read *REPLICATION Parameter* in *Event Replicator for Adabas Reference Guide*.
- The ADARUN LRPL parameter must specify the size of the Adabas replication pool. For more information, read *LRPL Parameter* in *Event Replicator for Adabas Reference Guide*.
- Using the Adabas Online System or the REPLICATION parameter of the ADADBS utility, activate or deactivate replication for the files in the Adabas database. For more information, read *Adabas Online System Features Supporting Event Replicator for Adabas* in *Event Replicator for Adabas Administration and Operations Manual* or *ADADBS REPLICATION Function* in *Event Replicator for Adabas Reference Guide*.
- Use the ADALOD LOAD RPLTARGETID parameter or the Adabas Online System screens to identify the target Event Replicator Server to which replicated records from each database file should be transmitted. You can also optionally specify the primary key for replication (RPLKEY parameter) for the file and whether or not before images of data storage are collected for replication (RPLDSBI parameter) during an update command to the file. For more information, read *Adabas Online System Features Supporting Event Replicator for Adabas* in *Event Replicator for Adabas Administration and Operations Guide* or *ADALOD LOAD Parameters* in *Event Replicator for Adabas Reference Guide*.

The ADAREP utility provides information about the status of replication for the database and files, as well as for the Event Replicator Server.

Event Replicator Server Nucleus Replication Setup

The following replication setup should be performed for an Event Replicator Server.

- If you are defining the Event Replicator Server, be sure to specify the DEFINE REPTOR=YES parameter of the ADADEF utility to indicate that the database is an Event Replicator Server. For more information, read *ADADEF DEFINE REPTOR Function* in *Event Replicator for Adabas Reference Guide*.

  You can modify this setting later using the MODIFY REPTOR parameter of the ADADEF utility. For more information, read *ADADEF MODIFY REPTOR Function* in *Event Replicator for Adabas Reference Guide*.

- Event Replicator Server definitions must be defined for:
  - The target destinations to which you want replicated data sent.
  - Subscriptions that specify the replication processing, transformation, and filtering that should occur for specific files.
  - The messaging system input queues you will be using.

  Other definitions can also be specified, but are not required as, in many cases, defaults are provided. For complete information about Event Replicator Server definitions, read *Replication Definition Overview and Maintenance* in *Event Replicator for Adabas Administration and Operations Guide* or *Event Replicator Initialization Parameters* in *Event Replicator for Adabas Reference Guide*.

  These definitions can be specified using either or both DDKARTE initialization parameters in the Event Replicator Server startup job or definitions in the Replicator system file. Replicator system file definitions are maintained using the Adabas Online System's Adabas Event Replicator Subsystem. If your definitions are to be stored in the Replicator system file, you will need to use the ADALOD LOAD REPLICATOR parameter to load a Replicator system file into the Event Replicator Server. For more information, read *ADALOD LOAD Parameters* in *Event Replicator for Adabas Reference Guide*.

- The ADARUN RPLPARMS parameter must be specified to identify where the replication definitions should be read from. For more information, read *RPLPARMS Parameter* in the *ADALOD LOAD ParametersEvent Replicator for Adabas Reference Guide*.
- The ADARUN LRPL parameter must specify the size of the Event Replicator Server replication pool. For more information, read *LRPL Parameter* in *Event Replicator for Adabas Reference Guide*.
- Using the Adabas Online System or the REPTOR parameter of the ADADBS utility, activate or deactivate replication for the files in the Event Replicator Server. Note that if replication is deactivated for specific files, replication data from the Adabas nucleus for those files will not be processed by the Event Replicator Server. For more information, read *Adabas Online System Features Supporting Event Replicator for Adabas* in *Event Replicator for Adabas Administration and Operations Guide* or *ADADBS REPTOR Function* in *Event Replicator for Adabas Reference Guide*.

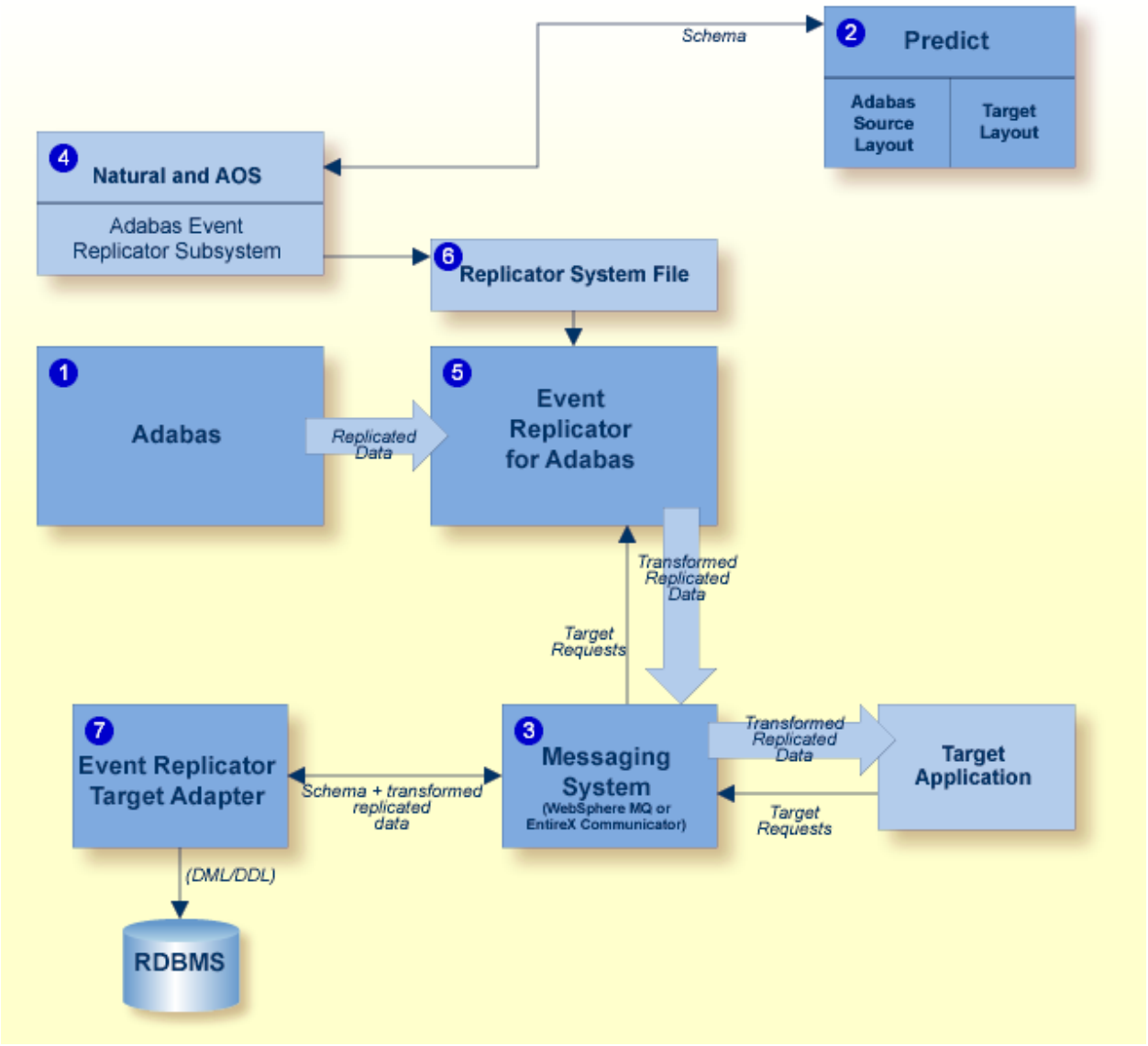
- Using the Adabas Online System or the REPTOR parameter of the ADADBS utility, activate or deactivate specific destination or subscription definitions. Note that if a destination definition is deactivated, replication data from the Adabas nucleus will not be sent to those destinations. Likewise, if a subscription definition is deactivated, replication data for the files covered by the subscription may not be processed (depending on what other subscription definitions are active). For more information, read *Adabas Online System Features Supporting Event Replicator for Adabas* in *Event Replicator for Adabas Administration and Operations Guide* or *ADADBS REPTOR Function* in *Event Replicator for Adabas Reference Guide*.
- If you will be using subscription logging, use the ADALOD LOAD SLOG parameter to load the SLOG system file into the Event Replicator Server. For more information, read in *Event Replicator for Adabas Reference Guide*.

  In addition, you will need to activate subscription logging for specific destinations. For more information, read *Using the Subscription Logging Facility* in *Event Replicator for Adabas Administration and Operations Guide*.

The ADAREP utility provides information about the status of replication for the Event Replicator Server and files, as well as for the Adabas database.

6 Getting Started

With a basic understanding of how the Event Replicator for Adabas delivers replicated Adabas data to a target application, you can begin to install, configure, and run the system. The following diagram depicts the basic products you need to install to get started using Event Replicator for Adabas:



The sequence in which these products should be installed and implemented is as follows:

1. **Adabas**
2. **Predict**
3. **Messaging System (webMethods EntireX or IBM WebSphere MQ)**
4. **Natural and Adabas Online System (AOS)**
5. **Event Replicator for Adabas**

6. **Specifying the Replication Definitions**
7. **Event Replicator Target Adapter**

Descriptions of each of these products are provided in this chapter.

1. Adabas

Adabas version 8.1.3 or later should already be installed, with ZAPs applied from the ARF*vrs*.MVSZAPS data set and any subsequent ARF*vrs*.MVSZX*nn* data sets (if they have been provided) for z/OS. Review the $README members of these data sets for details on the ZAPs.

Event Replicator code in this version can be used to replicate data from Adabas 8 databases. However, the Event Replicator Server must be running the same or a later version of Adabas as the Adabas database whose data is being replicated.

For information on installing Adabas, refer to your Adabas documentation.

2. Predict

If you will be using the Event Replicator for Adabas feature that allows you to generate global format buffers (GFBs) and a field table (GFFT) using Predict, Predict 4.5.1 must also be installed. If this version of Predict is not installed, you will not be able to use this feature. Global format buffers (GFBs) and an associated field table (GFFT) can be generated using the Adabas Event Replicator Subsystem.

Note: Generated global format buffers (and the corresponding field table) are required when using the Event Replicator Target Adapter to replicate data to a relational database, but they are not required otherwise.

For information on installing Predict, refer to your Predict documentation. For information on generating GFBs and the associated field tables using Adabas Event Replicator Subsystem, read *Generating a GFB using the Adabas Event Replicator Subsystem*, in *Adabas Event Replicator Subsystem User's Guide*.

3. Messaging System

A messaging system for use by Event Replicator for Adabas should be installed and configured unless you are planning on replicating data from one Adabas database to another (in which case Entire Net-Work can be used for communication between the two databases). On z/OS platforms, Event Replicator for Adabas supports either webMethods EntireX version 7.3.4 or later (preferred messaging system) or IBM WebSphere MQ version 5.3.1 or later.

On z/VSE platforms, Event Replicator for Adabas supports webMethods EntireX version 7.2.3.

On BS2000 platforms, Event Replicator for Adabas supports webMethods EntireX 7.1.4 or later.

Notes:

1. The term *MQSeries* is sometimes used in this documentation when referring to the product now known as *WebSphere MQ*.
2. The WebSphere MQ interface is not yet available on z/VSE.

For information on installing webMethods EntireX or IBM WebSphere MQ, refer to your webMethods EntireX or IBM WebSphere MQ documentation. Once the appropriate messaging system is installed, read the section entitled *Integrating the Messaging System with the Target Application* (in the *Event Replicator for Adabas Administration and Operations Guide*) for details on how to configure the messaging system for use with Event Replicator for Adabas.

4. Natural and Adabas Online System (AOS)

If you intend to use the Adabas Event Replicator Subsystem to maintain replication definitions, the following products must be installed:

1. Natural 4.2.3 or later must be installed. Replication itself is independent of your version of Natural. For information on installing Natural, refer to your Natural documentation.
2. A licensed copy of Adabas Online System (AOS) 8 is required to support Event Replicator for Adabas. Be sure to follow the installation instructions in the AOS manual for licensed versions.

   If you only use a demo copy of AOS, note that:

   - Only limited information concerning Event Replicator for Adabas is available to you with the demo copy.
   - The same versions and maintenance levels of the AOS demo code are required as for a licensed copy of AOS.

5. Event Replicator for Adabas

Install Event Replicator for Adabas, as described in in *Event Replicator for Adabas Installation Guide*.

The Event Replicator for Adabas installation procedure describes steps that will:

- Install the Event Replicator for Adabas software.
- Link WebSphere MQ (if that is your messaging system) to the Event Replicator.
- Create the Event Replicator Server.
- Explain how to load a Replicator system file (to store the Event Replicator Server definitions) onto the Event Replicator Server.
- Explain how to load the ARF*vrs*.INPL data set required for the Adabas Event Replicator Subsystem (if it is to be used) into the Natural system file libraries.
- Explain how to customize the Event Replicator Server startup JCL.
- Explain how to customize the Adabas nucleus job for replication.

6. Specifying the Replication Definitions

Event replication occurs based on replication definitions you specify. These definitions can be:

- Specified as initialization parameters, which are read from the DDKARTE statements of the Event Replicator Server startup job. For more information, read *Event Replicator Initialization Parameters* in *Event Replicator for Adabas Reference Guide*.
- Read from the Replicator system file at Event Replicator Server startup. (An initial Replicator system file is loaded onto the Event Replicator Server during installation.) The definitions are maintained in the Replicator system file using the Adabas Event Replicator Subsystem. For more information, read *Maintaining Replication Definitions Using the Adabas Event Replicator Subsystem* in *Adabas Event Replicator Subsystem User's Guide*.

Note: If you use a Replicator system file to store your replication definitions, you can also use the RPLREFRESH command to refresh resource definitions in your Event Replicator Server configuration while the Event Replicator Server is still running. For more information, read *RPLREFRESH Command*, in *Event Replicator for Adabas Administration and Operations Guide*.

You can elect to use one or both methods, depending on what works best for you. The method used is controlled by ADARUN parameter `RPLPARMS`.

Once the replication definitions have been correctly specified, you can activate replication.

7. Event Replicator Target Adapter

The Event Replicator Target Adapter is an additional Event Replicator product that can be used to transform and apply replicated data to a relational database, such as DB2, MySQL, Oracle, SQL Server, or Sybase.

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

Once the rest of the Event Replicator products have been installed and configured, you can install and configure Event Replicator Target Adapter, as described in *System Requirements* and *Installing the Event Replicator Target Adapter* in *Event Replicator Target Adapter Installation Guide* as well as in *Event Replicator Target Adapter Administration* in *Event Replicator Target Adapter Administrator's Guide*.

7 Useful Documentation Links

To learn about:	Read:
The current release of Event Replicator for Adabas	*Event Replicator for Adabas Release Notes* in *Event Replicator for Adabas Release Notes*.
The prerequisites and procedures for installing Event Replicator for Adabas	*Event Replicator for Adabas Installation* in *Event Replicator for Adabas Installation Guide*.
The prerequisites and procedures for installing Entire Net-Work Administration and Entire Net-Work Client	*Entire Net-Work Administration documentation* in *Entire Net-Work Administration Installation Guide* and *Entire Net-Work Client Administration* in *Entire Net-Work Client Installation and Administration Guide*.
The prerequisites and procedures for installing Event Replicator Target Adapter	*System Requirements* and *Installing the Event Replicator Target Adapter* in *Event Replicator Target Adapter Installation Guide* *Event Replicator Target Adapter Administration* in *Event Replicator Target Adapter Administrator's Guide* .
■ Understanding replication definitions ■ Subscription user exits ■ Using subscription logging ■ Using transaction logging ■ Integrating the messaging system with the target application ■ Operator commands pertinent to replication ■ Adabas Online System (AOS) screens pertinent to Event Replicator for Adabas	*Event Replicator for Adabas Administration and Operations* in *Event Replicator for Adabas Administration and Operations Guide*.
Maintaining replication definitions in the Replicator system file	*Using the Adabas Event Replicator Subsystem* in *Adabas Event Replicator Subsystem User's Guide*.

To learn about:	Read:
■ ADARUN initialization parameters pertinent to replication ■ Adabas and Event Replicator for Adabas utilities pertinent to replication ■ Event Replicator initialization parameters you can use in the Event Replicator Server startup job to specify the definitions needed to run Event Replicator for Adabas successfully ■ Use of the C5 command to submit messages to Event Replicator Server destinations	*Event Replicator for Adabas Reference* in *Event Replicator for Adabas Reference Guide*
Configuring your messaging system (webMethods EntireX or WebSphere MQ) for use with Event Replicator for Adabas.	*Integrating the Messaging System with the Target Application* (in the *Event Replicator for Adabas Administration and Operations Guide*).
The information your application programmer needs to enable your target application to interface with the Event Replicator for Adabas.	*Event Replicator for Adabas Programmer's Reference* in *Event Replicator for Adabas Application Programmer's Reference Guide*
The messages and codes that can occur during Event Replicator for Adabas processing.	*Event Replicator for Adabas Messages and Codes* in *Event Replicator for Adabas Messages and Codes Manual*
Using Event Replicator Target Adapter to replicate data to an RDBMS.	*Using the Event Replicator Target Adapter* in *Event Replicator Target Adapter User Guide*.

Index

S

T

U

W