Event Replicator for Adabas

Administration and Operations

Version 4.1.1

October 2022

ADABAS & NATURAL

This document applies to Event Replicator for Adabas Version 4.1.1 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-ADMIN-411-20221108

Table of Contents

Preface

To use Event Replicator for Adabas, you must supply various definitions and user exits. This document describes how to create those definitions and user exits as well as other administrative tasks and operations an Event Replicator for Adabas user can perform.

This document is organized as follows:

Replication Definition Overview and Maintenance	Provides an overview of the kinds of replication definitions you must set up to use the Event Replicator for Adabas and how you can do so.
Maintaining Replication Definitions Using the Adabas Event Replicator Subsystem	Describes the Adabas Event Replicator Subsystem and how to use it to maintain replication definitions in the Replicator system file.
Replaying Replicated Records	Describes synchronized, unsynchronized, and replay-only replay processing with the Event Replicator for Adabas.
Using the Subscription Logging Facility	Describes how to use the *subscription logging facility*, or *SLOG facility*, to ensure that data replicated to specific destinations is not lost if problems occur on those destinations.
Reducing the Risk of Replication Pool Overflows	Describes how you can reduce the risk of the 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.
Using Transaction Logging	Describes how to use transaction logging to log transaction processing and status change events that occur while the Event Replicator Server is operational.
Using the Event Replicator Server Subscription User Exit	Describes the Event Replicator Subscription User Exit.
Messaging System Integration	Provides guidelines to integrating WebSphere MQ and webMethods EntireX messaging systems with the Event Replicator for Adabas.
Creating a Sequential Output File	Describes how to create a sequential output file of replicated transactions.
Replicating Security Definitions	Describes the requirements, restrictions, and peculiarities of replicating security definitions from the Adabas security file in the database.
Pertinent Operator Commands	Describes the operator commands pertinent to Event Replicator for Adabas.
AOS Features Supporting Event Replicator for Adabas	Describes Adabas Online System (AOS) features that support Event Replicator for Adabas.

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 Replication Definition Overview and Maintenance

To use Event Replicator for Adabas and customize its processing, you must supply various replication definitions. 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. 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*, elsewhere in this 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`.

Definition Descriptions

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.

Definition Type	Defines	How many definitions are required?
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 Event Replicator Subsystem, read *Maintaining Subscription Definitions* in *Adabas Event Replicator Subsystem User's Guide*.	Required. At least one subscription definition is required for data replication to occur.
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*	Required. At least one SFILE definition is required for data replication to occur.

Definition Type	Defines	How many definitions are required?
	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*.	
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 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*.	Not required. At least one IQUEUE definition is required for every EntireX Communicator or WebSphere MQ target you 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*	Not required. No resend buffer definition is required. If you elect to retransmit a transaction, at

Definition Type	Defines	How many definitions are required?
	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*.	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.

Definition Specification Sequence

The applicable definitions and the sequence in which they should be set up varies, depending on the destination. Five destinations are supported in the Event Replicator for Adabas: Adabas, webMethods EntireX, IBM WebSphere MQ, File, and Null. The following table describes each of these destination types and lists the definitions that apply to the destination in the order in which they should be defined.

Destination Type	Description	Definition List and Order of Creation
Adabas	Data is replicated to one or more Adabas files.	1. destination definitions, as necessary (referenced by subscription definitions) 2. global format buffer definitions, if needed (can be referenced by the SFILE definitions) 3. subscription definition, including at least one SFILE definition 4. one or more SFILE definitions (included in the subscription definition) 5. initial-state definition
webMethods EntireX	Replicated data is written to an output queue via webMethods EntireX.	1. destination definitions, as necessary (referenced by subscription definitions) 2. IQUEUE definition 3. global format buffer definitions, if needed (can be referenced by the SFILE definitions) 4. subscription definition, including at least one SFILE definition

Destination Type	Description	Definition List and Order of Creation
		5. one or more SFILE definitions (included in the subscription definition) 6. initial-state definition
File	Replicated data is written to the CLOG, using TLOG URBLTDOD records.	1. destination definitions, as necessary (referenced by subscription definitions) 2. global format buffer definitions, if needed (can be referenced by the SFILE definitions) 3. subscription definition, including at least one SFILE definition 4. one or more SFILE definitions (included in the subscription definition) 5. initial-state definition
WebSphere MQ	Replicated data is written to an output queue via IBM WebSphere MQ.	1. destination definitions, as necessary (referenced by subscription definitions) 2. IQUEUE definition 3. global format buffer definitions, if needed (can be referenced by the SFILE definitions) 4. subscription definition, including at least one SFILE definition 5. one or more SFILE definitions (included in the subscription definition) 6. initial-state definition
Null	Data replication is tested without actually sending the data to a destination.	1. destination definitions, as necessary (referenced by subscription definitions) 2. global format buffer definitions, if needed (can be referenced by the SFILE definitions) 3. subscription definition, including at least one SFILE definition 4. one or more SFILE definitions (included in the subscription definition) 5. initial-state definition

3 Replaying Replicated Records

Replay processing is used to deliver replication data that has already been delivered or should have been delivered to the target application. Using replay processing, you can read the sequential (merged) PLOG of an Adabas database and, based on the parameters you specify, send related data to one or more Event Replicator Servers. The Replay Utility, ADARPL, is the mechanism through which Event Replicator for Adabas supports replay processing. For more information about this utility, read *ADARPL Utility: PLOG Replication Replay*, in the *Event Replicator for Adabas Reference Guide*.

Replay processing in Event Replicator for Adabas can be run in any of three modes: *synchronized*, *unsynchronized*, and *replay-only*. These modes are described in ***Understanding Replay Modes***, elsewhere in this chapter.

In addition, be sure to read about ADARPL prerequisites, described in *ADARPL Prerequisites* in *Event Replicator for Adabas Reference Guide*.

When Is Replay Necessary?

Some reasons why you might need to replay replicated records are:

- The target application does not process the replicated data correctly or has some sort of failure.
- A failure occurs with the message queue tool.
- The Event Replicator replication pool fills up. This might occur if the message queue tool is down for a prolonged period.
- The Adabas replication pool fills up. This might occur if the Event Replicator Server is down for a prolonged period.
- Replication was turned off for a particular file, subscription, or destination for some reason.
- You need to send the data to another destination.

In all of these cases, you can use Event Replicator for Adabas's replay processing to redeliver the replication data that was lost.

Understanding Replay Modes

When you invoke replay processing, you must select a replay mode. Replay processing in Event Replicator for Adabas can be run in any of three modes: *synchronized*, *unsynchronized*, and *replay-only*. All modes replay replicated data reconstructed from protection data in the PLOG. However, they vary in the following ways:

- They vary in the steps that must be taken to initiate and run them.
- They vary in how they handle new transactions from Adabas while replay processing is occurring.

This section covers the following topics:

Synchronized Mode

Synchronized mode is the recommended mode. During synchronized replay processing, the Event Replicator Server suspends new Adabas transactions. When the replay processing is complete, the new Adabas transactions are automatically synchronized with the replayed data. This mode is only available using the online Adabas Event Replicator Subsystem screens.

The net effect of synchronized mode replay processing is that the target application receives replicated data reconstructed from the PLOG data sets before it receives any new replicated data produced by Adabas. The data is then processed in the chronologically correct sequence.

When running a synchronized replay:

- The Event Replicator Server will reactivate all files, subscriptions, and destinations involved in the replay that are inactive.

  Note: Files that are active in the Event Replicator Server but inactive in the source Adabas nucleus will not be reactivated. If you would like a file in this state to be reactivated during the synchronized replay, set the file to inactive status in the Event Replicator Server before staring the synchronized replay.

- All new Adabas data for the subscriptions and destinations involved in the replay is held in the Event Replicator replication pool until the replay processing is completed.

  If an SLOG has been defined, all new data is written to the SLOG instead. The advantage of using an SLOG is that replay processing makes less use of the replication pool, thus reducing the risk of a replication pool overflow.

- When replay processing is complete, the new data held in the replication pool is delogged and processed as usual.

  If an SLOG was used, the Event Replicator Server reads the held transactions from the SLOG, processes them as usual, and deletes them. If additional new transactions are received while this delogging process is occurring, they are also written to the SLOG until the delogging process has caught up with the logging process.

- If synchronized replay processing fails, the Event Replicator Server will deactivate the files, subscriptions, and destinations involved in the replay that it originally activated.

- If an SLOG has *not* been defined and synchronized replay processing takes so long that the new replication data from Adabas fills up the replication pool, the Event Replicator Server will discard the new data and automatically change the replay processing to ***replay-only mode***.

- While replication data is stored in the SLOG file, the Event Replicator Server will not shut down normally (using the ADAEND command). It can be brought down using a HALT command and it can be canceled or otherwise terminated abnormally. If during the next session, the Event Replicator Server detects data on the SLOG originating from a replay process that took place in the previous session, it deletes this leftover data from the SLOG.

When synchronized replay processing is initiated, a token is assigned to the replay process and can be referenced using the ADARPL batch utility. For information on running the ADARPL utility, read *ADARPL Utility: PLOG Replication Replay* in *Event Replicator for Adabas Reference Guide*.

Unsynchronized Mode

During unsynchronized replay processing, the new Adabas transactions are processed concurrently with the replayed transactions, but no synchronization is performed. This mode is only available through batch runs of the ADARPL utility. For information on running the ADARPL utility, read *ADARPL Utility: PLOG Replication Replay* in *Event Replicator for Adabas Reference Guide*.

The net effect of unsynchronized mode replay processing is that the target application receives replicated data reconstructed from the PLOG data sets at the same time and interleaved with any new replicated data produced by Adabas. The data is not processed in the chronologically correct sequence.

When running an unsynchronized replay:

- The Event Replicator Server requires that all files, subscriptions, and destinations involved in the replay be active. It will not perform any automatic activation of these resources.
- All new Adabas data for the subscriptions and destinations involved in the replay are processed as soon as they are received.

When unsynchronized replay processing is initiated, a token is assigned to the replay process. This token can be used to cancel the replay process, if necessary.

Replay-Only Mode

During replay-only processing, replay processing is performed on the replicated transactions in the PLOG, but any new Adabas transactions for the files, subscriptions, and destinations involved in the replay are discarded. This mode is only available using the online Adabas Event Replicator Subsystem screens.

The net effect of replay-only mode replay processing is that the target application receives only replicated data reconstructed from the PLOG data sets. When replay processing is complete, another replay process should be initiated to pick up any new Adabas transactions discarded for the files, subscriptions, and destinations involved in the replay.

When running a replay-only mode replay:

- The Event Replicator Server requires that some or all of the files and subscriptions involved in the replay must be inactive before replay processing starts so that no replication data from Adabas can be processed using these resources.
- A replay-only mode replay processing will be disallowed if one or more destinations involved are closed.
- When Event Replicator Server starts replay-only mode replay processing, it activates the necessary inactive files, subscriptions, and destinations so that data from the PLOGs only can use them, but blocks and discards all the new data from Adabas for those files, subscriptions, and destinations.
- When processing is complete, the Event Replicator Server deactivates the files, subscriptions, and destinations that were inactive when replay-only mode processing was initiated.

Note: Files that are active in the Event Replicator Server but inactive in the source Adabas nucleus are not considered inactive in this context. If you would like a file in this state to be activated during the replay-only replay, set the file to inactive status in the Event Replicator Server before starting the replay-only replay.

When replay-only mode replay processing is initiated, a token is assigned to the replay process and can be referenced using the ADARPL batch utility. For information on running the ADARPL utility, read *ADARPL Utility: PLOG Replication Replay* in *Event Replicator for Adabas Reference Guide*.

Prerequisites

Before you can initiate replay processing using the Adabas Event Replicator Subsystem, the following prerequisites must be met:

- Verify that the correct PLOG is used for the run and that it is a sequential PLOG, not a dual PLOG. You can use the PLOG data set list to help determine which PLOG data sets should be used. For more information, read *Reviewing and Managing the PLOG Data Set List*, in *Adabas Event Replicator Subsystem User's Guide*.
- Verify that the target application can handle duplicate records.
- The Adabas database must be active. The Replay Utility will attempt to issue a call to Adabas to obtain the GCB, FCBs, and FDTs from the nucleus.
- Verify that all ADARPL utility prerequisites are satisfied. For more information, read *ADARPL Prerequisites* in *Event Replicator for Adabas Reference Guide*.

Identifying Replay Processing Resources

Prior to initiating a replay process, we recommend that you identify resources involved in the replay process. When you initiate a replay request, specific resources are requested. Data from the PLOG is only processed by the resources involved. If multiple resources of different types (subscriptions, destinations, or files) are requested, data is only replayed for the resources common to all requested resources. This section explains this more fully.

To identify the replay resources actually used by the replay process, you must examine the data flow paths through the Event Replicator Server that are initiated by each resource requested for the replay. Each data flow path is defined as a unique *one file-one subscription-one destination* combination, such that the subscription takes data from the file and delivers it to the destination.

This examination process is best described through a series of examples, using the following resource definitions (where S*x* denotes a subscription name, F*x* denotes a file number, and D*x* denotes a destination name):

1. S1: F1, F2, D1, D2

   Subscription S1 includes processing information (SFILE definitions) for files F1 and F2 to destinations D1 and D2.

2. S2: F2, F3, D2, D3.

   Subscription S2 includes processing information (SFILE definitions) for files F2 and F3 to destinations D2 and D3.

Eight unique data flow paths are identified by these definitions:

- F1, S1, D1
- F1, S1, D2
- F2, S1, D1
- F2, S1, D2
- F2, S2, D2
- F2, S2, D3
- F3, S2, D2
- F3, S2, D3

The remainder of this section uses these example definitions to describe the data flow paths and ultimate effect on replay processing in four different replay scenarios:

- Replaying Only One Resource
- Replaying Multiple Resources of One Type

Replaying Only One Resource

If you only specify one resource in the replay request, the effect of the replay processing is determined by the union of the constituents of all data flow paths going through the one resource.

For example, based on the **example definitions** described earlier in this section, suppose you specify D2 as the resource for the replay request. In this case, the resources involved in the replay are F1, F2, F3, S1, S2, and D2. The data flow paths are:

- F1, S1, D2
- F2, S1, D2
- F2, S2, D2
- F3, S2, D2

Any transactions flowing through these data paths will be replayed.

Replaying Multiple Resources of One Type

If you specify multiple resources of one type (destination, subscription, or file) in the replay request, the effect of the replay processing is determined by the union of the constituents of all data flow paths going through any specified resource.

For example, based on the **example definitions** described earlier in this section, suppose you specify D1 and D3 as the resources for the replay request. In this case, the resources involved in the replay are F1, F2, F3, S1, S2, D1, and D3. The data flow paths are:

- F1, S1, D1
- F2, S1, D1
- F2, S2, D3
- F3, S2, D3

Any transactions flowing through these data paths will be replayed.

Replaying Multiple Resources of Different Types

If you specify multiple resources of different types (destination, subscription, or file) in the replay request, the effect of the replay processing is determined by the union of the constituents of all data flow paths that are common to the data flow paths grouped by type.

For example, based on the **example definitions** described earlier in this section, suppose you specify S1 and D2 as the resources for the replay request. In this case, the resources involved in the replay are F1, F2, S1, and D2. But the data flow paths used for the replay must be the data flow paths common to both the S1 subscription and the D2 destination.

The data flow paths for the S1 subscription are:

- F1, S1, D1
- F1, S1, D2
- F2, S1, D1
- F2, S1, D2

The data flow paths for the D2 destination are:

- F1, S1, D2
- F2, S1, D2
- F2, S2, D2
- F3, S2, D2

However, the only data flow paths the S1 subscription and the D2 destination share are:

- F1, S1, D2
- F2, S1, D2

Any transactions flowing through these two data paths will be replayed.

Replaying Resources With Nothing in Common

It is an error to request a replay resource that has no data flow paths in common with other requested resources. When this happens, the entire replay request will be rejected.

For example, based on the **example definitions** described earlier in this section, suppose you specify S1, D2, and D3 as the resources for the replay request. In this case, the resources involved in the replay should be F1, F2, S1, D2, and D3. But, as you will see below, the D3 data flow paths have nothing in common with the data flow paths for S1 and D2:

The data flow paths for the S1 subscription are:

- F1, S1, D1

- F1, S1, D2
- F2, S1, D1
- F2, S1, D2

The data flow paths for the D2 destination are:

- F1, S1, D2
- F2, S1, D2
- F2, S2, D2
- F3, S2, D2

The data flow paths for the D3 destination are:

- F2, S2, D3
- F3, S2, D3

Since the D3 data flow paths are only for subscription S2 and the S1 data flow paths do not include D3, there is no common data flow path for this replay request and the replay request is in error.

Initiating Replay Processing

Replay processing can be initiated in any of the following ways.

1. It can be initiated in a batch job using the ADARPL utility without specifying a replay process token. In this case, an *unsynchronized* replay is initiated. For complete information on initiating replay processing using the ADARPL utility without a replay process token, read *ADARPL Utility: PLOG Replication Replay* in *Event Replicator for Adabas Reference Guide*, using the syntax described in *Syntax for Initiating ADARPL Without A Token*.
2. It can be initiated using a replay process token produced in the Adabas Event Replicator Subsystem. This method involves a combination of the Adabas Event Replicator Subsystem and a batch ADARPL utility job. In this case, you first use the Adabas Event Replicator Subsystem to generate a *synchronized* or *replay-only* replay request. The replay request is assigned a token that you then use in the batch ADARPL utility job. For information on initiating synchronized and replay-only replay requests using the Adabas Event Replicator Subsystem and ADARPL, read *Initiating a Replay Request Using the Adabas Event Replicator Subsystem*, in *Adabas Event Replicator Subsystem User's Guide*.
3. It can be initiated using a replay process token produced by a standalone application programming interface (API) provided by Software AG. In this case, you first use the API to generate a *synchronized* or *replay-only* replay request. For complete information on initiating replay processing using the standalone API, read ***Initiating a Replay Request Using the Standalone API*** elsewhere in this section.

4. It can automatically be initiated by the Event Replicator Server whenever a replay process token is produced by the Adabas Event Replicator Subsystem or by a standalone API provided by Software AG. In this case, specific JCL code must be incorporated into the Event Replicator Server startup JCL and, if running, the Event Replicator Server must be stopped and restarted. Then you must generate a *synchronized* or *replay-only* replay request using the Adabas Event Replicator Subsystem or the Software AG-supplied standalone API. For complete information on automating replay processing, read ***Automating Replay Processing*** elsewhere in this section.

The remainder of this section describes the steps for initiating a replay request using the standalone API and lists what's returned from a run of the standalone API:

- Initiating a Replay Request Using the Standalone API
- What's Returned from a Standalone API Run

Initiating a Replay Request Using the Standalone API

To generate a synchronized or replay-only replay request using the standalone API, complete the following steps:

1 In the Natural library provided by Software AG, edit the library object ARFP003. This library object is a sample program you can use to call the Event Replicator for Adabas program ARFN003.

If you use your own program instead of ARFP003, included the SYSRPTR library (the INPL library) in the library concatenation for your program job when calling the ARFN003 program. The SYSRPTR library provides a routine that is invoked by ARFN003. You can use the Natural subprogram, USR1025N, to do this.

2 Search for the comment "Set values below" in ARFP003. A sample of this section of the program is provided below and the "Set values below" section is highlighted in green.

```
1660 *
1670 *  Set values below.
1680 *
1690 #MODE       = 'S'
1700 #DBID       = H'0001'
1710 #REPTOR-ID  = H'0002'
1720 #FROM-DATE  = '2007/11/01'
1730 #FROM-TIME  = '01:02:03'
1740 #TO-DATE    = '2007/11/01'
1750 #TO-TIME    = '21:22:23'
1760 #START-DATE = '          '
1770 #START-TIME = '        '
1780 #DEST-LIST  = 'NULL01'
1790 #SUB-LIST   = ' '
1800 #AUTOMATED  = 'N'
1810 #TIMEOUT    =  900
1820 *
```

```
1830 CALLNAT 'ARFN003'
1840          #MODE
1850          #DBID
1860          #REPTOR-ID
1870          #FROM-DATE
1880          #FROM-TIME
1890          #TO-DATE
1900          #TO-TIME
1910          #START-DATE
1920          #START-TIME
1930          #DEST-LIST
1940          #SUB-LIST
1950          #TOKEN
1960          #RESPONSE
1970          #SUBCODE
1980          #AUTOMATED
1990          #TIMEOUT
2000          #MESSAGE
2010 *
2020 WRITE '----- COMPLETED ------'
2030 WRITE 'RESPONSE:' #RESPONSE
2040 WRITE 'SUBCODE: ' #SUBCODE
2050 WRITE 'TOKEN:   ' #TOKEN
2060 WRITE 'MSG:     ' #MESSAGE
2070 WRITE '------ E N D ---------'
2080 *
2090 END
***** End of list *****
```

3 Supply values for the variables to ARFP003 listed below the comment. Descriptions of all variables are provided in the table below as well as in the ARFP003 program itself.

Caution: Do *not* modify the order of the variables as listed in the `CALLNAT 'ARFN003'` section and below of the program. If you do, the API will either fail or your results will not be valid.

Variable Name	Description
#AUTOMATED	Indicate whether or not you want the replay automated or not. Valid values are "Y" (perform an automated replay) or "N" (do not perform an automated replay). An automated replay will automatically perform **steps 7 through 9** of this procedure. A non-automated replay will not perform these steps automatically, and you will need to perform them manually. For complete information about automating replay processing, read ***Automating Replay Processing***, elsewhere in this section. **Note:** If the `RECORDPLOGINFO` parameter has been set to NO, you cannot run an automated replay.
#DBID	The database ID of the Adabas database from which you want replicated transactions replayed.

Variable Name	Description
#DEST-LIST	A list of destinations for which the replay request should be initiated. When the replay request is initiated, transactions will be replayed that were originally destined for the destinations on this list. Up to 60 eight-byte entries can be specified in the list.
#MODE	The replay mode to be used. Valid values are "S" (synchronized) or "R" (replay only). For complete information about the differences between replay modes, read ***Understanding Replay Modes***, elsewhere in this section.
#FROM-DATE #FROM-TIME	The date and time from which replicated transactions should be replayed. Dates should be specified in YYYY/MM/DD format; times should be specified in HH:MM:SS format. Replay processing will start with transactions in the PLOG that ended at or after this date and time. From dates and times must be earlier than the current date and time and earlier than the specified end date and time.
#REPTOR-ID	The database ID of the Event Replicator Server to which the replayed transactions will be sent. This is also the server with the Replicator system file that stores the destination and subscription definitions requested for replay processing.
#START-DATE #START-TIME	The date and time of the PLOG entries that should be used as a starting point for the replay processing. This date and time are used to identify the PLOG with which to start replay processing. Dates should be specified in YYYY/MM/DD format; times should be specified in HH:MM:SS format. Replay processing will search the PLOG with this start date and time first for records that match the other replay processing criteria listed on this screen. A start date and time must be specified if an automated replay is requested.
#SUB-LIST	A list of subscriptions for which the replay request should be initiated. When the replay request is initiated, transactions will be replayed that were originally initiated by the subscriptions on this list. Up to 60 eight-byte entries can be specified in the list.
#TIMEOUT	Optionally, specify the length of time, in seconds, at which the replay request should time out.
#TO-DATE #TO-TIME	The date and time to which replicated transactions should be replayed. Dates should be specified in YYYY/MM/DD format; times should be specified in HH:MM:SS format. Replay processing will stop with transactions in the PLOG that ended before this date and time. End dates and times must be later than the specified start date and time. If no end date and time are specified, the end time is the current time (the time the replay request is issued).

4 When all variables have been supplied to your satisfaction, save ARFP003

5 In an application you have created, add a call to ARFP003 and save your application.

6 Run your application.

The replay request is generated and a replay token is assigned to it. This replay token is displayed in an API message and in the Event Replicator Server job log.

Make note of this token number as it is used in **step 9** if you are initiating replication replay using a batch ADARPL job.

If you have automated replication replay processing, this token number is picked up automatically by the generated replay jobstream and you can skip the remaining steps in this procedure. For complete information about automating replay processing, read ***Automating Replay Processing***, elsewhere in this section.

For complete information about what's returned from this run, read ***What's Returned from a Standalone API Run***, elsewhere in this section.

7 This step should be performed only if the #AUTOMATED variable is set to "N" (an automated replay is not requested).

If necessary, issue a force-end-of-PLOG request to the Adabas database and wait until the resulting PLCOPY job has copied or merged the latest PLOG data set. This is necessary only when the PLOG for the selected replay end date and time has not yet been copied or merged, for example, if no end date and time were specified in the replay request.

8 This step should be performed only if the #AUTOMATED variable is set to "N" (an automated replay is not requested).

Identify the sequential PLOG data sets that contain the protection data for the replicated records you need replayed. The PLOG data sets must build a complete sequence from the PLOG that includes the replay processing start time to the latest PLOG you copied or merged in the previous step.

9 This step should be performed only if the #AUTOMATED variable is set to "N" (an automated replay is not requested).

Run an ADARPL utility job, using the syntax described in *Syntax for Initiating ADARPL With A Token* in *Event Replicator for Adabas Reference Guide*. Be sure to specify:

- A concatenated list of the PLOG data sets you identified in the previous step.
- The replay request token assigned in **step 6**. This token should be specified in the ADARPL `TOKEN` parameter.
- The Event Replicator Server ID of the Event Replicator Server to which the replayed transactions should be sent. This token should be specified in the ADARPL `RPLTARGETID` parameter.

For more information about using the ADARPL Utility, in general, read *ADARPL Utility: PLOG Replication Replay* in *Event Replicator for Adabas Reference Guide*.

What's Returned from a Standalone API Run

The following parameters may be returned by the standalone API:

Parameter Name	Description
#MESSAGE	A message associated with the response code or subcode.
#RESPONSE	The response code issued from an attempt to initiate the replay.
#SUBCODE	The subcode associated with the response code (#RESPONSE).
#TOKEN	The token number assigned the initiated replay.

Cancelling Replay Processing

You can cancel a replay process if you decide that it is not producing the desired results. However, you will then have to determine how to get the replicated data back in sync with the source database.

To cancel replay processing:

- Issue the RPLCLEANUP command. This command will stop replay processing (if it is running when the RPLCLEANUP command is entered) and will clean up any open transactions in the Event Replicator Server that are associated with replay processing. For more information, read *RPLCLEANUP Command* in *Event Replicator for Adabas Administration and Operations Guide*.

Automating Replay Processing

Automated ADARPL processing requires that you specify two additional JCL statements in the Event Replicator Server nucleus startup JCL: DDJCLIN and DDJCLOUT. This section describes the steps you need to perform to set up automated ADARPL processing and describes the automated replay JCL skeleton and provides some sample JCL.

- Sample Automated Replay JCL with Skeleton Statements

Initiating Automated ADARPL Processing

To initiate automated ADARPL processing:

1 Create an appropriate automated replay JCL skeleton. This skeleton can be coded directly in the Event Replicator Server startup JCL or in a sequential data set and will be tailored by the Event Replicator Server during automated replay processing. The **sample JCL** given elsewhere in this section provides an example of coding the automated replay JCL skeleton directly in the Event Replicator Server startup job. For more information about the skeleton itself, read ***The Automated ADARPL Skeleton***, elsewhere in this section.

2 Add a DDJCLIN JCL statement to the Event Replicator Server nucleus startup JCL. This JCL statement identifies the sequential data set containing the automated replay JCL skeleton or specifies the skeleton itself. The **sample JCL** given elsewhere in this section provides an example of coding the automated replay JCL skeleton directly in the Event Replicator Server startup job.

3 Add a DDJCLOUT JCL statement to the Event Replicator Server nucleus startup JCL. This JCL statement identifies the location of the generated jobstream for automated replay processing. As the Event Replicator Server tailors the automated replay JCL skeleton, it writes the generated jobstream to the file identified by the DDJCLOUT JCL statement in 80-byte records. The file is closed once the skeleton has been completely processed.

   The DDJCLOUT JCL statement may specify a sequential data set or, in z/OS systems, it may direct the output to the internal reader for immediate job processing. The z/OS internal reader is requested by coding `SYSOUT=(*,INTRDR)` on the DDJCLOUT JCL statement.

4 If the Event Replicator Server is running, stop and restart it to pick up the new JCL specifications.

5 Generate replay process tokens in any of the following ways:

   - Generate a replay process token using the Adabas Event Replicator Subsystem. Using the Adabas Event Replicator Subsystem, generate a *synchronized* or *replay-only* replay request. For information on initiating synchronized and replay-only replay requests using the Adabas Event Replicator Subsystem and ADARPL, read *Initiating a Replay Request Using the Adabas Event Replicator Subsystem*, in *Adabas Event Replicator Subsystem User's Guide*.
   - Generate a replay process token using a standalone application programming interface (API) provided by Software AG. Using the API, generate a *synchronized* or *replay-only* replay request. For complete information on initiating replay processing using the standalone API, read ***Initiating a Replay Request Using the Standalone API*** elsewhere in this section.

   Note: In all replay requests, be sure to turn automation *on*, by specifying "Y" for the **Automated** field in the Adabas Event Replicator Subsystem or by specifying "Y" for the #AUTOMATED variable in the Software AG-supplied API.

Once a replay request is generated, it is assigned a token that will automatically be detected by the Event Replicator Server and used for automated replay processing.

The Automated ADARPL Skeleton

The automated replay JCL skeleton can be coded directly in the Event Replicator Server startup JCL or in a sequential data set. It is a jobstream containing 80-byte records that include platform-dependent JCL and utility control statements. Designated points in the jobstream will be automatically tailored by the Event Replicator Server when an ADARPL token is encountered and if automated replay is specified in the associated replay requests. Only columns 1 to 72 of the skeleton jobstream are examined for tailoring.

The JCL skeleton should be similar to the JCL below. Tailoring will occur at the points designated by keywords beginning with "%":

```
//ADARPL  JOB
//*
//*  ADARPL: Sample JCL skeleton for automated ADARPL generation
//*
//RPL    EXEC  PGM=ADARUN
//STEPLIB  DD  DISP=SHR,DSN=ADABAS.Vvrs.LOAD       <=== Adabas load lib
//DDASSOR1 DD  DISP=SHR,DSN=EXAMPLE.DB%DBID.ASSOR1 <=== Adabas ASSO
//DDDRUCK  DD  SYSOUT=*
//DDPRINT  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//*  The following record will be replaced with a concatenation of
//*  sequential PLOG data sets
%SEQUENTIAL
//DDCARD   DD  *
ADARUN PROG=ADARPL,DBID=%DBID,SVC=svc,DEVICE=3390
/*
//DDKARTE  DD  *
ADARPL REPLAY
ADARPL LRPL=1500K
*    The following record will be replaced with ADARPL control
*    statements
%KARTE
/*
```

The following keywords in this skeleton will be tailored:

Keyword	Tailoring Description
%DBID	This keyword may appear in any position on any record in the ADARPL skeleton. It identifies locations in the jobstream that should be replaced with the five-byte numeric DBID specified in the replay request identified by the ADARPL token. The DBID is padded with leading zeros.

Keyword	Tailoring Description
%SEQUENTIAL	This keyword must appear in column 1 and be the only text in a record in the ADARPL skeleton. It identifies the location in the jobstream where platform-dependent PLOG JCL statements will be generated by the Event Replicator Server.
%KARTE	This keyword must appear in column 1 and be the only text in a record in the ADARPL skeleton. It identifies the location where all other ADARPL utility parameters will be generated by the Event Replicator Server. The %KARTE keyword does not generated any platform-dependent JCL, only additional ADARPL control statements (typically the RPLTARGETID and TOKEN parameters). It must be preceded by JCL that identifies the file and an initial ADARPL statement that invokes ADARPL utility processing and , optionally, provide values for the NU and LRPL parameters. More than one ADARPL statement can precede and follow the %KARTE keyword. For complete information about the ADARPL syntax, read *Syntax for Initiating ADARPL With A Token (Synchronized and Replay-only Replay Modes)* , in the *Event Replicator for Adabas Reference Guide*.

Sample Automated Replay JCL with Skeleton Statements

The following z/OS sample shows the additions you need to make to the Event Replicator Server startup JCL to support automated replay processing. The automated replay skeleton JCL is read in as specified by the DDJCLIN statement, tailored, and then written out as specified by the DDJCLOUT statement. In this example, the skeleton is included directly in the startup JCL and is delimited by the " ## " characters. In addition, the tailored output is directed to the z/OS internal reader for immediate job processing, as directed by the DDJCLOUT statement.

```
//DDJCLOUT DD  SYSOUT=(*,INTRDR)                    <=== Job output
//DDJCLIN  DD  DATA,DLM='##'                        <=== JCL skeleton
//ADARPL  JOB
//*
//*  ADARPL: Sample JCL skeleton for automated ADARPL generation
//*
//RPL    EXEC  PGM=ADARUN
//STEPLIB  DD  DISP=SHR,DSN=ADABAS.Vvrs.LOAD       <=== Adabas load lib
//DDASSOR1 DD  DISP=SHR,DSN=EXAMPLE.DB%DBID.ASSOR1 <=== Adabas ASSO
//DDDRUCK  DD  SYSOUT=*
//DDPRINT  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//*  The following record will be replaced with a concatenation of
//*  sequential PLOG data sets
%SEQUENTIAL
//DDCARD   DD  *
ADARUN PROG=ADARPL,DBID=%DBID,SVC=svc,DEVICE=3390
/*
//DDKARTE  DD  *
ADARPL REPLAY
ADARPL LRPL=1500K
*    The following record will be replaced with ADARPL control
*    statements
```

```
%KARTE
/*
//
##                                                  <=== End of DDJCLIN
//*
```

4 Using the Subscription Logging Facility

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 those destinations. In order for this to occur, the SLOG facility must be activated for those destinations.

Once activated for a destination and when that destination becomes unavailable, the SLOG facility performs subscription logging by writing the replication data to a Replicator system file called the *SLOG system file* . This is an Adabas system file on the Event Replicator Server. When the destination is reopened and available, the SLOG facility sends the data it recorded back to the destination, while continuing to record new data to be sent to that destination. Once the destination logging catches up with the current data being replicated, normal replication processing resumes. This means that when all of the data that has been written to the SLOG system file for the destination has been successfully sent to the destination, Event Replicator Server will stop logging data to the SLOG system file for the destination and recommence sending it directly to the destination.

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.
4. The SLOG system file is also used for **synchronized replication replay**.

Setting Up Subscription Logging

Setting up subscription logging requires the following steps:

- Step 1. Calculate the maximum amount of space required.
- Step 2. Define the SLOG system file.
- Step 3. Activate subscription logging for the appropriate destinations.

- Step 4. Restart the Event Replicator Server.

Step 1. Calculate the maximum amount of space required.

The installation must make an estimate of the amount of space that will be required for the SLOG system file. The data that is written to the SLOG file are the URB* control blocks and associated data that is written to the messaging system for each transaction. Therefore, the most appropriate mechanism for determining how much data will be written is to monitor the amount of data written to a specific destination during normal processing.

To calculate the maximum amount of space, we recommend that you follow the calculation described in these substeps:

1 Determine the maximum amount of time that a destination may be unavailable. This may be related to maintenance windows, service level agreements, or other factors.

2 Determine the amount of data that is sent to the destination during its highest level of activity.

3 Based on the figures from these first substeps 1 and 2 (above), calculate the amount of data that will be written to a destination, assuming maximum outage time during the highest level of activity.

4 Add a 50% contingency and use the result to calculate the space required in the SLOG system file for the destination.

5 Repeat substeps 1 through 4 (above) for each destination where the SLOG facility will be activated. If multiple destinations are associated with the same subscription *and* if the destinations are only written to from that *one* subscription, only one destination should be included in the calculation as the SLOG facility can optimize the recording of data.

6 Based on the total space requirements for all destinations for which the SLOG facility will be activated, establish a space requirement for the SLOG system file and verify that the Event Replicator Server is large enough to contain it.

Step 2. Define the SLOG system file.

Using the space calculation from Step 1, use the sample job ADALODSL to define the SLOG system file to the Event Replicator Server. Ensure all blank parameters in the sample job are correctly set. Sample job ADALODSL is supplied in member ADALODSL of the MVSJOBS data set on z/OS, in member ADALODSL.X of the ARF*vrs* library on z/VSE, and ADALODSL(J) on BS2000.

Step 3. Activate subscription logging for the appropriate destinations.

Review the destination definitions for the destinations for which you want subscription logging activated. Change the value of the `DLOG` subparameter (in the `DESTINATION NAME` parameter) to "YES" (`DLOG=YES`) for all destinations for which you want subscription logging activated. The default for the `DLOG` parameter is "NO", so subscription logging will only be active on destinations where it has been explicitly requested.

You can also activate subscription logging in these destination definitions using the **Allow Logging** field using the Adabas Event Replicator Subsystem (see the *Adabas Event Replicator Subsystem User's Guide*).

Note: Consider specifying DAERROR=CLOSE for Adabas destinations defined with DLOG=YES. When an error is encountered and DAERROR=CLOSE, the Adabas transaction will be backed out, the Adabas destination will be closed, and the related replication data will be written to the SLOG system file.

Step 4. Restart the Event Replicator Server.

Restart the Event Replicator Server with the new parameter settings.

Starting Subscription Logging

Once subscription logging has been activated for a destination, it will automatically occur if:

- You close the destination using the ADADBS REPTOR function. For more information, read *ADADBS REPTOR Function* in *Event Replicator for Adabas Reference Guide*.
- An unrecoverable error occurs on the destination.
- The destination becomes full.

Stopping Subscription Logging

Once subscription logging is started for a destination, it will automatically stop when the destination becomes available again and all of the logged data is delogged from the SLOG system file to the destination.

Delogging SLOG File Data

Delogging is the process of reading the logged data for a destination in the SLOG system file and sending it to the destination. This will occur automatically for destinations for which subscription logging is activated and for which data has been logged if:

- You open the destination using the ADADBS REPTOR function. For more information, read *ADADBS REPTOR Function* in *Event Replicator for Adabas Reference Guide*.
- The destination becomes full, on the basis that a full condition is expected to clear itself.

Deleting SLOG System File Data

Data logged is deleted from the SLOG system file when:

- The data has been delogged and successfully sent to the destination.
- The destination is deactivated.
- The data is logged for a destination and the Event Replicator address space has been terminated. When the Event Replicator Server is subsequently restarted and if data exists in the SLOG system file for a destination that no longer exists or for which subscription logging has been deactivated, all SLOG system file data for that destination is deleted.

Note: In cases where multiple destinations with subscription logging activated are being logged to from a single subscription, the physical data in the SLOG system file will only be deleted when it has been sent to all destinations which experienced a failure. This is due to an optimization in the Event Replicator Server.

Maintaining the SLOG System File

The SLOG system file must be maintained in the same manner as any other system file. The SLOG system file may be deleted, reloaded, saved, restored, refreshed, or reordered. For more information about Adabas system files and utilities, refer to your Adabas Utilities documentation. For complete information about Adabas utility changes specific to the SLOG system file, read *Utilities Used with Replication* in *Event Replicator for Adabas Reference Guide*.

We recommend that you start the Event Replicator Server with the ADARUN `RPLPARMS` parameter set to "NONE" when maintenance on the SLOG system file is required. In particular, ADARUN RPLPARMS=NONE is required if you want to delete or refresh the SLOG file.

Error Conditions for the SLOG Facility

The only error condition expected for the SLOG facility is if the SLOG system file becomes full. When this happens, the destination with the largest number of items on the SLOG system file is identified and deactivated. Note that all SLOG system file data for this destination will be deleted, resulting in data loss for this destination.

In the event of any other error, the SLOG facility will deactivate the destination on which the failure occurred. This will result in all SLOG system file data for that destination being deleted, resulting in data loss for that destination.

5 Reducing the Risk of Replication Pool Overflows

To reduce the risk of a 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 Server replication pool, but then written to the SLOG system file, freeing up the space in the Adabas nucleus and Event Replicator Server replication pools. Such compressed input replication transactions written to the SLOG system file are called *database-related input transactions*.

To specify whether and when this happens, a `LOGINPUTTRANSACTION` (Log Input Transaction) parameter is provided. When logging of database-related input transactions is activated, every database-related input transaction from an Adabas nucleus will end up in the SLOG system file after the input phase. The `LOGINPUTTRANSACTION` parameter can be set as a threshold value so that this processing is only activated when a specified percentage of the Event Replicator Server replication pool space is used. When this threshold is exceeded, SLOG system file usage begins. If replication pool usage drops, the threshold is no longer exceeded, and all database-related input transactions have been processed, SLOG system file usage for database-related input transactions is turned off and normal processing resumes using only the replication pool. After being logged to the SLOG system file, the processing of database-related input transactions from the SLOG system file begins:

1. The transactions are returned to the replication pool, using a throttling mechanism so that only a limited amount of replication pool space is used at a time. Each transaction is queued to the assignment phase, at which point it is processed in the normal way.
2. Once each delogged transaction is fully processed by the assignment, subscription, and output phases, it is marked for deletion from the SLOG system file and the Event Replicator Server replication pool. It is deleted in chronological order (when all earlier database-related input transactions have been deleted, it is deleted).

Once processing of database-related input transactions from the SLOG system file has begun, all database-related input transactions are handled this way.

Note: The SLOG system file can also be used by the Event Replicator for subscription logging by destinations and for synchronized replay logging. The logging of database-related input transactions from the Adabas nucleus takes priority over these other uses of the SLOG system file.

You can also request that output replication transactions are written to the SLOG system file before the transactions are sent to a destination. Once an output replication transaction is written to the SLOG system file, it will be removed from the Event Replicator Server replication pool. To specify whether and when this happens, the `LOGOUTPUTTRANSACTION` (Log Output Transaction) parameter must be provided.

The output replication transactions are written to the specified destination from the SLOG. Each output replication transaction is queued to the output phase and is processed as normal. Once an output replication transaction is fully processed, it is marked for deletion from the SLOG system

file. Processed output replication transactions are deleted in a chronological order (when all earlier output replication transactions to the destination have been deleted, it is deleted).

Prerequisites

In order to use the `LOGINPUTTRANSACTION` parameter, the SLOG system file must be defined for your Event Replicator Server. For complete information on the SLOG system file, all of its uses, and suggestions for determining its size, please read ***Setting Up Subscription Logging***, elsewhere in this guide.

Error Processing

If an unexpected error is encountered writing or processing database-related input transactions on the SLOG system file, SLOG processing of database-related input transactions is suspended and all of the following actions are taken:

1. Replication is deactivated for the files with data on the SLOG system file.
2. Transactions held for writing to the SLOG system file are queued for the assignment phase for replication processing and processed in the normal manner.
3. The normal SLOG deletion routine will attempt to delete all of the database-related input transactions currently on the SLOG system file.
4. Processing of database-related input transactions on the SLOG system file is suspended until all database-related input transactions have been deleted.
5. No new database-related input transactions will be written to or held for writing to the SLOG system file nor will they be delogged from the SLOG system file.

When SLOG processing of database-related input files is suspended, an inactive file (with data in database-related input transactions on the SLOG system file) may not be activated until all database-related input transactions have been deleted from the SLOG system file.

6 Using 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.

Transaction logging is optional.

What Data Can Be Logged?

Two types of replication events can be logged to the TLOG file:

1. Processing events, which occur as a transaction is processed in each stage of replication processing, or when a user request for a transaction is sent through the input queues to the Event Replicator Server. Examples of user requests are requests for initial-state data and retransmission (resend buffer) requests. When a processing event occurs, an event-type record is written to the TLOG along with the data associated with the event.
2. State-change events, which occur when something happens within the Event Replicator Server that changes the status of the Event Replicator Server. For example, a state-change event occurs when a destination is closed. When a state-change event occurs, only an event-type record is written to the TLOG.

You can select what data is logged based on transaction logging parameters. These parameters can be stored as global, subscription-specific, or destination-specific parameters in the Replicator system file or they can be specified as initialization parameters, which are read from the DDKARTE statements of the Event Replicator Server startup job.

About the TLOG File

The TLOG file, when defined, resides in the current Adabas command log (CLOG). A new subcode is assigned to data specifically for the TLOG. This enables the TLOG to reuse the proven and reliable command logging infrastructure already in place for Adabas. It also means that you can reuse your existing command log operational procedures to handle TLOG data.

Each TLOG entry contains a URBL record and may be followed by other records, depending on the transaction event being logged. Each record written to the TLOG has only one control block per record; any other control blocks or data related to the same event are written as separate TLOG records. The control blocks must all fit in one CLOG record. There are two exceptions to this:

- The payload associated with the before and after images of a replicated record can span CLOG records. In this case, as much of the payload as possible is put into the first record following the URBD or input transaction information.

- Large input requests can span CLOG records.

In both cases, if the payload does not fit, additional payload-type records are written to ensure that all of the payload is available for printing on the TLOG.

Potential Space Problems

The TLOG records are written to the CLOG asynchronously with the processing of transactions by the Event Replicator Server. TLOG records are first written to the Event Replicator replication pool and then written from the pool to the CLOG. Therefore, activating transaction logging will have an impact on both your CLOG usage and your Event Replicator replication pool usage. For example, sometimes TLOG processing may not be able to write to the CLOG from the pool immediately (such as when CLOG rotation occurs). In this case, the TLOG records are stored in the Event Replicator replication pool until they can be logged to the CLOG. Such a situation might cause the Event Replicator replication pool to fill up.

The impact on CLOG usage depends on:

- The amount of TLOG data written. This is controlled by the transaction events that you have requested be logged and how long that logging has occurred.
- The amount of data flowing through the Event Replicator Server.

The impact on the Event Replicator replication pool usage depends on:

- The amount of TLOG data written. This is controlled by the transaction events that you have requested be logged.
- The amount of data flowing through the Event Replicator Server.

These factors should be considered when allocating the Event Replicator Server replication pool and the CLOG or when determining that their sizes should be increased to accommodate TLOG data collection.

If TLOG processing starts to use more than its defined limit of the Event Replicator replication pool (read about the `TLMAX` parameter), TLOG processing is suspended and a record is built at the end of the TLOG chain indicating that TLOG records were discarded. A counter is maintained of all the records that are lost. TLOG processing remains suspended until Event Replicator replication pool usage returns to below the level set by the `TLRESTART` parameter.

If the Event Replicator replication pool becomes full, the Event Replicator Server makes a decision about what resources should be freed up, based on the pool usage. If the TLOG is not using sufficient resources to merit freeing its resources, a replication pool overflow record (URBL) is written with a count of zero (0). If the TLOG is using too much of the replication pool, a decision to free up TLOG resources is made. In this case, a URBL record is written that includes the number of TLOG records that were freed to clear the Event Replicator replication pool overflow condition.

Setting Up Transaction Logging

To set up transaction logging:

1 Create and activate command logging for the Event Replicator address space via the CLOG. For complete information on doing this, refer to your other Adabas documentation.

2 Optionally, specify which transaction events you want logged, as well as the percentage of the Event Replicator replication pool that should be used (`TLMAX` parameter) and the replication pool usage level below which TLOG processing can continue when the pool fills up (`TLRESTART` parameter).

This step is not necessary as default values are supplied for all TLOG parameters. The default for all transaction events is "0" (no logging); the default for `TLMAX` is "50" percent, and the default for `TLRESTART` is "40" percent.

To specify which transaction events you want logged, use the transaction log parameters (read *Transaction Log (TLOG) Settings*), the subscription-specific log parameters (`STLFILTER`, `STLINPUT`, and `STLOUTPUT`), and the destination-specific log parameters (`DTLASSIGN`, `DTLCOMP`, `DTLSLOGREAD`, and `DTLSLOGWRITE`) described in *Event Replicator for Adabas Reference Guide*. These parameters can also be specified using fields in the Replicator system file using the Adabas Event Replicator Subsystem.

Starting Transaction Logging

Once you have set up the transaction logging parameters, stop and restart the Event Replicator Server to start transaction logging.

Dynamically Modifying Logging Settings

You can dynamically modify the transaction log levels and transaction parameters you have set using the TLOG operator command. For more information, read ***TLOG***, elsewhere in this manual.

Stopping Transaction Logging

To stop transaction logging, dynamically modify the transaction log levels of all transaction events to "0" (no logging). Read ***Dynamically Modifying Logging Settings***, previously in this manual.

Printing TLOG Records

To print the TLOG records from an Adabas command log file, use the ADARPP utility, the TLOG print utility. For more information, read *ADARPP Utility* in *Event Replicator for Adabas Reference Guide*.

7 Using the Event Replicator Subscription User Exit

The Event Replicator Server gives control to its subscription user exit after a record to be replicated has been processed according to the applicable subscription definitions. For each combination of subscription and file, the user exit to be called can be specified using the SFSEXIT parameter.

The same or different user exits can be specified for different subscriptions or files. If no user exit is specified, each replicated record is delivered according to the other parameters specified in the subscription, in particular the formats used for decompression.

The output parameters of the exit are preset such that if the exit does not modify them, Event Replicator Server processing is the same as if the user exit was not specified. A sample subscription record user exit, UEXREPT1, is provided in the source library that accompanies the Event Replicator for Adabas.

Caution: Sample user exits and programs are not supported under any maintenance contract agreement.

The subscription user exit can:

- Filter the selected record (suppressing its shipment to the target application), *or*
- Modify the record (for example, remove, insert, or transform data in the record).

Calling Sequence

The subscription user exit is called:

- During Event Replicator Server session start,
- For every replicated data record that matches a subscription and file combination for which the exit has been specified, *and*
- During normal Event Replicator Server session termination.

When called during Event Replicator Server session start, the exit may acquire resources that it needs throughout the session and perform other one-time operations. When called during Event Replicator Server session termination, the exit should release the resources acquired during the session.

When called for a data record that is being replicated, the exit is given information about the record and the surrounding transaction, including the following:

- The subscription name (URBTSNAM) and version indicator (URBTUSRV).
- The transaction sequence number within the subscription (URBTTSNR).
- The 28-byte communication ID of the user who performed the transaction (URBTGUID).
- The originating database ID (URBTDBID).

- The file number and ISN of the record (URBRFNR and URBRISN).
- The record sequence number within the transaction (URBRRSNR).
- The type of update operation (insert, update, delete) on the record (URBRTYP).
- If present, the before image of the record (URBDDATA/URBDLEND pointed to by URBPABI), decompressed according to the corresponding format defined in the subscription (SFBBI/SGFORMATBI or SFBKEY/SGFORMATKEY parameters).
- The type (data storage or primary key) of before image (URBDTYP), if a before image is present. The before image consists only of the primary key of the record if a primary key has been defined for the file and the key was updated or deleted.
- If present, the after image of the record (URBDDATA/URBDLEND pointed to by URBPAAI), decompressed according to the corresponding format defined in the subscription (SFBAI/SGFORMATAI parameter).

The user exit may do any of the following with the data record passed to it:

- Filter the record (i.e., suppress its shipment to the destination(s) defined in the subscription). This is done by setting URBXRETC to URBXRFLT.
- Modify the record contents in place. The before image of the record is located at URBD.URBDDATA off the address given in URBPABI. The after image is located at URBD.URBDDATA off the address given in URBPAAI.
- Make either record image shorter, by setting the new length in URBD.URBDLEND. The exit must not make a record image longer. To send longer data to the destination(s) than is extracted during decompression, extra space can be provided by adding appropriate format elements (e.g., '20X') to the formats used for decompression.
- Suppress the delivery of the before and/or after image while keeping the delivery of the control information for the record (URBR) intact. To suppress the delivery of either record image, set the associated record length in URBDLEND to zero.
- Set up to 8 bytes of error code information in the URBXERRC field. If error code information is set, no record data (URBD) will be delivered to the destination(s) for this record.

The exit should not set any response data in other fields of the URB* control blocks, as such data will be ignored and lost.

The call to the user exit is made using the standard mainframe subroutine linkage. At entry to the user exit, the general registers are set as follows:

- Register 1 contains the address of the URBP parameter address list.
- Register 13 contains the address of a standard 72-byte register save area.
- Register 14 contains the return address.
- Register 15 contains the user exit entry point address.

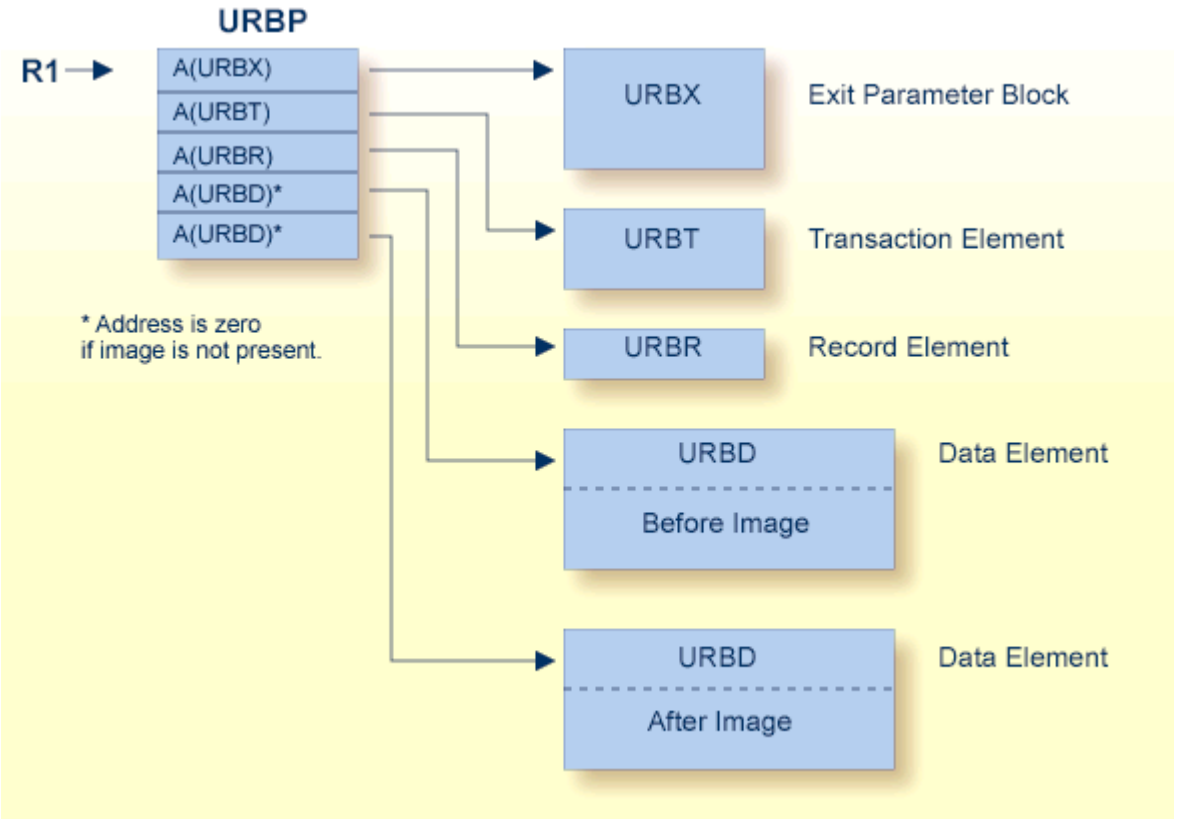
The exit is called in AMODE 31 and primary address space mode.

On exit, the contents of register 15 are immaterial. The exit sets fields in the URBX and data in the URBDs to give instructions to the Event Replicator Server regarding how to further process the input record.

The exit must restore registers 2-13 to their previous values prior to returning control to the Event Replicator Server. It must return control in the same PSW key, program state and mask, and address space control mode active on entry.

Input Parameters

The diagram below provides an overview of the subscription user exit input parameters.



URBP Parameter Address List

```
         MACRO
         URBP
URBP     DSECT
*------------------------------------------------------------------------*
*        URBP -- Subscription user exit parameter list                   *
*                                                                        *
*        DSECT URBP is used by the Event Replicator for Adabas.          *
*                                                                        *
*        DSECT URBP describes the parameter address list passed          *
*        to the Reptor subscription user exit.                           *
*------------------------------------------------------------------------*
*                                                                        *
*        During Reptor session start, the user exit is called once       *
*        with the Initialize function.                                   *
*                                                                        *
*        While the Reptor is running, the exit is called with the        *
*        Process-record function for every record that is being          *
*        replicated. The exit may filter the record (suppress its        *
*        shipment to the destination(s)) or modify its contents.         *
*                                                                        *
*        During Reptor shutdown (normal termination only), the exit      *
*        is called once with the Terminate function.                     *
*                                                                        *
*------------------------------------------------------------------------*
*                                                                        *
*        The subscription user exit is called using a                    *
*        standard BALR 14,15 interface, as follows:                      *
*                                                                        *
*        On entry,                                                       *
*           R1  -> URBP parameter address list                           *
*           R13 -> Standard 72-byte register save area                   *
*           R14 -> Return point                                          *
*           R15 -> User exit entry point                                 *
*                                                                        *
*        On exit,                                                        *
*           information may be returned in the first parameter           *
*                                                                        *
*------------------------------------------------------------------------*
URBPURBX DS    A             Addr of URBX parameter block
*
*  Only for the Process-record function (URBXFUNC=URBXFREC):
*
URBPURBT DS    A             Addr of URBT
URBPURBR DS    A             Addr of URBR
URBPABI  DS    A             Addr of URBD of before image
*                             Zero if no before image
URBPAAI  DS    A             Addr of URBD of after image
```

```
*                              Zero if no after image
*
*  The user exit may change the before/after image record contents
*  at URBD.URBDDATA and may decrease their lengths at URBD.URBDLEND.
*  However, it must not increase the record data lengths. If the
*  exit sets a length to zero, the corresponding record image will
*  not be shipped to the destination(s).
*
*  The last supplied address in the URBP parameter address list is
*  marked with the high bit set.
*
URBPL    EQU   *-URBP         DSECT length
         MEND
```

URBX Parameter Block

```
         MACRO
         URBX
URBX     DSECT
*-----------------------------------------------------------------------*
*        URBX -- Subscription user exit parameter block                 *
*                                                                       *
*        DSECT URBX is used by the Event Replicator for Adabas.         *
*                                                                       *
*        DSECT URBX describes the first parameter passed to the         *
*        Reptor subscription user exit.                                 *
*-----------------------------------------------------------------------*
URBXEYE  DS    CL4      (in)  URBX eye-catcher 'URBX'
URBXLEN  DS    F        (in)  Length of URBX
*
URBXVERS DS    CL2      (in)  Version indicator for exit parameters:
URBXVER1 EQU   C'01'           First version
*                             Future releases may allow a new layout
*                             of the subscription user exit parameters
*                             (URBP and URBX DSECTs). In this case
*                             URBXVERS will be set to a different
*                             value.
URBXVERH DS    CL2      (in)  Version indicator for URB* DSECTs,
*                              corresponding to URBHVERS
*
URBXUSER DS    A    (in/out)  Word for use by user exit:
*                              Initially (Initialize function), zero
*                              Then, value set by exit in previous call
*
URBXFUNC DS    X        (in)  Function code:
URBXFINI EQU   X'01'           Initialize
URBXFTRM EQU   X'02'           Terminate
URBXFREC EQU   X'03'           Process record
*
```

```
URBXRETC DS    X       (out)  Return code:
URBXRSHP EQU   X'00'           Ship record (default setting)
URBXRFLT EQU   X'01'           Filter record (other nonzero
*                               values cause filtering, too)
*
URBXJNAM DS    CL8      (in)  Reptor job name
*
URBXERRC DS    CL8     (out)  Other error code information:
*                              Passed to destination in URBR.URBRERRC
*                              (default setting is blanks)
*
         DS    XL60           Reserved area
         DS    0D
URBXL    EQU   *-URBX         DSECT length
         MEND
```

Controlling Delete Transaction Processing (SFREPLICATEDELETE=UPDATE Processing)

Using your subscription user exit and the SFILE SFREPLICATEDELETE=UPDATE parameter setting, you can control how physical delete transactions are handled on your target database.

If a subscription definition's SFILE includes the specification SFREPLICATEDELETE=UPDATE and an input record for the DBID/file is for a delete, the before and after images of the input record are passed to your subscription user exit. This allows your subscription user exit to determine the processing of replicated physical delete transactions on your target database. Your subscription user exit can decide if the physical delete transaction should be:

- physically deleted from your target database
- converted to an update
- ignored and not sent at all.

When a physical delete transaction is detected on the replicated database, an after image URBD is created along with a before image URBD and both are sent to the subscription user exit. The after image is only created if the SFREPLICATEDELETE=UPDATE setting is specified and if the before image is a copy of data storage (no key image is specified). Subscription definitions with SFREPLICATEDELETE=UPDATE should not have a primary key defined to the file. If a primary key is not defined, then the before image of the URBD will be a copy of data storage. If a key image is received when SFREPLICATEDELETE=UPDATE is specified, no after image is created and the subscription user exit can only request that the physical delete transaction be processed as a physical delete or ignored (it cannot be converted to an update).

Note: If SAGTARG is specified as the destination class of the subscription, you can still flag a field as being the primary key in the GFFT data. This is because filtering is done before

the subscription user exit is called, so the SAGTARG filtering will be performed on the physical delete transaction and not on an update transaction created from the delete.

When SFREPLICATEDELETE=UPDATE is specified, your subscription user exit must indicate how delete transactions should be handled:

- If you decide that you want physical delete transactions to be physically deleted from your target database, your subscription user exit must produce a return code indicating this. Event Replicator for Adabas will then release the after image URBDs and the delete transactions will be processed as normal physical deletes.
- If you decide that you want physical delete transactions to be converted to update transactions, your subscription user exit is responsible for changing fields in the after image URBD and returning a response code indicating that the deletes should be sent as updates.
- If you decide that you want physical delete transactions to be ignored (not sent), your subscription user exit must produce a return code indicating this. Event Replicator for Adabas will then release the after image URBDs and perform no further processing for the physical delete transaction.

8 Messaging System Integration

The messaging system (for example, webMethods EntireX or WebSphere MQ) plays a key role in the operation of the Event Replicator for Adabas; however, administration and operation of the messaging system is outside the scope of the Event Replicator for Adabas.

This chapter describes guidelines to integrating these messaging systems with the Event Replicator for Adabas.

Using WebSphere MQ as the Messaging System

Note: If you are running on z/OS using IBM WebSphere MQ Series definitions for your Event Replicator DESTINATION or IQUEUE definitions, a S0D3 abend can occur if you run it as a started task and specify the parameter REUSASID=YES. This is a documented IBM WebSphere MQ Series issue.

To integrate the Event Replicator for Adabas with the WebSphere MQ messaging system:

1 The Adabas Event Replicator Server delivered code must first be linked with the latest WebSphere MQ interface modules on site. Essentially, the delivered ADAMQT module must be linked with the CSQBSTUB module which is the WebSphere MQ interface module. JCL such as the following should be used to create the required Event Replicator Server load module called ADAMQS.

```
//LINK     EXEC PGM=IEWL,PARM='LIST,XREF,REUS,RENT'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(3,1))
//ADALIB   DD DISP=SHR,DSN="ADABAS Event Replicator Server Load Library"
//MQSLIB   DD DISP=SHR,DSN="WebSphere MQ SCSQLOAD Library"
//SYSLMOD  DD DISP=SHR,DSN="User Load Library"
//SYSLIN   DD *
 INCLUDE ADALIB(ADAMQT)
 INCLUDE MQSLIB(CSQBSTUB)
 ENTRY ADAMQS
 NAME  ADAMQS(R)
/*
```

2 For each destination to be used by the Event Replicator for Adabas the user needs to ensure that the `DMQQMGRNAME` and `DMQQNAME` parameters identify correctly defined resources within the WebSphere MQ installation. You must also ensure that the Event Replicator for Adabas job or started task is defined to the security system to have access to the WebSphere MQ resources identified in this step.

The `DMQQMGRNAME` parameter should identify the WebSphere MQ Subsystem to be used and `DMQQNAME` should identify a queue defined within that subsystem:

```
ADARPD DESTINATION NAME=[destname]
ADARPD DTYPE=MQSERIES
ADARPD DMQQMGRNAME=CSQ1
ADARPD DMQQNAME=SYSTEM.DEFAULT.LOCAL.QUEUE
ADARPD DMQDYNQNAME=SOME.MEANINGFUL.NAME
```

In the above example, the WebSphere MQ Subsystem name is CSQ1 and the queue to be used by the Event Replicator Server is SYSTEM.DEFAULT.LOCAL.QUEUE. If this actually defines a model queue, then `DMQDYNQNAME` should be specified to assign some name to the queue to be used by the Event Replicator for Adabas. Note that if the name contains special characters (such as periods), it should be enclosed in quotes as illustrated above. For complete information on the conventions that should be used when coding Event Replicator for Adabas parameters and subparameters, read *Conventions* in *Event Replicator for Adabas Reference Guide*.

3 The following table does not list all possible parameters used in defining local queues, merely those which may directly impact on the use of the queue by the Event Replicator for Adabas. For a complete guide to queue definition please consult your WebSphere MQ administrator or refer to the *MQSC Command Reference Manual*.

WebSphere MQ Attribute	Impact on Adabas Replication Services Operation
PUT	Set to ENABLED if you want the Event Replicator Server to use the queue as an output queue.
GET	Set to ENABLED if you want the Event Replicator Server to use the queue as an input queue.
MAXDEPTH	Set this integer value to the number of messages the queue needs to contain.
MAXMSGL	Set this integer value to the maximum expected message length to be handled by the queue.
DEFPSIST	The default persistence parameter determines whether messages on the queue will be retained over a failure or outage of the queue manager. The Event Replicator Server requires that this parameter be set to Yes, to ensure that messages are not lost.
SHARE | NOSHARE	If SHARE is set, more that one application may retrieve messages from the queue at a time. If NOSHARE is set only one application may retrieve messages from the queue at a time. Set according to the needs of your use of the Event Replicator Server.

For further discussion of these attributes please refer to your WebSphere MQ System Administrator or the *WebSphere MQ System Administration Guide*.

4 Consult with your WebSphere MQ System Administrator to ensure that the above resources are correctly defined and that any dynamic queue names used adhere to the naming conventions in force at your installation.

Using webMethods EntireX as the Messaging System

For many of the tasks described in this section, you will need to communicate with the person responsible for webMethods EntireX in your installation. The following gives details of what is required from an Event Replicator for Adabas perspective but for more details on any webMethods EntireX issues referred to here, please refer to the webMethods EntireX documentation. To enable the Event Replicator for Adabas to use webMethods EntireX as a messaging system, webMethods EntireX must be installed and available for use. This is the task of the person responsible for webMethods EntireX in your installation.

1. The Event Replicator for Adabas requires webMethods EntireX at a version as described in *Messaging System (webMethods EntireX or IBM WebSphere MQ) Requirements* in the *Installation* documentation.
2. In order for Event Replicator for Adabas to communicate with webMethods EntireX, the webMethods EntireX interface module must be available to the Event Replicator Server address space. When a webMethods EntireX queue of any nature is defined to Event Replicator for Adabas, the Event Replicator address space will attempt to load a module specified by the parameter `ETBBROKERNAME`. It is therefore necessary that you specify the webMethods EntireX load library in the STEPLIB concatenation in the Event Replicator Server JCL. For more information refer to your webMethods EntireX documentation.

   Note: If you elect to use the BROKER module under z/OS, review the security considerations in the EntireX Broker documentation for administration of Broker stubs under z/OS.

   If you will be using an open systems version of webMethods EntireX:

   - You must also have the z/OS Broker stub installed. This stub is included with Event Replicator for Adabas under the EXXxxx.MVSL0xx data set.
   - If you intend to use an open systems version of webMethods EntireX with security turned on, you must use ACI version 8 or later. To use an older version of the webMethods EntireX ACI, contact your Software AG support representative for assistance.

   Caution: In high throughput situations, we recommend that you use webMethods EntireX installed on z/OS.

   Use the following parameter to specify the name of the webMethods EntireX interface module to be used by the Event Replicator Server:

```
ADARPD ETBBROKERNAME=BROKER
```

This parameter is described in *ETBBROKERNAME (EntireX Broker Stub Program) Setting* in *Event Replicator for Adabas Reference Guide*. This setting can also be specified in the Replicator system file using the Adabas Event Replicator Subsystem.

3. For each webMethods EntireX destination to be used by the Event Replicator Server, the following destination parameters must be specified:

```
ADARPD DESTINATION NAME=[destname]
ADARPD  DTYPE=ETBROKER
ADARPD  DETBBROKERID= [BrokerID]
ADARPD  DETBSERVICECLASS=[classname]
ADARPD  DETBSERVICENAME=[servicename]
ADARPD  DETBSERVICE=[service] ↩
```

These parameters are described individually in *Event Replicator Initialization Parameters* in *Event Replicator for Adabas Reference Guide*. These settings can also be specified in the Replicator system file using the Adabas Event Replicator Subsystem.

4. For this destination to successfully be used with webMethods EntireX, the ETBBROKERID setting must correctly identify an webMethods EntireX Broker that is active in the system. This takes the standard format of an webMethods EntireX Broker ID, a number of examples of which follow:

Broker ID Example	Description
'10.128.200.202:18024:TCP'	The broker must be running on IP address 10.128.200.202 and listening on TCP/IP Port 18024.
myhost:18025:TCP	The host on which the broker is running must be defined such that a `gethostbyname('myhost')` request will return the IP address of that host. The broker itself must be running on that host and listening on port 18025.
ETB50500:SVC249:NET	The broker must be active as node 50500 on ADABAS Router SVC 249.

Once the Broker can be resolved, the appropriate attribute definitions must be in place for the service that the Event Replicator Server will use. The following sample service definition illustrates what is required in order for the previous sample definition to work correctly:

```
DEFAULTS=SERVICE
  CONV-LIMIT         = UNLIM
  CONV-NONACT        = 4M
  LONG-BUFFER-LIMIT  = UNLIM
  NOTIFY-EOC         = YES
  SERVER-NONACT      = 5M
  SHORT-BUFFER-LIMIT = UNLIM
  TRANSLATION        = NO
  MAX-UOWS           = 1000
  CLASS = [classname], SERVER = [servicename], SERVICE = [service]
```

In addition, the following table documents the various attributes and parameters that can impact the operation of Event Replicator Server when using webMethods EntireX as a messaging system. Software AG recommends that you review these parameters with the person in your installation responsible for webMethods EntireX.

ETB Broker Attribute	Impact on Adabas Replication Services Operation
AUTOLOGON	As the Event Replicator Server processing requires that messages are delivered as Units of Work, this must be set to 'NO'
CLIENT-NONACT	This should be set to as high a value as possible to prevent the ETB from timing out an Event Replicator Server due to lack of activity on a regular basis. The Event Replicator Server can recover from this time-out by logging on again, however, it could create an unacceptable overhead.
DEFERRED	If this is set to 'NO', it will be necessary for the ETB Consumer that will receive data from the Event Replicator Server to be active before the Event Replicator Server is started. If this is set to 'YES', the Event Replicator Server may start submitting data to the ETB Consumer before it is active, however, due to the amounts of data, this should be used with care in the event that ETB experiences a resource shortage before the ETB Consumer starts accepting the data.
MAX-MSG	This will determine how many SENDs will be required to output information on a given transaction to the broker. The Event Replicator Server logic will determine this during startup and use it to break up any messages that are longer than the maximum.
MAX-UOWS	This must be set to a value that is at least equal to the number of subtasks the Event Replicator Server is running plus 1 for the main task. The Event Replicator Server logic uses Units of work to output data to broker and could theoretically get to a point where one unit of work is active on each task. If there are other users of the same ETB in the system that use Units of Work, this figure should be reviewed based on their requirements.
NEW-UOW-MESSAGES	If this is set to 'NO', it will cause problems for the Event Replicator Server processing, as it will not be possible to set up new Units of Work.
NUM-CLIENT	This must be calculated as follows: Number of Event Replicator Server subtasks + 1 for main task + Number of ETB Input Queues. This is the requirement for the Event Replicator Server alone. If other systems are using the broker in question, this must be added to the requirements of the other systems.
PSTORE	If messages are intended to be persistent over ETB failure, this parameter must be set up along with the associated resources that are required for persistence. We strongly recommend that you use this feature.
SECURITY	This must be set to NO as the Event Replicator Server logic does not present user IDs that may be authenticated to the Broker.
UOW-MSGS	This must be set in association with the MAX-MSG parameters. The largest amount of data than can be replicated by the Event Replicator Server can be

ETB Broker Attribute	Impact on Adabas Replication Services Operation
	calculated by multiplying UOW-MSGS by MAX-MSG. Any transaction data above this length cannot be replicated.
UWTIME	If Units of Work are to be persistent, this parameter should be reviewed.
UWSTATP	The default 0 removes UOW status information once it has been received and committed by the consumer. Setting it to 1 or higher allows to keep this information for debugging purposes, but fills up Broker's permanent store fairly quickly.

Note: A sample program is provided, in source form, of a Natural-written target application that communicates with Event Replicator for Adabas using webMethods EntireX. You can use this Natural sample to build your own target application for Event Replicator for Adabas. The sample program is in the Event Replicator for Adabas SYSRPTR library and is named EXARFTA. It simply displays the data sent from the Event Replicator Server. You can run this program from Natural or edit it and follow the instructions in the program itself.

9 Creating a Sequential Output File

You can create a sequential output file of replicated transactions, if needed.

To create a sequential output file of replicated transactions, complete the following steps:

1 Define a File destination to the Event Replicator Server using either DDKARTE statements in the Event Replicator Server startup job or using the Adabas Event Replicator Subsystem. Using a File destination definition, replicated data can be written to the CLOG, using TLOG URBLTDOD records.

   For complete information on defining File destinations, read *DESTINATION Settings* and *DTYPE Parameter* in *Event Replicator for Adabas Reference Guide*. To define a File destination using the Adabas Event Replicator Subsystem, read *Adding a File Destination Definition*, in *Adabas Event Replicator Subsystem User's Guide*.

2 Verify that an appropriate Event Replicator Server subscription definition uses the File destination. For complete information on subscription definitions, read *SUBSCRIPTION Settings* in *Event Replicator for Adabas Reference Guide* or read *Maintaining Subscription Definitions*.

3 Start up the Event Replicator Server.

   As the Event Replicator Server runs, it will write replicated data to the File destination as TLOG URBLTDOD records.

4 When you are ready to create your sequential output file, issue the Adabas FEOFCL command to the Event Replicator Server. This will close the current dual or multiple command log (CLOG) and switch to another command log. This command is valid only if dual or multiple command logging is in effect.

5 Once the CLOG has been closed and switched, run the Adabas ADARES CLCOPY utility to copy the closed CLOG data set (with an earlier time stamp) to a new sequential data set.

   Once the CLCOPY function completes successfully, the copied CLOG data set is marked as empty. This function may, therefore, be used only once for any given CLOG data set.

6 Once the new sequential data set has been created, write a program to select TLOG records of URBLTYPE=URBLTDOD from it.

The output from this program, wherever you choose to store it, will contain the replicated transaction records.

You can optionally use:

- Utility ADARPE to extract records from the sequential CLOG file. For more information, read *ADARPE Utility: Extract TLOG Records*, in the *Event Replicator for Adabas Reference Guide*.
- Utility ADARPP to print some or all of the TLOG records on the sequential CLOG file. For more information, read *ADARPP Utility: Print TLOG Records*, in the *Event Replicator for Adabas Reference Guide*.

10 Replicating Security Definitions

Event Replicator for Adabas allows you to replicate modifications in the security definitions in an Adabas database. However, to do this, there are some requirements and restrictions of which you need to be aware.

Note: Adabas Security Facilities, including the Adabas security utility (ADASCR) can be obtained only by special request. If you are interested in Adabas Security Facilities, please contact your Software AG sales representative.

Restrictions

The following restrictions exist for security definition replication:

- Security definition replication only replicates security definition actions performed by the ADASCR utility. It does not replicate security definitions used by Adabas SAF Security.
- Any security file you identify in an SFILE definition cannot also be used in an initial-state definition. In other words, Event Replicator for Adabas does not support the use of security files in initial-state definitions. If you want to populate a target database with an initial snapshot of the security definitions, read ***Initial-State Processing of Security Definitions***, elsewhere in this section.

Requirements

To replicate security definitions, the following requirements must be met:

1. You must have a supported version of Adabas installed.
2. The following other SFILE parameters settings are required:
   - The `SFSECURITYFILE` parameter in the SFILE definition must be set to "YES".
   - A format buffer may *not* be specified in the SFILE definition. This means that no values may be specified for the `SFBAI`, `SFBBI`, `SFBKEY`, `SGFORMATAI`, `SGFORMATBI`, or `SGFORMATKEY` parameters.
   - No value may be specified for the `SFDEFAULTACODE` parameter.
   - A transaction filter definition may *not* be specified. This means that no value may be specified for the `SFFILTER` parameter.
   - The default value of *`YES`* must be specified for the `SFREPLICATEDELETE`, `SFREPLICATEINSERT`, `SFREPLICATENOTCHANGED`, and `SFREPLICATEUPDATE` parameters.
   - A subscription exit may *not* be specified. This means that no value may be specified for the `SFSEXIT` parameter.
   - When you specify SFSECURITYFILE=YES, you indicate that the file specified in the `SFILE` parameter is the file number of the security file for the source database identified by the

SFDBID parameter. Therefore, if SFSECURITYFILE=YES is specified for other subscription SFILE definitions using the same source database (with the same SFDBID setting), the same value must be set for each of the SFILE parameters in the SFILE definition. In other words, it is invalid for a source database (SFDBID setting) to have different file numbers specified for the security file in different subscriptions. For example, source database 10 cannot have the security file specified as both file 15 in one SFILE definition and 20 in another SFILE definition.

Finally, if SFSECURITYFILE=YES is specified for a source database (SFDBID parameter), then any Adabas destination definitions with same database specified as the destination input database (SFDBID setting equals the DAIDBID setting), must also specify identical file numbers for both the DAIFILE (input file) and DATFILE (target file) parameters.

3. Replication must be turned on for the security file in the source database. You can turn replication ON for the security file:
   - when you load the security file
   - using the ADADBS REPLICATION function
   - using Adabas Online System (AOS).

Initial-State Processing of Security Definitions

Any security file you identify in an SFILE definition cannot also be used in an initial-state definition. In other words, Event Replicator for Adabas does not support the use of security files in initial-state definitions.

To populate a target database with an initial snapshot of your security definitions, use one of the following methods:

- Save the source database and restore it on the target database.
- Once replication is turned on for the source security file, run ADASCR (Adabas security utility) jobs to set all of the security definitions. Replication will then replicate the security definitions to the target database.
- Run the ADASCR utility on the target database to initially set up the security definitions there.

11 Pertinent Operator Commands

Event Replicator for Adabas processes the same operator commands that may be given to an Adabas nucleus. For example, ADAEND will terminate the Event Replicator for Adabas. For more information about Adabas operator commands and how to enter them, read the Operations documentation for Adabas (in *Adabas Operations*).

If you choose to secure both an Event Replicator Server nucleus and the operator command using Adabas SAF Security, only use the first eight (8) characters (or less) of the command name in the respective ADAEOPTB table (macro). Putting the truncated versions of the commands (RPLREFRESH or RPLCLEAN, for example) into ADAEOPTB is sufficient for Adabas SAF Security to identify the commands and allow or disallow them to function as expected. For more information about the ADAEOPTB table, refer to your Adabas SAF Security documentation.

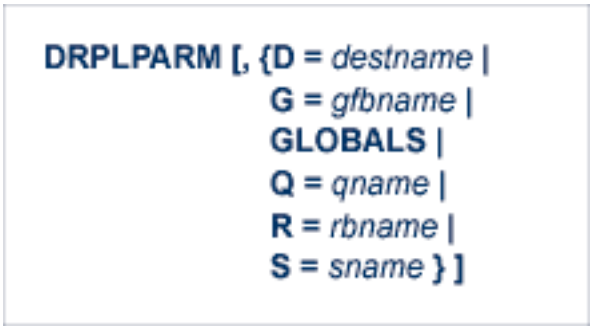
This chapter describes all of the operator commands pertinent only to the Event Replicator for Adabas. The following table categorizes the commands as they apply to the source Adabas nucleus, the Event Replicator Server nucleus, and the ADARPL utility. For detailed information about each command, click on the command name in the table.

Command Category	Applicable Commands
Source Adabas nucleus	DONLSTAT DRPLSTAT ONLRESUME ONLSTOP ONLSUSPEND RPLCONNECT RPLCONNECTCOUNT RPLCONNECTINTERVAL
Event Replicator Server nucleus	DRPLPARM DRPLSTAT RPLCHECK RPLCLEANUP RPLCONNECT RPLCONNECTCOUNT RPLCONNECTINTERVAL RPLREFRESH TLOG
ADARPL utility	DSTAT

DONLSTAT Command

Use the `DONLSTAT` command to display the status of each active reorder, invert online, or Event Replicator for Adabas initial-state process together with the process ID.

DRPLPARM Command



Use this command to display information about the replication definitions that are currently in use by an Event Replicator Server. This command can only be issued against an Event Replicator Server.

Each parameter of the DRPLPARM command is optional and is described in the following table. If you do not specify any of these parameters, information about all replication definitions in the Event Replicator Server is displayed.

Parameter	Displays information for:
D=*destname*	The specified destination definition (*destname*).
G=*gfbname*	The specified global format buffer definition (*gfbname*).
GLOBALS	All global value definitions only.
Q=*qname*	The specified input queue definition (*qname*).
R=*rbname*	The specified resend buffer definition (*rbname*).
S=*sname*	The specified subscription definition (*sname*).

DRPLSTAT Command

Use this command to display the replication-related statistics for an Adabas database (with replication turned on) or for an Event Replicator Server.

When issued against an Adabas database (with replication turned on), the statistics listed include:

- The total number of replication transactions completely processed.
- The current number of pending replicated transactions (transactions that have been committed, but not yet processed)
- The current number of incomplete transactions that will be replicated (but are not yet committed).

When issued against an Event Replicator Server, the statistics related to destinations, global values, and subscriptions in the database are listed. Replay Utility (ADARPL) statistics are also included.

The syntax for DRPLSTAT is:

DRPLSTAT [, {D = *destname* **| GLOBALS | S =** *sname* **| TOKENS}]**

The DRPLSTAT parameters are *always* optional and should be used only when the command is issued against an Event Replicator Server; the parameters are not valid when DRPLSTAT is issued for an Adabas database.

Note: Errors will occur if you attempt to run DRPLSTAT for an Adabas database using any of the parameters.

DRPLSTAT parameters are described in the following table. If you do not specify any of these parameters, replication-related statistics about all destinations, global values, and subscriptions in the Event Replicator Server are displayed.

Parameter	Displays replication-related statistics for:
D=*destname*	The specified destination (*destname*).
GLOBALS	All global values only.
S=*sname*	The specified subscription (*sname*).
TOKENS	ADARPL or ADALOD tokens. When a synchronized or replay-only request is submitted, a token is created in the Event Replicator Server nucleus. The TOKENS option of the DRPLSTAT facility allows you to view the details of the token, including the DBID, subscription, destination, start date, start time, end date, and end time in the token.

DSTAT Command

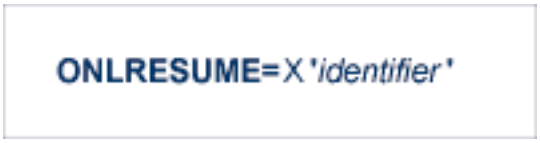
DSTAT

Use the DSTAT command to display statistics about the current Adabas nucleus status.

When this command is issued against a running Event Replicator ADARPL job, the Replay Utility (ADARPL) statistics are displayed.

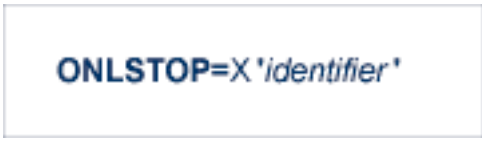
Note: After issuing a REFRESHSTATS, DSTAT displays the refreshed statistics.

ONLRESUME Command



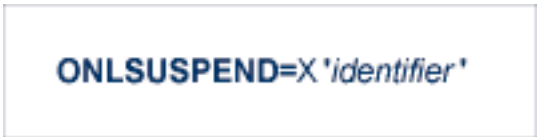
Use the `ONLRESUME` command to resume a previously suspended online reorder, invert, or Event Replicator for Adabas initial-state process.

ONLSTOP Command



Use the `ONLSTOP` command to stop an online reorder, invert, or Event Replicator for Adabas initial-state process cleanly. The process continues up to its next interrupt point in order to produce a consistent state, and then terminates after performing all necessary cleanup.

ONLSUSPEND Command



Use the `ONLSUSPEND` command to suspend an online reorder, invert, or Event Replicator for Adabas initial-state process. The process continues up to its next interrupt point in order to produce a consistent state, performs a command throwback, and enters a state where it cannot be selected for processing. This command is useful if the online process is consuming too much of the nucleus resources.

RPLCHECK Command

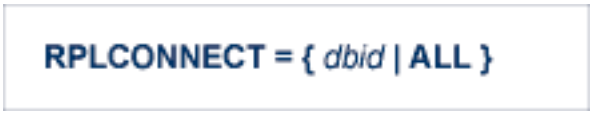
Use this command to perform the replication cross-check function for all active databases known (defined in one or more subscriptions) to the Event Replicator Server. When this command is run using the ADADBS OPERCOM function, the information about the cross-check function is printed to the ADADBS DDDRUCK data set. The information printed by ADADBS is the same as the information printed by the Event Replicator Server during the cross-check process initiated by the RPLCHECK operator command.

Note: This command can only be issued against an Event Replicator Server; it is not valid for the Adabas nucleus. If this command is issued against a database that is not an Event Replicator Server, error messages result.

RPLCLEANUP Command

Use this command if the Replay Utility (ADARPL) or an ADALOD job (with RPLLOAD=YES) abends and to stop replay processing (if it is running when the RPLCLEANUP command is entered). This command will clean up any open transactions in the Event Replicator Server that are associated with ADARPL or ADALOD (with RPLLOAD=YES) processing. Specify the token ID returned at startup for *tokenid* to perform this cleanup for a particular run; specify "ALL" to clean up transactions related to any run. Token IDs for running ADARPL processes or ADALOD jobs (with RPLLOAD=YES) can be listed using the **DRPLSTAT command**.

RPLCONNECT Command



Use this command to dynamically force a connection attempt to either a specific Event Replicator Server or Adabas database ID or to all related Event Replicator Server or Adabas database IDs.

One of the parameters of the RPLCONNECT command must be specified. There is no default. The parameters are described in the following table:

Parameter	Forces a reconnection attempt with:
ALL	All known Event Replicator Server or Adabas database IDs
dbid	The specified Event Replicator Server or Adabas database ID.

RPLCONNECTCOUNT Command

RPLCONNECTCOUNT = *nnn*

Use this command to dynamically specify the number of connection attempts made for the Adabas or Event Replicator Server nucleus after an attempt fails (response 148 is issued).

For *nnn*, specify a valid integer ranging from zero (0) through 2147483647. A value of zero indicates that no connection attempts should occur; a value of zero makes the most sense in situations where the Adabas database and the Event Replicator Server execute together on the same logical partition (LPAR). If the Adabas database and the Event Replicator Server execute on different LPARs, however, setting a real value using this command helps avoid errors that might arise if network problems occur because the network is not started or a network connection between the Adabas database and the Event Replicator Server is lost.

RPLCONNECTINTERVAL Command

RPLCONNECTINTERVAL = *nnn*

Use this command to dynamically specify the interval between connection attempts made for the Adabas or Event Replicator Server nucleus after an attempt fails (response 148 is issued).

For *nnn*, specify the number of seconds for the interval, ranging from zero (0) through 2147483647 seconds. A value of zero indicates that no connection attempts should occur; a value of zero makes the most sense in situations where the Adabas database and the Event Replicator Server execute together on the same logical partition (LPAR). If the Adabas database and the Event Replicator Server execute on different LPARs, however, setting a real value using this command helps avoid errors that might arise if network problems occur because the network is not started or a network connection between the Adabas database and the Event Replicator Server is lost.

RPLREFRESH Command

You can use the RPLREFRESH command to refresh resource definitions in your Event Replicator Server configuration while the Event Replicator Server is running or to abort scheduled (pending) RPLREFRESH processing. This command can be used to avoid the system downtime usually necessary to make configuration changes.

Two types of refresh processing can occur:

- A *partial refresh* occurs when a specific resource is refreshed. This is only supported if the Event Replicator Server resource definitions are stored in the Replicator system file and *not* if they are provided via DDKARTE statements in the Event Replicator Server startup job. In other words, ADARUN RPLPARMS=FILE must be specified or the `RPLPARMS` parameter must not be specified and the Replicator system file must be loaded on the Event Replicator Server.
- A complete refresh (using the ALL option) refreshes all Event Replicator Server resource definitions. It can be run whether the Event Replicator Server resource definitions are specified in either the Replicator system file or in DDKARTE statements in the Event Replicator Server startup job.

Note: On z/VSE systems, you cannot refresh resource definitions, if they are defined via an in-stream reader on SYSIPT.

You can refresh the configurations of the following resources:

- Subscription definitions, including the other definitions required by a subscription (such as global format definitions and transaction filter definitions).
- Destination definitions
- Resend buffer definitions

Note: Special processing is necessary if you want to modify a resend buffer; you cannot use the RPLREFRESH command to dynamically modify a resend buffer. For information on modifying a resend buffer, read ***Resend Buffer Modification and Refresh Requirements***, later in this section.

- Initial-state definitions
- Input queue definitions
- Subscription user exit

Note: Subscription user exits can be only refreshed with SX=*name* parameter, one subscription user exit at a time. Subscription user exits cannot be refreshed by `RPLREFRESH,ALL` command. For information on refreshing a subscription user exit, read ***Refreshing Subscription User Exit***, later in this section.

- Global definitions.

Note: The SUBTASKS, NPADACALLS, ETBBROKERNAME, and MAXOUTPUTSIZE initialization parameter settings *cannot* be changed using RPLREFRESH.

This section covers the following topics:

- Command Syntax
- Refresh Processing

Command Syntax

RPLREFRESH can be used to initiate refresh processing or to abort previously requested RPLREFRESH processing. The syntax of the command differs, depending on which function you are trying to perform:

- Initiating a Refresh Attempt for a Single Resource
- Initiating a Refresh Attempt for Global Resources
- Initiating a Refresh Attempt for All Resources
- Aborting a Refresh attempt
- Refreshing Subscription User Exit

Initiating a Refresh Attempt for a Single Resource

The RPLREFRESH syntax to initiate a refresh attempt of a single resource is:

```
RPLREFRESH,{D | I | Q | R | S | SX}=name
```

Specify "D" to refresh a destination resource, "I" to refresh an initial-state resource, "Q" to refresh an input queue resource, "R" to refresh a resend buffer resource, "S" to refresh a subscription resource, or “SX” to refresh a subscription user exit. Specify the name of the resource for *name*.

Initiating a Refresh Attempt for Global Resources

The RPLREFRESH syntax to initiate a refresh attempt for the global value parameters is:

```
RPLREFRESH,GL[obals]
```

For a complete description of the global values available to the Event Replicator Server, read *Setting Global Values using the Adabas Event Replicator Subsystem* in *Adabas Event Replicator Subsystem User's Guide*.

Initiating a Refresh Attempt for All Resources

The RPLREFRESH syntax to initiate a refresh attempt for all Event Replicator Server resources is:

```
RPLREFRESH,ALL
```

When ALL is specified, the RPLREFRESH processing logic will compare the current configuration with the configuration specified in the Replicator system file or in the DDKARTE statements in the Event Replicator Server startup job. Each resource that has been changed will be refreshed by the system.

Refreshing all resources is implemented by making a series of RPLREFRESH passes through the resources in an hierarchical sequence to ensure that resources without dependencies are refreshed immediately, followed by lower-level resources associated with the refreshed resources and the resources that depend on them. All additions and modifications occur as part of the primary loop processing with resource deletion occurring when all additions and modifications have completed or failed.

RPLREFRESH,ALL processing will continually retry refresh attempts until all resources have been refreshed, the refresh attempt fails, or the time limit to wait for resources to be quiesced is exceeded.

RPLREFRESH,ALL Additions and Modifications

The following hierarchical sequence for refresh additions and modifications is used:

1. Global value resource definitions are refreshed first. This is only done once, the first time through, because it cannot fail and does not currently require a quiesce.
2. Input queue definitions are refreshed, assuming they are closed. No other Event Replicator Server definitions have dependencies on input queue definitions.
3. Resend buffer definitions are refreshed third as subscription definitions and, indirectly, initial-state definitions have dependencies on them.
4. Destinations are refreshed fourth as subscription and initial-state definitions have dependencies on them.
5. Subscriptions are refreshed fifth as initial-state definitions have dependencies on them.
6. Initial-state definitions are refreshed last because they have dependencies on many other Event Replicator Server definitions.

When an error occurs during the addition or modification of a resource in this hierarchical processing, the refresh of that resource and the refresh of any other resources with dependencies on it are removed from the RPLREFRESH,ALL attempt.

If a resource cannot be modified because it must be quiesced first, the resource and any other resources with dependencies on it are left until the next pass of the RPLREFRESH, ALL processing. During the next pass, an attempt is made to modify the resource again.

RPLREFRESH,ALL Deletions

Once all modifications and additions have completed or failed, each of the resources that are to be deleted during RPLREFRESH processing are deleted in the following sequence:

1. Initial-state definitions are deleted first, because no other Event Replicator Server definitions have dependencies on them.
2. Subscription definitions are deleted second because any initial-state definitions with dependencies on them should have already been deleted.
3. Destination definitions are deleted third because any subscription and initial-state definitions with dependencies on them should have already been deleted.
4. Resend buffer definitions are deleted fourth because any subscription and initial-state definitions with dependencies on them should have already been deleted.
5. Input queue definitions are deleted last.

When an error occurs during the deletion of a resource in this hierarchical processing, the refresh of that resource and the refresh of any other resources with dependencies on it are removed from the RPLREFRESH,ALL attempt.

If a resource cannot be deleted because it must be quiesced first, the resource and any other resources with dependencies on it are left until the next pass of the RPLREFRESH, ALL processing. During the next pass, an attempt is made to delete the resource again.

Aborting a Refresh attempt

The RPLREFRESH syntax to abort a scheduled (delayed) refresh attempt is:

```
RPLREFRESH,AB[ort]
```

Note: Refresh of a subscription user exit cannot be aborted.

For information about what scheduled, or pending, refresh processing is, read *Refresh Processing*, next in this guide.

Refreshing Subscription User Exit

```
RPLREFRESH,SX=subscription_user_exit_name
```

Only one subscription user exit will be refreshed at a time and its name stays the same.

This new parameter is supported only for z/OS operating system, ADABAS version 8.4 or higher and running APF-authorized.

EXITLIB DD must be specified for the Reptor job.

The specified exit is loaded from the EXITLIB module with the specified name, both during Reptor start and during an RPLREFRESH operation. To change a loaded exit, the new version of the exit must be placed under the same name in the EXITLIB.

The new version of the exit becomes effective immediately for **all** subscriptions for which it has been specified.

Once the new version of the exit has been successfully loaded, a Terminate function for the old version of the exit will be called and the old version will be is deleted.

For the new version of the exit the *Initialize* function will be called.

If the new version of the exit cannot be loaded, the old version stays in effect (and its *Terminate* function is not called).

Refreshing an exit with RPLREFRESH,SX=*exit_name* is different from refreshing an exit as part of an RPLREFRESH,S=*subscription_name* operation in that

- Refreshing a subscription loads only an exit with a new name. If the name is not changed, no new exit is loaded and the old exit stays in effect.
- Refreshing a subscription may change the exit only for this subscription. Refreshing an exit changes the exit for all subscriptions for which it has been specified.

Refresh Processing

Refresh processing occurs in the following sequence:

1. The parameters for the resource are read from the Replicator system file and validated.
2. If this is a request to refresh a subscription definition, the global format buffer definitions and filter definitions are also read and validated.

   If the global format buffer validation parameter (`FBVALIDATION`) is set to something other than "NONE", the validation behaves in the following fashion during a refresh:

   - If `FBVALIDATION` is set to "ABORT", when a format buffer error is encountered during refresh processing, the refresh request fails.
   - If `FBVALIDATION` is set to "DEAC", when a format buffer error is encountered during refresh processing, the refresh request fails.
   - If `FBVALIDATION` is set to "WARN", when a format buffer error is encountered during refresh processing, a warning message is issued and refresh processing continues.
3. If this is an attempt to refresh an initial-state definition, the definition is automatically deactivated (if it is not already inactive) or it is rescheduled.
4. The system determines what kind of a refresh request is being submitted, based on the existence of the resource definition in the running Replicator system file.

- If the resource definition exists in the running configuration as well as the refresh request configuration, the request is to modify the resource.
- If the resource definition exists in the running configuration, but *not* in the refresh request configuration, the request is to delete the resource.
- If the resource definition does *not* exist in the running configuration, but does exist in the refresh request configuration, the request is to add the resource.

5. The system is checked to determine if the refresh request can be performed at all. Appropriate messages are issued if the refresh request cannot be performed.
6. The system is checked to determine if it can perform the refresh immediately or if the refresh must be delayed. A refresh can only occur immediately when the resources in the system are in a state acceptable to Event Replicator for Adabas for a refresh. The refresh state requirements vary, depending on the resource definition involved and depending on the change for that resource. For further information, read the following sections, as appropriate:
   - ***Destination Refresh Requirements***
   - ***Global Value Refresh Requirements***
   - ***Initial-State Refresh Requirements***
   - ***Input Queue Requirements***
   - ***Resend Buffer Modification and Refresh Requirements***
   - ***Subscription Refresh Requirements***

   If these conditions are not met, the refresh request is delayed for five minutes.

   Note: It is during this wait (delay) time that an `RPLREFRESH,ABORT` request can be issued for a refresh request.

   If, after five minutes, the conditions are still not met, the refresh request will fail. Otherwise, the refresh occurs when the conditions for the refresh are right.

Messages are issued following an RPLREFRESH attempt, indicating whether the refresh was successful or not. If the attempt was unsuccessful, additional messages will identify the reason why it failed.

- Destination Refresh Requirements
- Global Value Refresh Requirements
- Initial-State Refresh Requirements
- Input Queue Requirements
- Resend Buffer Modification and Refresh Requirements

- Subscription Refresh Requirements

Destination Refresh Requirements

A refresh for a destination definition can only occur immediately if the definition is not in use; otherwise, the refresh is delayed.

The following table describes each parameter that can be provided to a destination and the refresh state requirements for the parameter. When more than one parameter is changed, the requirements are cumulative so that all requirements must be met before refresh processing can occur.

Parameter	Requirement
DACTIVE	None; this parameter can be refreshed at any time.
DAIDBID	This parameter can be refreshed only if the Adabas destination is not actively producing data.
DAIFILE	This parameter can be refreshed only if the Adabas destination is not actively producing data.
DARC	This parameter can be refreshed only if the destination is not actively outputting data.
DATDBID	This parameter can be refreshed only if the Adabas destination is not actively producing data.
DATFILE	This parameter can be refreshed only after the destination is closed.
DCOMMITTHRESHHOLD	None; this parameter can be refreshed at any time.
DETBBROKERID	This parameter can be refreshed only after the destination is closed.
DETBSERVICE	This parameter can be refreshed only after the destination is closed.
DETBSERVICECLASS	This parameter can be refreshed only after the destination is closed.
DETBSERVICENAME	This parameter can be refreshed only after the destination is closed.
DEXIT	This parameter can be refreshed only if the destination is not actively producing data.
DEXITWORKSIZE	This parameter can be refreshed only if the destination is not actively producing data.
DLOG	This parameter setting can be changed from "NO" to "YES" at any time. However, it can only be changed from "YES" to "NO" if there is no data on the SLOG file and no data is available to be written to the SLOG for the destination.
DMQDYNQNAME	This parameter can be refreshed only after the destination is closed.
DMQQMGRNAME	This parameter can be refreshed only after the destination is closed.
DMQQNAME	This parameter can be refreshed only after the destination is closed.
DQFULLDELAY	None; this parameter can be refreshed at any time.
DTLASSIGN	None; this parameter can be refreshed at any time.
DTLCOMP	None; this parameter can be refreshed at any time.
DTLSLOGREAD	None; this parameter can be refreshed at any time.

Parameter	Requirement
DTLSLOGWRITE	None; this parameter can be refreshed at any time.
DTYPE	This parameter can be refreshed only after the destination is closed.

Global Value Refresh Requirements

The following table describes the refresh requirements for some of the global value parameters that can be specified for the Event Replicator Server. When more than one parameter is changed, the requirements are cumulative so that all requirements must be met before refresh processing can occur.

Parameter	Requirement
IRMSGINTERVAL	None; this parameter can be refreshed at any time, without need to quiesce any part of the Event Replicator Server.
IRMSGLIMIT	None; this parameter can be refreshed at any time, without need to quiesce any part of the Event Replicator Server.
MAXOUTPUTSIZE	This parameter cannot be changed with RPLREFRESH.
NPADACALLS	This parameter cannot be changed with RPLREFRESH.
SUBTASKS	This parameter cannot be changed with RPLREFRESH.
TLOG Parameters	None; these parameters can be changed at any time, without need to quiesce any part of the Event Replicator Server.
VERIFYMODE	This parameter cannot be changed with RPLREFRESH.

Initial-State Refresh Requirements

A refresh for an initial-state definition can only occur immediately if the definition is not in use; otherwise, the refresh is delayed.

The following table describes each parameter that can be provided to an initial-state definition and the refresh state requirements for the parameter. When more than one parameter is changed, the requirements are cumulative so that all requirements must be met before refresh processing can occur.

Parameter	Requirement	Operations
IDESTINATION	This parameter can be refreshed only if no instances of the initial-state request are active.	Modify
IFILE	This parameter can be refreshed only if no instances of the initial-state request are active.	Add
		Delete
IDBID	This parameter can be refreshed only if no instances of the initial-state request are active.	Add
		Delete

Parameter	Requirement	Operations
IMAXREQ	This parameter can be increased at any time, without need to quiesce any part of the Event Replicator Server. However, if this parameter needs to decreased, no instances of the initial-state request can be active.	Modify
INITIALSTATE	This definition can be refreshed only if no instances of it are active.	Add
		Delete
ISNLIST	This parameter can be refreshed only if no instances of the initial-state request are active.	Modify
ISUBSCRIPTION	This parameter can be refreshed only if no instances of the initial-state request are active.	Modify
SELCRIT	This parameter can be refreshed only if no instances of the initial-state request are active.	Modify

When adding an initial-state definition during RPLREFRESH processing, all of the destinations and subscriptions referenced by the initial-state definition must already be defined to the Event Replicator Server. Also, you can only delete an initial-state definition during RPLREFRESH processing if there are no active requests in progress for the initial-state definition.

If any of the following conditions is true, the initial-state definition refresh can only be scheduled.

- If changes have been made to the destination, subscription, or file definitions referenced by the initial-state definition, there may be no active requests in progress for the initial-state definition.
- If the IMAXREQ parameter value has been decreased, there may be no active requests in progress for the initial-state definition.

Input Queue Requirements

The input queue definition must be closed if you are modifying or deleting the definition using RPLREFRESH. Otherwise, the refresh request will fail.

Resend Buffer Modification and Refresh Requirements

A resend buffer definition cannot be dynamically modified and refreshed using RPLREFRESH. You can only add or delete resend buffer definitions using RPLREFRESH.

Note: If you want to delete a resend buffer definition during an RPLREFRESH run, the resend buffer definition may not be referenced by any subscription.

If you want to modify a resend buffer and refresh its definition, you must follow complete these steps:

1. Delete the resend buffer definition. This may require removing references to it from one or more subscriptions.
2. Recreate the resend buffer definition, with the required changes.

3. Restore the references to the resend buffer definition in the appropriate subscriptions.
4. Use RPLREFRESH to refresh the resend buffer definition.
5. Use RPLREFRESH to refresh the subscriptions that reference the resend buffer definition.

Subscription Refresh Requirements

A refresh for a subscription definition can only occur immediately if the definition is not in use; otherwise, the refresh is delayed.

The following table describes each parameter that can be provided to a subscription and the refresh requirements for the parameter. When more than one parameter is changed, the requirements are cumulative so that all requirements must be met before refresh processing can occur. If the requirements listed in this table are not met, the refresh request is delayed (scheduled) for a later time.

The *Operations* column indicates whether the changes to the parameter affect the addition, deletion, or modification of a subscription definition.

Parameter	Requirement	Operations
SACODE	None; this parameter can be refreshed at any time.	Modify
SACTIVE	None; this parameter can be refreshed at any time.	Modify
SARC	None; this parameter can be refreshed at any time.	Modify
SDESTINATION	If an attempt is being made to add a destination to the subscription, the refresh attempt can occur at any time. If an attempt is being made to delete a destination from the subscription, the refresh attempt can occur only when all transactions for the subscription to that destination are processed and if all SLOG data for the destination is also processed.	Add, Delete
SFBAI	This parameter can be refreshed only when all transactions for the subscription have been processed and if all the SLOG data for any destination to which the subscription sends data has also been processed.	Modify
SFBBI	This parameter can be refreshed only when all transactions for the subscription have been processed and if all the SLOG data for any destination to which the subscription sends data has also been processed.	Modify
SFBKEY	This parameter can be refreshed only when all transactions for the subscription have been processed and if all the SLOG data for any destination to which the subscription sends data has also been processed.	Modify
SFDBID	If an attempt is being made to add a subscription definition, it is delayed when input transactions are queued for assignment to any subscription. In this case, the refresh attempt is scheduled for later, when all input transactions have processed.	Add, delete

Parameter	Requirement	Operations
	If an attempt is being made to delete a subscription definition during a refresh, it can occur only when: ■ All transactions have completed output processing for the subscription. ■ All SLOG data has been processed for all destinations to which the subscription sends data. ■ No initial-state requests are ongoing for the subscription.	
SFFILTER	None; this parameter can be refreshed at any time.	Modify
SFILE	If an attempt is being made to add a subscription definition, it is delayed when input transactions are queued for assignment to any subscription. In this case, the refresh attempt is scheduled for later, when all input transactions have processed. If an attempt is being made to delete a subscription definition during a refresh, it can occur only when: ■ All transactions have completed output processing for the subscription. ■ All SLOG data has been processed for all destinations to which the subscription sends data. ■ No initial-state requests are ongoing for the subscription.	Add, delete
SFREPLICATEDELETE	None; this parameter can be refreshed at any time.	Modify
SFREPLICATEINSERT	None; this parameter can be refreshed at any time.	Modify
SFREPLICATENOTCHANGED	None; this parameter can be refreshed at any time.	Modify
SFREPLICATEUPDATE	None; this parameter can be refreshed at any time.	Modify
SFSEXIT	None; this parameter can be refreshed at any time.	Modify
SGFORMATAI	This parameter can be refreshed only when all transactions for the subscription have been processed and if all the SLOG data for any destination to which the subscription sends data has also been processed.	Modify
SGFORMATBI	This parameter can be refreshed only when all transactions for the subscription have been processed and if all the SLOG data for any destination to which the subscription sends data has also been processed.	Modify
SGFORMATKEY	This parameter can be refreshed only when all transactions for the subscription have been processed and if all the SLOG data for any destination to which the subscription sends data has also been processed.	Modify

Parameter	Requirement	Operations
SIDESTINATION	If an attempt is being made to add a destination to the subscription, the refresh attempt can occur at any time. If an attempt is being made to delete a destination from the subscription, the refresh attempt can occur only when all transactions for the subscription to that destination are completed and if all SLOG data for the destination is also processed.	Add, Delete
SINCREMENTIS	None; this parameter can be refreshed at any time.	Modify
SRESENDBUFFER	None; this parameter can be refreshed at any time.	Modify
STLFILTER	None; this parameter can be refreshed at any time.	Modify
STLINPUT	None; this parameter can be refreshed at any time.	Modify
STLOUTPUT	None; this parameter can be refreshed at any time.	Modify
SUBSCRIPTION	If an attempt is being made to add a subscription definition, it is delayed when input transactions are queued for assignment to any subscription. In this case, the refresh attempt is scheduled for later, when all input transactions have processed. If an attempt is being made to delete a subscription definition during a refresh, it can occur only when: ■ All transactions have completed output processing for the subscription. ■ All SLOG data has been processed for all destinations to which the subscription sends data. ■ No initial-state requests are ongoing for the subscription.	Add, delete
SVERSION	None; this parameter can be refreshed at any time.	Modify
SWCODE	None; this parameter can be refreshed at any time.	Modify

TLOG Command

Use this command to dynamically alter the level of transaction logging occurring for replication. The complete format of this command is:

```
TLOG[,[S=subscription|D=destination]]event-name=level
```

where *subscription* is an optional subscription definition name, *destination* is an optional destination definition name, *event-name* is the valid name of a TLOG parameter, as defined in the following table, and *level* is a valid transaction logging level as defined for each parameter.

The following table lists the valid TLOG parameters, their ranges, their defaults, and whether or not the S or D options need to be specified with a TLOG operator command for the parameter. Complete descriptions of each parameter can be found by clicking on the link in the table reading

about the appropriate parameter in *Transaction Log (TLOG) Settings* in *Event Replicator for Adabas Reference Guide*.

Event-Name (Parameter)	Value Range	Default	S or D Option Required for Command?
DDTLASSIGN	0-3	0	D required
DTLCOMP	0-3	0	D required
DTLSLOGREAD	0-3	0	D required
DTSLOGWRITE	0-3	0	D required
STLFILTER	0-3	0	S required
STLINPUT	0-3	0	S required
STLOUTPUT	0-3	0	S required
TLCOMP	0-1	0	No
TLINPUT	0-3	0	No
TLISTATE	0-3	0	No
TLISTATECOMP	0-3	0	No
TLNOSUB	0-3	0	No
TLQCOMP	0-3	0	No
TLREQERR	0-3	0	No
TLREQRECV	0-2	0	No
TLREQREJECT	0-2	0	No
TLRETRANS	0-3	0	No
TLSTATUS	0-3	0	No

12 AOS Features Supporting Event Replicator for Adabas

Adabas Online System (AOS) includes several other features that support Event Replicator for Adabas. Presentation of AOS is different depending on whether the database is an Event Replicator Server (Replication) database or a standard Adabas database and depending on whether or not replication is turned on for an Adabas database. An Event Replicator Server is one for which the REPTOR parameter of ADADEF DEFINE is set to YES (REPTOR=YES).

Notes:

1. Any AOS screen not described in this section is unchanged for the Event Replicator for Adabas and remains the same as presented in the current version of the AOS documentation.
2. AOS is not able to modify file parameters for Adabas files incorporated in Predict. This is because AOS cannot modify the FCB of these files. Error messages are produced when such an attempt is made. We recommend that you use a supported version of Predict to make file parameter updates for Adabas files incorporated in Predict.

Screen Differences

Main Menu Differences

The only change that appears on the AOS Main Menu when an Event Replicator Server is selected, is the addition of the word "Replicator" on the upper left side of the screen. If a standard Adabas database is selected, no word is shown. Here is the Main Menu for a standard Adabas database.

```
16:24:57          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
                               -  Main Menu  -                      PMAIN02


             Code  Basic Services              Code  Other Services
             ----  ----------------------      ----  ---------------------
              A    Session monitoring           1    Adabas Cache Facility
              C    Checkpoint maintenance       2    Delta Save Facility
              F    File maintenance             3    Trigger Maintenance
              M    Database maintenance         4    AOS Security
              O    Session opercoms             5    Transaction Manager
              R    Database report              6    Adabas Statistics
              S    Space calculation            7    Vista
              ?    Help                         8    Fastpath
              .    Exit                         9    SAF Security
             ----  ----------------------      ----  ---------------------

 Code ....... _
 Database ... 1955    (WIS1955)
MENU04 : - Session Monitoring -  Function interrupted
Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                                                ↩
```

Here is the Main Menu for an Event Replicator Server:

```
16:24:28          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator                     -  Main Menu  -                      PMAIN02


             Code  Basic Services              Code  Other Services
             ----  ----------------------      ----  ---------------------
              A    Session monitoring           1    Adabas Cache Facility
              C    Checkpoint maintenance       2    Delta Save Facility
              F    File maintenance             3    Trigger Maintenance
              M    Database maintenance         4    AOS Security
              O    Session opercoms             5    Transaction Manager
              R    Database report              6    Adabas Statistics
              S    Space calculation            7    Vista
              ?    Help                         8    Fastpath
              .    Exit                         9    SAF Security
             ----  ----------------------      ----  ---------------------

 Code ....... _
 Database ... 1234    (WIS1234)
MENU04 : - Session Monitoring -  Function interrupted
Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                                                ↩
```

Display Parameter Screen Differences

The Display Parameters screen consists of a series of screens that display general database parameters. The final screen displays parameters related to replication.

If a standard Adabas database is selected and replication is turned on for the database, the last screen of the Display Parameters series of screens displays parameters information related the replication of the Adabas database.

```
15:30:30          ***** A D A B A S  BASIC  SERVICES *****          2013-04-03
DBID 1955                  -  Display Parameters  -                 PACPD22


---- Replication Parameters -------
Replication .......... YES
RPWARNPercent ........ 0
RPWARNINCrement ...... 10
RPWARNINTerval ....... 60
RPWARNMessagelimit ... 5
RPCONNECTCount ....... 0
RPCONNECTInterval .... 0
RPLSORT .............. YES
LRPL ................. 100000









                                                                Page 6 of 6
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                         -                Menu       ↩
```

If an Event Replicator Server is selected, the last screen of the Display Parameters series of screens displays parameter information for the Event Replicator Server database used during replication.

```
15:31:58          ***** A D A B A S  BASIC  SERVICES *****          2013-04-03
DBID 1234                  -  Display Parameters  -                 PACPD22


------- Reptor Parameters -------
RPLParms ............. File
RPWARNPercent ........ 0
RPWARNINCrement ...... 10
RPWARNINTerval ....... 60
RPWARNMessagelimit ... 5
RPCONNECTCount ....... 0
RPCONNECTInterval .... 0
LRPL ................. 25000000









                                                                Page 6 of 6
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                        -                Menu       ↩
```

Display General DB-Layout Screen Differences

The Display General DB-Layout screen consists of a series of screens that display general database information. These screens vary slightly, depending on whether an Event Replicator Server or a standard Adabas database is selected.

If an Event Replicator Server is selected, the first screen of the Display General DB-Layout series of screens indicates whether this is an Event Replicator Server. This information appears in the **Replicator Facility** field on the first screen:

```
15:38:45          ***** A D A B A S  BASIC  SERVICES *****       2013-04-03
DBID 1234                - Display General DB-Layout -              PDRG012


 Database Name ............ WIS1234
 Database Number .......... 1234
 Database Version ......... ?
 Database Load Date ....... 2008-08-25 11:48:38

 System Files ............. 19 , 0 , 0 , 89 , 30 , 0 , 0
 Maximum Number of Files .. 100
 Number of Files Loaded ... 3
 Highest File Loaded ...... 89

 Size of RABN ............. 3 Bytes
 Current Log Tape Number .. 563
 Delta Save Facility ...... Inactive     Replicator Facility ...... Yes
 Recovery Aid Facility .... Inactive
 Universal Encoding Sup. .. Inactive


PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                                          Menu        ↩
```

If a standard Adabas database is selected and replication is *not* started, the first screen of the Display General DB-Layout series of screens does not include the **Replication Facility** field on the first screen. If replication is started, the first screen looks the same as if an Event Replicator Server database is selected. The following is an example of a standard Adabas database *without* replication started.

```
15:39:17          ***** A D A B A S  BASIC  SERVICES *****       2013-04-03
DBID 1955                - Display General DB-Layout -              PDRG012


 Database Name ............ WIS1955
 Database Number .......... 1955
 Database Version ......... ?
 Database Load Date ....... 2010-12-14 11:29:33

 System Files ............. 19 , 2 , 10 , 0 , 0 , 0 , 0
 Maximum Number of Files .. 1000
 Number of Files Loaded ... 31
 Highest File Loaded ...... 400
 Trigger File Number ...... 10

 Size of RABN ............. 3 Bytes
 Current Log Tape Number .. 332
 Delta Save Facility ...... Inactive
 Recovery Aid Facility .... Inactive
 Universal Encoding Sup. .. Inactive

PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                                        Menu        ↩
```

Display File Layout Screen Differences

The Display File Layout screen consists of a series of screens that display general file information. These screens vary slightly, depending on whether replication has been turned on for a standard Adabas database file.

If an Event Replicator Server file is selected, the screens are unchanged and remain as described in the AOS documentation.

If a standard Adabas database file is selected and if replication has *not* been turned on for the Adabas database file, the screens are unchanged and remain as described in the AOS documentation.

However, if a standard Adabas database file is selected and replication has been turned on for the database file, the first screen of the Display File Layout series of screens includes the field **Replication** with a value of "Yes".

```
16:37:26          ***** A D A B A S  BASIC  SERVICES *****       2013-04-02
DBID 1955                    - Display File Layout -                 PDRF043
 ****************
 *  File 1      *   EMPLOYEES
 ****************

 Records loaded ..... 1107          Date loaded .......... 2009-02-17 19:07:58
 TOP ISN ............ 1107          Date of last update .. 2012-04-20 10:22:03
 Max ISN expected ... 1695          Max Compr Rec Lngth .. 5060
 Minimum ISN ........ 1             Asso/Data Padding .... 10%/10%
 Size of ISN ........ 3 Bytes       Highest Index Level .. 3
 Number of Updates .. 188           RPLUPDATEONLY. No
 ISN Reusage ........ Yes           USERISN ...... No   PGMREFRESH ..... No
 Space Reusage ...... Yes           MIXDSDEV ..... No   NOACEXTENSION .. No
 ADAM File .......... No            Spanned rec .. No   MU/PE indices .. 1
 Ciphered File ...... No            Replication .. Yes  Privileged Use . No
 Coupled Files ...... None
 Blk per DS Extent .. 0
 Blk per UI Extent .. 0             Total Changed Blks ... 124
 Blk per NI Extent .. 0             Multi Client File .... 0
                                    Press Enter to display more information
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit      Refresh                             Menu       ↩
```

In addition, if replication has been turned on and you press the Enter key on the first screen of the Display File Layout series of screens, the following additional Display File Layout screen appears. If replication is not turned on for the file, this screen does not appear.

```
 02:08:31          ***** A D A B A S  BASIC  SERVICES *****       2013-04-02
 DBID 1955                    - Display File Layout -                 PDRF043



  Target Replicator ID ........... 1234
     Primary Key .................
     Replicate UPD before image .. Yes









 PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
 Help               Exit      Refresh                             Menu       ↩
```

The remaining Display File Layout screens appear as described in the existing AOS documentation.

The following table describes the replication information displayed on this screen. These parameters can be changed on the **Modify File Parameters** screen:

Field Name	Description
Primary Key	The primary key of the records to be replicated.
Replicate UPD before image	Indicates whether or not before images of data storage are collected for replication during the update of a record on a file.
Target Replicator ID	The database ID of the Event Replicator Server to be used for replication.

Modify File Parameters Screen Differences

The Modify File Parameters screen allows you to modify file parameters for a file in an Adabas database. This screen varies slightly, depending on whether the database in which the file resides is an Event Replicator Server or a standard Adabas database.

If an Event Replicator Server file is selected, the classic Modify File Parameters screen appears, as described in the AOS documentation. There are no changes to the fields shown on the screen or to the number of screens available.

If a standard Adabas database file is selected, the **Replication** field appears. In addition, if replication has been turned on for the file, the PF6 key is activated:

```
17:54:11          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
DBID 1955                -  Modify File Parameters  -               PFLM022

File No. ... 1
File Name .. EMPLOYEES                        in Parallel ....... YES
--------------------------------------------------------------------------------
ASSO PFAC ......... 10                Max. UI Blks per extent .. 0
DATA PFAC ......... 10                Max. NI Blks per extent .. 0
Max. RECL .........  5060             Max. DS Blks per extent .. 0
                                      ISN Reuse ................ ON_
New File Name ..... EMPLOYEES_______         with RESET ........ ___
New File No. ...... 1                 DS Reuse .................. ON_
User ISN .......... OFF                      with RESET ........ ___
File Password .....                   Mixed DS Device .......... OFF
Filereadonly ...... OFF               Program Refresh .......... OFF
Spanned Records ... OFF               Max occur system fields .. 0
MU/PE indices ..... 1                 Replication .............. ON
Reptor update only. OFF
AlphaNum Encoding . 0
WideChar Encoding . 0

PF1----- PF2------ PF3------ PF4------ PF6------ PF7----- PF8----- PF12-----
Help               Exit               Repl Parm                  Menu       ↩
```

If you press the PF6 key, the following pop-up File Replication Parameters screen appears on which you can specify additional information for the replicated file. (This pop-up screen also appears if you change the **Replication** field to ON when it was set OFF before):

```
  - File Replication Parameters -

*****************
* File   1      *
*****************

Target replicator ID .......... 1234_

Primary key for replication ...

Replicate UPD before image .... YES

in Parallel ................... NO_

PF3---
Cancel                                  ↩
```

You can control replication for an Adabas file using the Modify File Parameters screen as well as this pop-up screen. For more information, read *Controlling Replication of an Adabas Database File*, elsewhere in this section.

High Water Marks Screen Differences

The High Water Marks screen displays the maximum percent of use of selected pools and queues in the current session, and the date and time when the high point was reached. This screen varies slightly, depending on whether the current database is an Event Replicator Server or a standard Adabas database.

If an Event Replicator Server is selected, the classic High Water Marks screen appears, as described in the AOS documentation. There are no changes to the fields shown on the screen.

If the database is a standard Adabas database, statistics for Replication are shown on the second page of the High Water Marks screens:

```
 18:04:19          ***** A D A B A S  BASIC  SERVICES *****          2013-04-03
 DBID 1957                    -  High Water Marks  -                 PACUH12


  Pool / Queue       I    Size    I    Used    I %Used I   Date       Time   I
-------------------------------------------------------------------------------
Attached Buffer(NAB) I     409600 I      49152 I  12.0 I                      I
Command Queue  (NC)  I      38400 I        384 I   1.0 I 2013-04-02 12:54:20 I
Format Pool    (LFP) I     150000 I       6112 I   4.0 I 2013-04-03 12:02:12 I
Hold Queue     (NH)  I      83200 I          0 I   0.0 I 2013-04-03 14:00:29 I
ISN-List Table (LI)  I     360000 I          0 I   0.0 I                      I
Seq. Cmd. Table(LQ)  I      20000 I        436 I   2.1 I 2013-04-03 14:03:34 I
User Queue     (NU)  I      61200 I       1200 I   1.9 I 2013-04-02 12:55:16 I
Unique DE Pool (DUQ) I      50000 I          0 I   0.0 I                      I
Security Pool  (LCP) I      10000 I          0 I   0.0 I                      I
UQ File List   (UQF) I      19584 I        144 I   0.7 I 2013-04-02 12:55:16 I
ATM Trans. IDs (XID) I          0 I          0 I   0.0 I                      I
Work Pool      (LWP) I    1500000 I      36768 I   2.4 I 2013-04-03 14:00:32 I
Redo Pool      (LRDP)I          0 I          0 I   0.0 I                      I
                                                                  Page 1 of 2

 PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
 Help               Exit      Refresh                   +        Menu       ↩
```

```
 18:04:19          ***** A D A B A S  BASIC  SERVICES *****          2013-04-03
 DBID 1957                    -  High Water Marks  -                 PACUH12


  Pool / Queue       I    Size    I    Used    I %Used I   Date       Time   I
-------------------------------------------------------------------------------
Replication     (RPL)I    1500000 I        784 I   0.0 I 2013-04-02 12:54:20 I
Work Part 1      (LP)I       1000 I          1 I   0.1 I                      I
Work Part 2   (LWKP2)I        188 I          2 I   1.0 I 2013-04-03 12:02:12 I
Work Part 3          I       1502 I          0 I   0.0 I                      I
PLOG Prot buf(NPLOGB)I          1 I          1 I 100.0 I 2013-04-02 08:16:05 I
Work Prt1 Prot bf(NW)I          1 I          1 I 100.0 I 2013-04-02 08:16:05 I









                                                                  Page 2 of 2

 PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
 Help               Exit      Refresh             -                Menu       ↩
```

Managing Replication from AOS

You can manage replication from the Adabas Online System (AOS). Specifically, you can:

- Activate and deactivate Event Replicator definitions
- Activate and deactivate Event Replicator Servers and their files
- Open and close Event Replicator destination and IQUEUE definitions
- Display some of the Event Replicator global parameter settings
- Display Event Replicator statistics
- Run the Event Replicator **RPLCHECK** and **RPLCLEANUP** operator commands.
- Access the Adabas Event Replicator Subsystem from which you can maintain Event Replicator definitions.

Note: AOS is not able to modify file parameters for Adabas files incorporated in Predict. This is because AOS cannot modify the FCB of these files. Error messages are produced when such an attempt is made. We recommend that you use a supported version of Predict to make file parameter updates for Adabas files incorporated in Predict.

This section covers the following topics:

- Accessing the Adabas Event Replicator Subsystem from AOS

Controlling Replication of an Adabas Database File

You can control replication for an Adabas database file in AOS. Specifically, you can:

- turn replication on or off for the file
- indicate whether the change requests made in AOS should run simultaneously with any other running Adabas functions that might be modifying the file.
- specify the primary key for replication
- specify the replication target database ID
- indicate whether or not before images are stored when updates occur to the Adabas file.

To control replication for an Adabas database file in AOS:

1 Make sure the Adabas database ID is selected on any AOS screen.

2 Access the Modify File Parameters screen, as described in your AOS documentation.

```
17:54:11          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
DBID 1955                 -  Modify File Parameters  -               PFLM022

File No. ... 1
File Name .. EMPLOYEES                          in Parallel ....... YES
--------------------------------------------------------------------------------
ASSO PFAC ......... 10                 Max. UI Blks per extent .. 0
DATA PFAC ......... 10                 Max. NI Blks per extent .. 0
Max. RECL .........  5060             Max. DS Blks per extent .. 0
                                       ISN Reuse ................ ON_
New File Name ..... EMPLOYEES_______          with RESET ........ ___
New File No. ...... 1                  DS Reuse .................. ON_
User ISN .......... OFF                       with RESET ........ ___
File Password .....                    Mixed DS Device .......... OFF
Filereadonly ...... OFF                Program Refresh .......... OFF
Spanned Records ... OFF                Max occur system fields .. 0
MU/PE indices ..... 1                 Replication ............... OFF
Reptor update only. OFF
AlphaNum Encoding . 0
WideChar Encoding . 0

PF1----- PF2------ PF3------ PF4------ PF6------ PF7----- PF8----- PF12-----
Help              Exit               Repl Parm                  Menu       ↩
```

3 Use the **Replication** field to turn replication for the file ON and OFF.

4 Optionally, use the **in Parallel** field to indicate whether this change request (to turn replication processing on or off) should run simultaneously with any other running Adabas functions

that might be modifying the file. Specify YES to force this change request and other running Adabas functions to be processed simultaneously. If you specify NO, Adabas tries to complete this replication change request, but may reject it if other running Adabas functions against the file conflict with it.

Note: File integrity is always maintained. However, if you specify YES, the functions that are executing against the file may experience difficulty. Make sure you are aware of the impact on your environment and the consequences of performing your selected functions against a file while using the **in Parallel=YES** option.

The **in Parallel** field is always set to NO unless you manually change it. When you do change it, the YES setting is only used for the current replication change request. When you return to this screen, this field will be set back to NO.

5 Press the PF6 key to review and change additional information for the replicated file.

The following pop-up File Replication Parameters screen appears (this pop-up screen also appears if you change the **Replication** field to ON when it was set OFF before):

```
  - File Replication Parameters -

*****************
* File   1      *
*****************

Target replicator ID .......... 1234_

Primary key for replication ...

Replicate UPD before image .... YES

in Parallel ................... NO_

PF3---
Cancel                                  ↩
```

6 Fill in the fields on this pop-up screen, as described in the following table:

Field Name	Specify
in Parallel	Whether this change request (specified by any of the other fields on this screen) should run simultaneously with any other running Adabas functions that might be modifying the file. Specify YES to force this change request and other running Adabas functions to be processed simultaneously. If you specify NO, Adabas tries to complete this replication change request, but may reject it if other running Adabas functions against the file conflict with it. **Note:** File integrity is always maintained. However, if you specify YES, the functions that are executing against the file may experience difficulty. Make sure you are aware of the impact on your environment and the consequences of performing your selected functions against a file while using the in Parallel=YES option. This field is always set to NO unless you manually change it. When you do change it, the YES setting is only used for the current replication change request. When you return to this screen, this field will be set back to NO.
Primary key for replication	The primary key of the records to be replicated. The field name specified must be a descriptor on the file. Note that *descriptor* in this case is used generically, as the field may be a descriptor, subdescriptor, superdescriptor, etc. This is the equivalent of specifying the KEY parameter for the ADADBS REPLICATION function, as described in *Event Replicator for Adabas Reference Guide*. If the primary key is left blank (or set to blanks), it is the equivalent of specifying the NOKEY parameter for the ADADBS REPLICATION function, as described in *Event Replicator for Adabas Reference Guide*. For more information about how the primary key is used, read *Adabas Nucleus Detail Processing* in *Event Replicator for Adabas Concepts*.
Replicate UPD before image	Indicates whether or not before images of data storage are collected for replication during the update of a record on a file. This is the equivalent of specifying the DSBI parameter for the ADADBS REPLICATION function, as described in *Event Replicator for Adabas Reference Guide*. For more information about this processing, read *Nucleus Processing* in *Event Replicator for Adabas Concepts*.
Target replicator ID	The database ID of the Event Replicator Server to be used for replication. This is the equivalent of specifying the TARGET parameter for the ADADBS REPLICATION function, as described in *Event Replicator for Adabas Reference Guide*.

7 After making all modifications to the Modify File Parameters and File Replication Parameters screens, pressing Enter will save your modifications and return you to the prior menu.

Accessing the Event Replicator Server Definition Management Area

To manage Replication definitions from AOS, you must first access the Event Replicator Server definition management area of AOS.

To access the Event Replicator Server definition management area of AOS:

1 Make sure an Event Replicator Server ID is selected on any AOS screen.

2 Navigate to the Session Monitoring screen.

3 Select **Replicator Management** (option V) on the command line of the Session Monitoring screen.

The Replicator Management screen appears.

```
18:10:20          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator 1234          - Replicator Management -                  PRPT002

                       Code   Service
                       ----   -------------------------
                        A     Activate/deact/open/close
                        D     Display Reptor definitions
                        F     Display Reptor statistics
                        H     Perform RPLCheck
                        L     Perform RPLCleanup
                        P     Perform RPLRefresh
                        R     Parameter subsystem
                        ?     Help
                        .     Exit
                       ----   -------------------------

          Code ......... _
          Database ID .. 1234  (WIS1234)


Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help     File Serv Exit                                        Menu       ↩
```

Activating and Deactivating Replication Definitions and Databases

You can activate and deactivate subscription and destination definitions for your Event Replicator Server. You can also activate and deactivate the database itself or individual files within the database.

Caution: Be careful when you activate and deactivate replication definitions and databases, especially if replication is ongoing at the time. Whenever you activate or deactivate definitions or databases, you run the risk of altering what data is replicated and how that replication occurs. If the Event Replicator Server receives data from an Adabas database for which it has no active definitions, replication simply does not occur.

To get started activating and deactivating replication definitions and databases:

- Select A on the Replicator Management screen.

  The Replicator Activate/Deact/Open/Close screen appears.

```
18:10:20          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator 1234   - Replicator Activate/Deact/Open/Close -          PRPT002

    Code    Service                      Code    Service
    ----    ---------------------        ----    ----------------------
     B      Activate Subscription         R      Deactivate Subscription
     D      Activate Destination          S      Deactivate Destination
     F      Activate DBID/File            U      Deactivate DBID/File
     I      Activate DBID                 X      Deactivate DBID
     K      Open Destination              Y      Close Destination
     O      Open Iqueue                   Z      Close Iqueue
     ?      Help                          .      Exit
    ----    ---------------------        ----    ----------------------
            Code ........... _
            Subscription ... ________
            Destination .... ________
            Iqueue ......... ________
            Database ID .... _____
            File ........... ____


PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help              Exit                                                  ↩
```

This section describes:

- Activating Definitions
- Deactivating Definitions
- Activating Event Replicator Servers and Files

- Deactivating Event Replicator Servers and Files

Activating Definitions

To activate subscription or destination definitions:

1 On the Replicator Activate/Deact/Open/Close screen, tab to the **Subscription** field and specify the name of a valid, inactive, **subscription definition**.

Or:

Tab to the **Destination** field and specify the name of a valid, inactive, **destination definition**.

2 Tab to the **Code** field and type either a B to activate the subscription or a D to activate the destination (depending on which kind of definition you named in Step 1).

3 Press ENTER.

The specified definition is activated for use.

Deactivating Definitions

To deactivate subscription or destination definitions:

1 On the Replicator Activate/Deact/Open/Close screen, tab to the **Subscription** field and specify the name of a valid, activated, **subscription definition**.

Or:

Tab to the **Destination** field and specify the name of a valid, activated, **destination definition**.

2 Tab to the **Code** field and type either an R to deactivate the subscription or an S to deactivate the destination (depending on which kind of definition you named in Step 1).

3 Press ENTER.

The specified definition is deactivated and cannot be used until it is activated again.

Activating Event Replicator Servers and Files

To activate Event Replicator Servers or a file within a selected Event Replicator Server:

1 On the Replicator Activate/Deact/Open/Close screen, tab to the **Database ID** field and specify the valid database ID of an inactive Event Replicator Server.

2 Optionally, tab to the **File** field and specify the number of a valid, inactive, file within the Event Replicator Server identified in Step 1.

3 Tab to the **Code** field and type either a F to activate a file within a the database (if you specified a file number in Step 2) or an I to activate the entire database.

4 Press ENTER.

The specified Event Replicator Server and file are activated for use.

Note: If no specific file is selected in the **File** field, all *deactivated* files that are known to the selected Event Replicator Server are activated.

Deactivating Event Replicator Servers and Files

To deactivate Event Replicator Servers or a file within a selected Event Replicator Server:

1 On the Replicator Activate/Deact/Open/Close screen, tab to the **Database ID** field and specify the valid database ID of an active Event Replicator Server.

2 Optionally, tab to the **File** field and specify the number of a valid, active, file within the Event Replicator Server identified in Step 1.

3 Tab to the **Code** field and type either a U to deactivate a file within a the database (if you specified a file number in Step 2) or an X to deactivate the entire database.

4 Press ENTER.

The specified Event Replicator Server and file are deactivated and cannot be used until it is activated again.

Note: If no specific file is selected in the **File** field, all *activated* files that are known to the selected Event Replicator Server are deactivated.

Opening and Closing Event Replicator Destination and IQUEUE Definitions

You can open and close destination and IQUEUE definitions for your Event Replicator database.

To get started activating and deactivating replication definitions and databases:

- Select A on the Replicator Management screen.

  The Replicator Activate/Deact/Open/Close screen appears.

```
18:10:20          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator 1234   - Replicator Activate/Deact/Open/Close -          PRPT002

    Code    Service                       Code    Service
    ----    --------------------          ----    ----------------------
     B      Activate Subscription          R      Deactivate Subscription
     D      Activate Destination           S      Deactivate Destination
     F      Activate DBID/File             U      Deactivate DBID/File
     I      Activate DBID                  X      Deactivate DBID
     K      Open Destination               Y      Close Destination
     O      Open Iqueue                    Z      Close Iqueue
     ?      Help                           .      Exit
    ----    --------------------          ----    ----------------------
            Code ........... _
            Subscription ... ________
            Destination .... ________
            Iqueue ......... ________
            Database ID .... _____
            File ........... ____


PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help              Exit                                                   ↩
```

This section describes:

- Opening and Closing Destination Definitions
- Opening and Closing IQUEUE Definitions

Opening and Closing Destination Definitions

To open or close destination definitions:

1 On the Replicator Activate/Deact/Open/Close screen, tab to the **Destination** field and specify the name of a valid, inactive, **destination definition**.

2 Tab to the **Code** field and type either a K to open the destination or a Y to close the destination (depending on what you want to do).

3 Press ENTER.

The specified destination definition is either opened or closed, as requested.

Opening and Closing IQUEUE Definitions

To open or close IQUEUE definitions:

1 On the Replicator Activate/Deact/Open/Close screen, tab to the **Iqueue** field and specify the name of a valid **IQUEUE definition**.

2 Tab to the **Code** field and type either a O to open the IQUEUE definition or a Z to close the IQUEUE definition (depending on what you want to do).

3 Press ENTER.

The specified IQUEUE definition is either opened or closed, as requested.

Displaying Event Replicator Global Value Definitions

You can review some of the Event Replicator global values set in the Replicator system file using the Adabas Event Replicator Subsystem screens.

Note: You can review these settings; you cannot change them here. You must use the Adabas Event Replicator Subsystem to modify them.

To display Event Replicator global values:

1 Select D on the Replicator Management screen.

The Display Reptor Definitions menu appears.

```
17:12:33          ***** A D A B A S  BASIC  SERVICES *****          2017-09-12
Replicator 1234         - Display Reptor Definitions -               PRPTD02

Code   Service                          Code   Service
----   -------------------------        ----   -------------------------
 A     Global definitions                Q     Input queue definitions
 D     Destination definitions           R     Resend buffer definitions
 G     Global format definitions         S     Subscription definitions
 I     Initial-state definitions
 ?     Help
 .     Exit
----   -------------------------        ----   -------------------------

       Code ......... _
       Database ID .. 1234  (WIS1234)
       Select name .. ________




Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                                        Menu        ↩
```

2 Select A on the Display Reptor Definitions screen.

The Global Definitions screen appears.

```
17:31:13          ***** A D A B A S  BASIC  SERVICES *****          2017-09-12
Replicator 1234             - Global Definitions -                   PRPTD02

Global definitions -------------------------  --------------------------------
Number of subtasks ...........             3  Broker name ...... BKIMBTSO
Maximum output msg size ......       100,000  Verify mode ...... No
Input req. error msg limit ...            10  Rec. PLOG info ... No
Input req.err msg limit time .            60  FB Validation .... None
Max. decompressed rec. size ..        32,767  Retry interval ...              0
MAXVARRECORDSIZE .............        32,767  Retry count ......             10
Subtaskwait ..................            10  Loginputtransact .             80
NPADACALLS ...................             3   status of ....... Off
GQFULLDELAY ..................            60  GOpen ............ Yes
STATINTERVAL .................             0  SLOGACTHRESHOLD ..             70
SLOGCHECKINTERVAL ............             0  SLOGDSTHRESHOLD ..             70
ASSOTHRESHOLD ................            80  SLOGNITHRESHOLD ..             70
DATATHRESHOLD ................            80  SLOGUITHRESHOLD ..             70
Replicator version detected .. 361

                                       Press 'Enter' to see TLOG information

PF1-----           PF3-----   PF4-----  PF6--------  PF7--------  PF12-----
Help               Exit       Refresh                           Menu
```

```
17:31:13          ***** A D A B A S  BASIC  SERVICES *****         2017-09-12
Replicator 1234              - TLOG Definitions -                   PRPTD02

TLOG definitions ------------------      ------------------------------------
RPL-pool maximum % ............  50      Input request rejected ........   0
RPL-pool restart % ............  40      Subscription status request ...   0
Transaction input .............   0      Transaction resend request ....   0
No subscription match .........   0      Initial-state started .........   0
Queued for completion .........   0      Initial-state completed .......   0
Transaction completion ........   0      Input request error ...........   0
Input request received ........   0      Replicator version detected ... 361




PF1-----           PF3-----   PF4-----  PF6--------  PF7--------  PF12-----
Help               Exit       Refresh                           Menu
```

The global values displayed on this screen are summarized in the following table:

Parameter Name	Displays
Broker name	The default EntireX Broker stub name to be used by the Event Replicator Server if no other name is specified. EntireX Broker is a component of webMethods EntireX.
FB Validation	The level of format buffer validation that should occur for subscriptions. Possible values are a blank, "A", "D", "None", or "W". ■ If a blank or "None" are listed, format buffer validation is not performed at Event Replicator initialization, at the initial handshake of databases, or when an updated FDT is received. Format buffer validation still occurs during the subscription phase of transaction processing, with validation errors written to the URBRRSP field of the URBR record. ■ If "A" is listed, format buffer validation is performed. If validation errors are identified at Event Replicator Server initialization, the Event Replicator Server is terminated. If validation errors are identified after Event Replicator initialization, warning messages are issued for each format buffer in error. ■ If "D" is listed, format buffer validation is performed. If validation errors are identified, the subscription for which format buffer validation failed is deactivated and warning messages are issued for each format buffer in error. ■ If "W" is listed, format buffer validation is performed. If validation errors are identified, warning messages are issued for each format buffer in error.
GQFULLDELAY	The number of seconds between retry attempts when resending output transactions to a previously-full webMethods EntireX or MQSeries destination.

Parameter Name	Displays
GOpen	Whether or not destinations with DOPEN=GLOBAL and input queues with IQOPEN=GLOBAL specified in their definitions should be opened at Event Replicator Server startup.
Initial-state completed	The level of transaction logging that occurs when an initial-state information request has completed. Possible values are "0" (no logging), "1" (log event and URBS), "2" (log event and URBI, if available), or "3" (log event, URBS, and URBI, if available).
Initial-state started	The level of transaction logging that occurs when a user request for initial-state information for a file has started. Possible values are "0" (no logging), "1" (log event and URBS), "2" (log event and URBI), or "3" (log event, URBS, and URBI).
Input request error	The level of transaction logging that occurs when a user request is rejected due to an error in carrying out the request. Possible values are "0" (no logging), "1" (log event and URBS), "2" (log event and URBI), or "3" (log event, URBS, and URBI).
Input req. error msg limit time	The interval during which the message limit specified by the **Input request error message limit** field applies.
Input request error msg limit	The maximum number of input request error messages issued by Event Replicator Server during the interval set by the **Input req. error msg limit time** field (also on this screen).
Input request received	The level of transaction logging that occurs when a user request has been received. Possible values are "0" (no logging), "1" (log event but no data), or "2" (log event and the entire input buffer before and after translation if appropriate).
Input request rejected	The level of transaction logging that occurs when a user request has been received. Possible values are "0" (no logging), "1" (log event but no data), or "2" (log event and the entire input buffer before and after translation if appropriate).
`Loginputtransact` & `status of`	Whether and when database-related input transactions are written to the SLOG system file. If a number appears in the `Loginputtransact` parameter, it indicates a percentage of the setting of the LRPL parameter. this threshold is exceeded, database-related input transactions will be stored in and processed from the SLOG system file. When the threshold is no longer exceeded and all database-related input transactions have been processed, database-related input transactions from the Adabas nucleus are stored in and processed from the replication pool. The `status of` parameter indicates whether or not SLOG system file usage is always active for database-related input transactions. When "On" is shown, database-related input transactions from the Adabas nucleus are written to the SLOG system file. When "Off" is shown, the transactions are processed, as usual, using the Event Replicator Server replication pool.
Max. decompressed rec. size	The maximum length (in bytes) of any decompressed record that can be processed by the Event Replicator Server.
Maximum output message size	The maximum output message size for the Event Replicator for Adabas.

Parameter Name	Displays
MAXVARRECORDSIZE	The maximum length (in bytes) of variable decompressed records that can be processed by the Event Replicator Server.
No subscription match	The level of transaction logging that occurs when a transaction is not queued to any subscription in Event Replicator Server. Possible values are "0" (no logging), "1" (log event and input transaction data), "2" (log event, input transaction, and file/record information), or "3" (log event and all available input information for the event).
NPADACALLS	The maximum number of parallel Adabas calls that can be made.
Number of subtasks	The number of subtasks in Event Replicator Server.
Queued for completion	The level of transaction logging that occurs prior to a transaction being assigned to the completion queue. Possible values are "0" (no logging), "1" (log event and input transaction data, "2" (log event, input transaction, and file/record information), or "3" (log event and all input information available for the event).
Rec. PLOG info	Whether or not PLOG information is saved in the Replicator system file. Valid values are "Y" (store the information) or "N" (do not store the information).
Replicator version detected	The version of the Event Replicator Server detected.
Retry count	The default number of times that an attempt to retry opening a destination or input queue is attempted at the interval specified by the Retry interval parameter.
Retry interval	The default number of seconds between attempts to retry opening a destination or input queue.
RPL-pool maximum %	The maximum percentage of the Event Replicator Server replication pool that can be used for transaction log (TLOG) processing. .
RPL-pool restart %	The amount of available Event Replicator Server replication pool storage that must be available before transaction logging (TLOG logging) can restart.
Subscription status request	The level of transaction logging that occurs when a user request for status on an Event Replicator Server resource has been processed. Possible values are "0" (no logging), "1" (log event and URBS), "2" (log event and URBI), or "3" (log event, URBS, and URBI).
Subtaskwait	The number of seconds the Event Replicator Server will wait for the successful initialization of an input or output subtask.
Transaction completion	The level of transaction logging that occurs when a transaction has been fully processed and Adabas has been informed that the transaction was successfully replicated. Possible values are "0" (no logging) or "1" (log event and input transaction data).
Transaction input	The level of transaction logging that occurs when a transaction is taken off the input queue and put on the transaction assignment queue. Possible values are "0" (no logging), "1" (log event and input transaction data), "2" (log event, input transaction, and file/ record information), or "3" (log event and all available input information for the event).
Transaction resend request	The level of transaction logging that occurs when a user request to retransmit a specific transaction has been processed. Possible values are "0" (no logging), "1"

Parameter Name	Displays
	(log event and URBS), "2" (log event and URBI), or "3" (log event, URBS, and URBI).
Verify mode	Whether the Event Replicator for Adabas should run in verify (test) mode or not. Possible values are "Y" (run in verify mode) or "N" (do not run in verify mode).

Displaying Event Replicator Destination Definitions

You can review some of the Event Replicator destination definitions set in the Replicator system file using the Adabas Event Replicator Subsystem screens.

Note: You can review these settings; you cannot change them here. You must use the Adabas Event Replicator Subsystem to modify them.

To display Event Replicator destination definitions:

1 Select D on the Replicator Management screen.

The Display Reptor Definitions menu appears.

```
18:12:33          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator 1234         - Display Reptor Definitions -               PRPTD02

Code   Service                          Code   Service
----   -------------------------        ----   -------------------------
 A     Global definitions                Q     Input queue definitions
 D     Destination definitions           R     Resend buffer definitions
 G     Global format definitions         S     Subscription definitions
 I     Initial-state definitions
 ?     Help
 .     Exit
----   -------------------------        ----   -------------------------

       Code ......... _
       Database ID .. 1234  (WIS1234)
       Select name .. ________



Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                                        Menu        ↩
```

2 Select D on the Display Reptor Definitions screen.

A list of destination definitions appears on the Destination Information screen.

```
10:50:10          ***** A D A B A S  BASIC  SERVICES *****          2013-04-06
Replicator 1234          - Destination Definitions -                PRPTD33

Mark  Name      Type    Op/Cls  Act/Inact
  -   --------  ------  ------  --------
  _   ADABAS1   Adabas  Open    Active
  _   DESTFILE  File    Closed  Active
  _   DI040155  ETB     Closed  Inactive
  _   DI042060  ETB     Closed  Inactive
  _   DI042200  ETB     Closed  Inactive
  _   DI046088  ETB     Closed  Inactive
  _   DI062026  ETB     Closed  Inactive
  _   DI062029  ETB     Closed  Inactive
  _   DI062035  ETB     Closed  Inactive
  _   DI062055  ETB     Closed  Inactive
  _   DI062079  ETB     Closed  Inactive
  _   DI062106  ETB     Closed  Inactive
  _   DI062143  ETB     Closed  Inactive
  _   DI064121  ETB     Closed  Inactive     Enter (S)elect (A)ctivate (O)pen
  _   DI120248  ETB     Closed  Inactive                (I)nactivate (C)lose

PF1-----  PF2-----  PF3-----  PF4-----  PF7--------  PF8--------  PF12-----
Help      Repos     Exit      Refresh   -           +           Menu       ↩
```

From this screen, you can review a destination definition in more detail (see the next step) or you can quickly activate, deactivate, open, or close a definition, as described in the following table.

Type this character in the Mark Column next to a destination definition:	To:
A	Activate the definition.
C	Close the definition.
I	Deactivate the definition.
O	Open the definition.
S	Select the definition for review. See the next step in these instructions.

Select a destination definition to review by typing an "S" in the corresponding **Mark** column and pressing Enter.

A Destination Definitions panel, such as this one, appears showing the details of the destination definition.

```
18:26:36          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator 1234          - Destination Definitions -                PRPTD33

-------------------------------------    ---------------------------------------
Name ................. ADABAS1          TLOG indicators:
Type ................. Adabas             Assignment ..........   0
                                         Output completion ...   0
Informational flags:                     Slog write ..........   0
  Destination is open                    Slog read ...........   0
  Destination is active                  Adabas ..............   0

                                         DAERROR setting ....... Altaction
                                         Open dest at startup .. Global
Open retry:
  Interval ........... Default
  Count .............. Default
  Time ............... N/A
  Retries remaining ..

                                                    Press 'Enter' to continue

PF1-----           PF3-----   PF4-----  PF6--------  PF7--------  PF12-----
Help               Exit                DB/File exp               Menu       ↩
```

If you press PF6 on the screen, another screen such as this one appears:

```
18:26:36          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator 1234          - Destination Definitions -                PRPTD33

--------------------------    --------------------
Name ............ ADABAS1
Input DBID ......  1955       Target DBID ..  1957
Input File ......     1       Target File ..     1

DAREPLICATEUTI .. Yes
DATMETHOD ....... ISN









Displaying   1 of   3                              Press 'Enter' to continue

PF1-----           PF3-----   PF4-----  PF6--------  PF7--------  PF12-----   ↩
```

Displaying Event Replicator Global Format Buffer (GFB) Definitions

You can review some of the Event Replicator global values set in the Replicator system file using the Adabas Event Replicator Subsystem screens.

Note: You can review these settings; you cannot change them here. You must use the Adabas Event Replicator Subsystem to modify them.

To display Event Replicator global format buffer (GFB) definitions:

1 Select D on the Replicator Management screen.

The Display Reptor Definitions menu appears.

```
18:12:33          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator 1234         - Display Reptor Definitions -               PRPTD02

Code   Service                           Code   Service
----   -------------------------         ----   -------------------------
 A     Global definitions                 Q     Input queue definitions
 D     Destination definitions            R     Resend buffer definitions
 G     Global format definitions          S     Subscription definitions
 I     Initial-state definitions
 ?     Help
 .     Exit
----   -------------------------         ----   -------------------------

       Code ......... _
       Database ID .. 1234  (WIS1234)
       Select name .. ________




Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                                        Menu        ↩
```

2 Select G on the Display Reptor Definitions screen.

A list of global format definitions appears on the Global Format Definitions screen.

```
10:47:59          ***** A D A B A S  BASIC  SERVICES *****          2013-04-06
Replicator 1234          - Global Format Definitions -               PRPTD42

Mark  Glb Fmt   DBID   File
 -    --------  -----  -----
 _    FILE200   42     200
 _    F040155   40     155
 _    F042060   42     60
 _    F046088   46     88
 _    F062026   62     26
 _    F062029   62     29
 _    F062035   62     35
 _    F062055   62     55
 _    F062079   62     79
 _    F062106   62     106
 _    F062143   62     143
 _    F064121   64     121
 _    F120248   120    248
 _    F215168   215    168


PF1-----  PF2-----  PF3-----  PF4-----  PF7--------  PF8--------  PF12-----
Help                Exit      Refresh                           Menu       ↩
```

3 Select a global format buffer definition to review by typing an "S" in the corresponding **Mark** column and pressing Enter.

A Selected Global Format panel appears showing the details of the GFB definition.

```
10:47:59          ***** A D A B A S  BASIC  SERVICES *****          2013-04-06
Replicator 1234           - Selected Global Format -                PRPTD42

Global format name .. FILE200                    Press 'Enter' to continue
Format buffer
 AA,10,A,BB,10,A.









PF1-----           PF3-----   PF4-----  PF6--------  PF7--------  PF12-----
Help               Exit                                         Menu       ↩
↩
```

Displaying Event Replicator Initial-State Definitions

You can review some of the Event Replicator initial-state definitions defined in the Replicator system file using the Adabas Event Replicator Subsystem screens.

Note: You can review these settings; you cannot change them here. You must use the Adabas Event Replicator Subsystem to modify them.

To display Event Replicator initial-state definitions:

1 Select D on the Replicator Management screen.

The Display Reptor Definitions menu appears.

```
18:12:33          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator 1234        - Display Reptor Definitions -               PRPTD02

Code   Service                          Code   Service
----   -------------------------        ----   -------------------------
 A     Global definitions                Q     Input queue definitions
 D     Destination definitions           R     Resend buffer definitions
 G     Global format definitions         S     Subscription definitions
 I     Initial-state definitions
 ?     Help
 .     Exit
----   -------------------------        ----   -------------------------

       Code ......... _
       Database ID .. 1234  (WIS1234)
       Select name .. ________




Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                                    Menu        ↩
```

2 Select I on the Display Reptor Definitions screen.

A list of initial-state definitions appears on the Initial-State Definitions screen.

```
18:30:53          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator 1234         - Initial-State Definitions -               PRPTD72

Mark  Name     Max Concur.   Subscription      # Subs    # Dests   # DB/Files
  -   -------- ------------ -------- --------  -------   -------   -------
  _   I040155             1 SI040155                1         1         1
  _   I042060             1 SI042060                1         1         1
  _   I042200             1 SI042200                1         1         1
  _   I046088             1 SI046088                1         1         1
  _   I062026             1 SI062026                1         1         1
  _   I062029             1 SI062029                1         1         1
  _   I062035             1 SI062035                1         1         1
  _   I062055             1 SI062055                1         1         1
  _   I062079             1 SI062079                1         1         1
  _   I062143             1 SI062143                1         1         1
  _   I064121             1 SI064121                1         1         1
  _   I120248             1 SI120248                1         1         1
  _   I215168             1 SI215168                1         1         1
  _   TEST                1                                   1         1
  _

PF1-----           PF3-----   PF4-----  PF7--------  PF8--------  PF12-----
Help               Exit       Refresh                          Menu        ↩
```

3 Select an initial-state definition to review by typing an "S" in the corresponding **Mark** column and pressing Enter.

An Initial-State Definitions panel appears showing the details of the initial-state definition.

```
 18:30:53          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
 Replicator 1234          - Initial-State Definitions -              PRPTD72 ↩

                                                                              ↩

 ------------------------------------                                         ↩

 Name ................. I040155                                              ↩

 Max Concurrent .......             1                                         ↩

 Subscriptions  -------- -------- -------- -------- -------- -------- --------
       SI040155                                                               ↩

                                                                              ↩

                                                                              ↩

                                                                              ↩

 Destinations   -------- -------- -------- -------- -------- -------- --------
       DI040155                                                               ↩

                                                                              ↩

                                                                              ↩

                                                                              ↩

 DBID-file-type   ----- ----- -  ----- ----- -  ----- ----- -  ----- ----- -  ↩

   40    155   A                                                              ↩

                                                                              ↩

                                                                              ↩

                                                                              ↩

 PF1-----           PF3-----   PF4-----  PF6--------  PF8--------  PF12----- ↩

 Help               Exit                Exp DBID/Fi               Menu      ↩
 ↩
```

If you press the PF6 key on the initial-state definitions screen, details regarding the DBID files associated with the initial-state definition appears on the Expanded DBID File screen.

```
18:30:53          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator 1234            - Expanded DBID File -                   PRPTD72

Initial state: I040155
Mark  DBID  File  Type
  -   ----- ----- --------
  _   40    155   All ISNs
  _
  _
  _
  _
  _
  _
  _
  _
  _
  _
  _
  _
  _

PF1-----           PF3-----   PF4-----  PF7--------  PF8--------  PF12-----
Help               Exit                                                  ↩
```

4 If a selection criteria buffer is defined for a DBID/file combination, you can select a DBID on the Expanded DBID File screen by entering an "S" in the corresponding **Mark** column.

The Selcrit Buffer panel pops up displaying selection criteria for the database. Here is an example.

```
                    - Selcrit Buffer -

  Initial state name .. I1991433
  DBID ................ 1955
  File ................ 23
  SB length ........... 6

  Value:
   AA,GE.




  Press Enter to continue
                                                                        ↩
```

Displaying Event Replicator Input Queue Definitions

You can review some of the Event Replicator input queue definitions defined in the Replicator system file using the Adabas Event Replicator Subsystem screens.

Note: You can review these settings; you cannot change them here. You must use the Adabas Event Replicator Subsystem to modify them.

To display Event Replicator input queue definitions:

1 Select D on the Replicator Management screen.

The Display Reptor Definitions menu appears.

```
18:12:33          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator 1234         - Display Reptor Definitions -              PRPTD02

Code   Service                           Code   Service
----   -------------------------         ----   -------------------------
 A     Global definitions                 Q     Input queue definitions
 D     Destination definitions            R     Resend buffer definitions
 G     Global format definitions          S     Subscription definitions
 I     Initial-state definitions
 ?     Help
 .     Exit
----   -------------------------         ----   -------------------------

       Code ......... _
       Database ID .. 1234  (WIS1234)
       Select name .. ________




Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                                        Menu       ↩
```

2 Select Q on the Display Reptor Definitions screen.

A list of input queue definitions appears on the Input Queue Definitions screen.

```
18:34:04          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator 1234          - Input Queue Definitions -                PRPTD54

Mark  Name      Type      Status
  -   --------  --------  ------
  _   ABCDEF1   MQSeries  Closed
  _   ETBIN1    ETBroker  Open
  _   ETBQUEUE  ETBroker  Closed
  _   E2        ETBroker  Closed
  _
  _
  _
  _
  _
  _
  _
  _
  _
  _                                          Enter (S)elect (O)pen (C)lose

PF1-----           PF3-----   PF4-----  PF7--------  PF8--------  PF12-----
Help               Exit       Refresh                           Menu       ↩
```

This shows the name, type and status of the input queue definition. From this screen, you can review an input queue definition in more detail or you can quickly open or close a definition, as described in the following table.

Type this character in the Mark Column next to an input queue definition:	To:
C	Close the definition.
O	Open the definition.
S	Select the definition for review. S

If you select a definition (by entering an "s" in the **Mark** column corresponding to the definition), a detailed description of the input queue definition appears, such as this one:

```
18:34:04          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator 1234          - Input Queue Definitions -                PRPTD54

---------------------------------------- ------------------------------
Name ................. ETBIN1              IQOPEN ............... Global
Type ................. ETBroker
Status ............... Open

Open retry:
  Interval ........... Default
  Count .............. Default
  Time ............... N/A
  Retries remaining ..         0

 ETB queue names:
  ETB broker id:      10.20.74.81:3800:TCP
  ETB Service:        OUT1
  ETB Service Name:   JRSERV
  ETB Service Class:  REPTOR
                                                 Press 'Enter' to continue

PF1-----           PF3-----   PF4-----  PF6--------  PF7--------  PF12-----
Help               Exit                                         Menu       ↩
```

Displaying Event Replicator Resend Buffer Definitions

You can review some of the Event Replicator resend buffer definitions defined in the Replicator system file using the Adabas Event Replicator Subsystem screens.

Note: You can review these settings; you cannot change them here. You must use the Adabas Event Replicator Subsystem to modify them.

To display Event Replicator resend buffer definitions:

1 Select D on the Replicator Management screen.

The Display Reptor Definitions menu appears.

```
18:12:33          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator 1234        - Display Reptor Definitions -               PRPTD02

Code   Service                          Code   Service
----   -------------------------        ----   -------------------------
 A     Global definitions                Q     Input queue definitions
 D     Destination definitions           R     Resend buffer definitions
 G     Global format definitions         S     Subscription definitions
 I     Initial-state definitions
 ?     Help
 .     Exit
----   -------------------------        ----   -------------------------

       Code ......... _
       Database ID .. 1234  (WIS1234)
       Select name .. ________




Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                                        Menu       ↩
```

2 Select R on the Display Reptor Definitions screen.

A list of resend buffer definitions appears on the Resend Buffer Information screen.

```
19:21:15          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator 1234         - Resend Buffer Information -                PRPTD62

ResB Name Size          I  ResB Name Size          I  ResB Name Size
--------  ------------- I  --------  ------------- I  --------  -------------
RSNDFMIX      1,048,576 I                          I
                        I                          I
                        I                          I
                        I                          I
                        I                          I
                        I                          I
                        I                          I
                        I                          I
                        I                          I
                        I                          I
                        I                          I
                        I                          I
                        I                          I
                        I                          I
                        I                          I


PF1-----           PF3-----   PF4-----  PF7--------  PF8--------  PF12-----
Help               Exit       Refresh                           Menu       ↩
```

Displaying Event Replicator Subscription Definitions

You can review some of the Event Replicator subscription definitions defined in the Replicator system file using the Adabas Event Replicator Subsystem screens.

Note: You can review these settings; you cannot change them here. You must use the Adabas Event Replicator Subsystem to modify them.

To display Event Replicator subscription definitions:

1 Select D on the Replicator Management screen.

The Display Reptor Definitions menu appears.

```
18:12:33          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator 1234         - Display Reptor Definitions -               PRPTD02

Code   Service                           Code   Service
----   -------------------------         ----   -------------------------
 A     Global definitions                 Q     Input queue definitions
 D     Destination definitions            R     Resend buffer definitions
 G     Global format definitions          S     Subscription definitions
 I     Initial-state definitions
 ?     Help
 .     Exit
----   -------------------------         ----   -------------------------

       Code ......... _
       Database ID .. 1234  (WIS1234)
       Select name .. ________




Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                                        Menu        ↩
```

2 Select S on the Display Reptor Definitions screen.

A list of subscription definitions appears on the Subscription Information screen.

```
18:37:40          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator 1234          - Subscription Definitions -                PRPTD23

Mark  Name      Status   Destinations                DBID  File  DBID  File
  -   --------  -------- -------- -------- --------  ----- ----- ----- -----
  _   SADABAS1  Active   ADABAS1                    1955  1     1955  3
  _   SI040155  Active   DI040155                   40    155
  _   SI042060  Active   DI042060                   42    60
  _   SI042200  Active   DI042200                   42    200
  _   SI046088  Active   DI046088                   46    88
  _   SI062026  Active   DI062026                   62    26
  _   SI062029  Active   DI062029                   62    29
  _   SI062035  Active   DI062035                   62    35
  _   SI062055  Active   DI062055                   62    55
  _   SI062079  Active   DI062079                   62    79
  _   SI062106  Active   DI062106                   62    106
  _   SI062143  Active   DI062143                   62    143
  _   SI064121  Active   DI064121                   64    121
  _   SI120248  Active   DI120248                   120   248
Enter (S)elect (A)ctivate (I)nactivate

PF1----- PF2-----  PF3-----   PF4-----  PF7--------  PF8--------  PF12-----
Help     Repos     Exit       Refresh   -            +            Menu       ↩
```

3 Select a subscription definition to review by typing an "S" in the corresponding **Mark** column and pressing Enter.

A Subscription Definition panel appears, such as the following, showing the details of the subscription definition.

```
18:37:40          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator 1234          - Subscription Definitions -               PRPTD23

---------------------------    -------------------------  TLOG ---------------
Name ........... SI040155      Architec Key.. 2           Input ..........   0
Status ......... Active        Alpha Code ...          0  Filter .........   0
Max trans seq ..          0    Wide Code ....          0  Output .........   0
Resend buffer ..                                         Filter match ...   0
Increment Initial-state sequence number ..... No         Filter nomatch..   0
Deactivate Subscription when file deactive .. Yes        Filter ignore ..   0

Destinations ---- -------- -------- -------- -------- -------- --------
DI040155



DBID  File   ----- -----  ----- -----  ----- -----  ----- -----  ----- -----
40    155

                                                 Press 'Enter' to continue

PF1-----           PF3-----   PF4-----  PF6--------  PF7--------  PF12-----
Help               Exit                Exp DBID/Fi               Menu       ↩
```

If you press the PF6 key on the subscription definitions screen, details regarding the DBID files associated with the subscription definition appears on the Expanded DBID File screen.

```
12:07:42          ***** A D A B A S  BASIC  SERVICES *****          2013-04-07
Replicator 1234            - Expanded DBID File -                   PRPTD23

Subscription: SI040155
Mark  DBID  File
  -   ----- -----
  _   40    155
  _
  _
  _
  _
  _
  _
  _
  _
  _
  _
  _
  _
  _

PF1-----           PF3-----   PF4-----  PF7--------  PF8--------  PF12-----
Help               Exit                                                   ↩
```

You can select a DBID/file pair for review by entering an "S" in the **Mark** column associated with the pair. The Selected DBID File screen appears providing additional information about the file.

```
12:07:42          ***** A D A B A S  BASIC  SERVICES *****          2013-04-07
Replicator 1234             - Selected DBID File -                  PRPTD23

------------------------ ------------------------ ------------------------
Name ..... SI040155 DBID File ... 40    155   Alpha code ...
                    Exit name ...
Indicator Flags     ------------------------ ------------------------
No indicators to display




 - Mark to view                          - Mark to view
 _ Filter ................
 _ Filter GLB for. after .               _ Local Filter FB after ...
 _ Filter GLB for. before.               _ Local Filter FB before ..
 _ Global format after ... F040155       _ Local FB after length ...
 _ Global format before .. F040155       _ Local FB before length ..
 _ Global format key .....               _ Local FB key length .....

                                                 Press 'Enter' to continue

PF1-----          PF3-----   PF4-----  PF6--------  PF7--------  PF12-----
Help              Exit                                         Menu       ↩
```

Displaying Event Replicator Statistics

You can review statistics about Event Replicator processing using the AOS screens.

To get started activating and deactivating replication definitions and databases:

- Select F on the Replicator Management screen.

  The Display Reptor Statistics screen appears.

```
01:54:52          ***** A D A B A S  BASIC  SERVICES *****          2013-03-14
Replicator 1234        - Display Reptor Statistics -                PRPTS04

                        Code   Service
                        ----   -------------------------
                         B     Global statistics
                         D     Destination statistics
                         I     Input queue statistics
                         L     Load/replay tokens
                         R     Subscription statistics
                         S     Output Subtask statistics
                         ?     Help
                         .     Exit
                        ----   -------------------------
          Code ......... _
          Database ID .. 1234  (WIS1234)



Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                                        Menu       ↩
```

This section covers the following topics:

- Displaying Event Replicator Global Statistics
- Displaying Event Replicator Destination Statistics
- Displaying Event Replicator Input Queue Statistics
- Displaying Event Replicator Subscription Statistics
- Displaying Event Replicator Output Subtask Statistics

In addition, to these statistics, if you select option **L** on the **Display Reptor Statistics** screen, the load/replay tokens you can use are listed.

Displaying Event Replicator Global Statistics

To display Event Replicator global statistics :

1 On the Display Reptor Statistics screen, tab to the **Code** field and type a B.

2 Press ENTER.

The Global Statistics screen appears, showing Event Replicator statistics.

```
18:39:31          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator 1234             - Global Statistics -                   PRPTS14

------------------------------------------------------------
Input transactions ...........                          0
Pending input transactions ...                          0
Output items .................                          2
Pending items ................                          0
Output messages ..............                          2
 Bytes sent ..................                        384
Input messages ...............                          0
 Bytes received ..............                          0
Input commits ................                          0
Input backouts ...............                          0
Items de-logged ..............                          0
Items logged .................                         13
Items on SLOG ................                        611



                                                      More: press PF8

PF1---  PF3---  PF4-----  PF5--------  PF6--------  PF7---  PF8---  PF12-----
Help    Exit    Refresh   DBID/Token                +       Menu      ↩
```

```
 18:39:31          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
 Replicator 1234             - Global Statistics -                   PRPTS14

 Database-related input transactions
 -------------------------------------------
 De-logged ......                         0
 Logged .........                         0
 On SLOG ........                         0
 FCBs on SLOG ...                         0

 Log input transaction status. Off









 PF1---  PF3---  PF4-----  PF5--------  PF6--------  PF7---  PF8---  PF12-----
 Help    Exit    Refresh   DBID/Token                -               Menu     ↩
 ↩
```

The PF7 and PF8 keys allow you to toggle between the two screens of this display.

When there are individual replay dbid/tokens available or being executed, pressing PF5 will show you a list of what is currently active.

Displaying Event Replicator Destination Statistics

To display Event Replicator destination statistics :

1 On the Display Reptor Statistics screen, tab to the **Code** field and enter a D.

The Destination Statistics screen appears, listing the destination definitions that have been defined.

```
18:41:18          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator 1234          - Destination Statistics -                PRPTS20

Mark   Dest Name  Type   Output items                   Pending items
   -    --------  ------ --------------------------     -------------
   _    ADABAS1   Adabas                          0                 0
   _    DESTFILE  File                           0                 0
   _    DI040155  ETB                            0                 0
   _    DI042060  ETB                            0                 0
   _    DI042200  ETB                            0                 0
   _    DI046088  ETB                            0                 0
   _    DI062026  ETB                            0                 0
   _    DI062029  ETB                            0                 0
   _    DI062035  ETB                            0                 0
   _    DI062055  ETB                            0                 0
   _    DI062079  ETB                            0                 0
   _    DI062106  ETB                            0                 0
   _    DI062143  ETB                            0                 0
   _    DI064121  ETB                            0                 0
Select detail information by marking with 'S'

PF1----- PF2------ PF3------ PF4-------   PF7-------  PF8-------  PF12-----
Help               Exit      Refresh      -           +           Menu       ↩
```

2 Select a destination by typing an "S" in the **Mark** column for the corresponding destination and pressing Enter.

The Destination Statistics screen appears, showing detailed statistics for the selected destination.

```
12:17:49          ***** A D A B A S  BASIC  SERVICES *****          2013-04-07
Replicator 1234           - Destination Statistics -                PRPTS20

Destination name   : DI040155
Destination type   : ETBroker
Output items ..................                          0
Pending items .................                          0
Item time at source ........... N/A
Messages ......................                          0
 Bytes ........................                          0
Commits .......................                          0
Commit time at destination .... N/A
Pending messages ..............                          0
 Pending bytes ................                          0
De-logged from SLOG ...........                          0
Logged to SLOG ................                          0
Items on SLOG .................                          0


Press 'Enter' to continue

PF1----- PF2------ PF3------ PF4-------   PF7-------  PF8-------  PF12-----
Help               Exit                                        Menu        ↩
```

Displaying Event Replicator Input Queue Statistics

To display Event Replicator input queue statistics :

1 On the Display Reptor Statistics screen, tab to the **Code** field and enter a I.

The Input Queue Statistics screen appears, listing the input queue definitions that have been defined.

```
18:45:31          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator 1234          - Input Queue Statistics -                 PRPTS50

Mark    IQ Name    Input items
   -    --------   --------------------------
   _    ABCDEF1                           0
   _    ETBIN1                            0
   _    ETBQUEUE                          0
   _    E2                                0
   _
   _
   _
   _
   _
   _
   _
   _
   _
   _
Select detail information by marking with 'S'

PF1----- PF2------ PF3------ PF4-------   PF7-------  PF8-------  PF12-----
Help               Exit      Refresh                             Menu       ↩
```

2 Select an input queue by typing an "S" in the **Mark** column for the corresponding input queue and pressing Enter.

The Input Queue Statistics screen appears, showing detailed statistics for the selected input queue

```
12:18:56          ***** A D A B A S  BASIC  SERVICES *****          2013-04-07
Replicator 1234          - Input Queue Statistics -                 PRPTS50


Input Queue name: ETBIN1

Input items ..............                              0
Pending items ............                              0
Item time at source ........ N/A
Messages .................                              0
 Bytes ...................                              0
Pending messages ..........                             0
 Bytes ...................                              0
Commits ..................                              0
Commit time at input queue . N/A
Backouts .................                              0



Press 'Enter' to continue

PF1----- PF2------ PF3------ PF4-------   PF7-------  PF8-------  PF12-----
Help               Exit                                          Menu       ↩
```

Displaying Event Replicator Subscription Statistics

To display Event Replicator subscription statistics:

1 On the Display Reptor Statistics screen, tab to the **Code** field and enter a D.

The Subscription Statistics screen appears listing the subscription definitions that have been defined.

```
18:42:53          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator 1234          - Subscription Statistics -                PRPTS30

Mark   Subscription  Output items
       Name
   -   --------      --------------------------
   _   SADABAS1                             0
   _   SI040155                             0
   _   SI042060                             0
   _   SI042200                             0
   _   SI046088                             0
   _   SI062026                             0
   _   SI062029                             0
   _   SI062035                             0
   _   SI062055                             0
   _   SI062079                             0
   _   SI062106                             0
   _   SI062143                             0
   _   SI064121                             0   Select detail information
   _   SI120248                             0   by marking with 'S'

PF1----- PF2------ PF3------ PF4-------   PF7-------  PF8-------  PF12-----
Help               Exit      Refresh      -           +           Menu       ↩
```

2 Select a subscription by typing an "S" in the **Mark** column for the corresponding subscription and pressing Enter.

The Subscription Statistics screen appears, showing detailed statistics for the subscription you selected.

```
18:42:53          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator 1234          - Subscription Statistics -                PRPTS30

Subscription name: SADABAS1
Input items ..............                          0
Output items .............                          0
 C5 data .................                          0
 Data lost ...............                          0
 Initial-state completed ..                         0
 Initial-state data .......                         0
 Security functions .......                         0
 User transactions ........                         0
 Utility functions ........                         0
Item time at source ....... N/A
Item processed ............ N/A                     Press 'Enter' to continue
                                                    Displaying 1   of 3
DBID File ................. 1955  1
Inserts .                          0 Initial state.                          0
Updates .                          0 Filtered out .                          0
Deletes .                          0

PF1----- PF2------ PF3------ PF4-------   PF7-------  PF8-------  PF12-----
Help               Exit                                                    ↩
↩
```

Displaying Event Replicator Output Subtask Statistics

To display Event Replicator output subtask statistics :

- On the Display Reptor Statistics screen, tab to the **Code** field and enter a S.

  The Output Subtask Statistics appears, showing Event Replicator statistics.

```
18:43:49          ***** A D A B A S  BASIC  SERVICES *****          2013-04-02
Replicator 1234         - Output Subtask Statistics -               PRPTS04

Subtask  Output items                      Subtask  Output items
----     --------------------------        ----     --------------------------
Main                              0
  1                               1
  2                               1
  3                               0
  4                               0

                                                      Press 'Enter' for messages

PF1----- PF2------ PF3------ PF4-------   PF7-------  PF8-------  PF12-----
Help               Exit      Refresh                             Menu        ↩
```

The total number of replicated transactions for each subtask are listed.

Running the RPLCHECK Command

You can run the **RPLCHECK command** from AOS to perform the replication cross-check function for all active databases known to the Event Replicator Server. For complete information, read *RPLCHECK Command*, elsewhere in this guide.

To run the RPLCHECK command :

1 Select H on the Replicator Management screen.

A RPLCheck popup screen appears.

```
               - RPLCheck -


Press Enter to perform a RPLCheck.




PF3---
End
                                                  ↩
```

2 Press Enter to run RPLCHECK or PF3 to quit out of the popup screen without running RPLCHECK.

Running the RPLCLEANUP Command

You can run the **RPLCLEANUP command** from AOS to stop replay processing if the Replay Utility (ADARPL) abends. For complete information, read ***RPLCLEANUP Command***, elsewhere in this guide.

To run the RPLCLEANUP command :

1 Select L on the Replicator Management screen.

A RPLCleanup popup screen appears.

```
            - RPLCleanup -


Press Enter to perform a RPLCleanup.
Specify Token number as required.

Token .......... _____

PF3---
End
                                                  ↩
```

2 Specify a token number on the popup screen and press Enter to run RPLCLEANUP or press PF3 to quit out of the popup screen without running RPLCLEANUP.

Running the RPLREFRESH Command

You can run the **RPLREFRESH command** from AOS to refresh resource definitions in your Event Replicator Server configuration while the Event Replicator Server is running or to abort scheduled (pending) RPLREFRESH processing. For complete information, read *RPLCLEANUP Command*, elsewhere in this guide.

To run the RPLREFRESH command :

1 Select P on the Replicator Management screen.

A RPLRefresh popup screen appears.

```
              - RPLRefresh -


           1 - Full refresh
           2 - Global resources
           3 - Subscription
           4 - Destination
           5 - Initial state
           6 - Input queue
           7 - Resend buffer
           8 - Abort refresh
           --------------------
Type ....
Name ....

Select 'Type' of RPLRefresh required
and enter Resource Name if required.
Press Enter.

PF3---
End
```

2 In the **Type** field, specify the number of the menu item that identifies the type of refresh you want to occur. Specifying **1** refreshes all resources; specifying **2** refreshes all global resources; specifying **8** aborts a scheduled resource refresh attempt. All of the other menu options specify that an individual resource be refreshed.

3 If you selected menu item **3**, **4**, **5**, **6**, or **7** in the **Type** field, you must also specify the name of the resource you wish to refresh in the **Name** field.

This step should be skipped if you selected menu items **1**, **2**, or **8** in the "Type" field.

4 When values for the "Type" and "Name" field have been specified, press Enter to start refresh processing.

Refresh processing starts. Be sure to check the console messages to determine when refresh processing completes and for possible refresh processing errors. Even if refresh processing appears to have occurred correctly in AOS, the console messages should be reviewed to verify that refresh processing occurred without error.

Accessing the Adabas Event Replicator Subsystem from AOS

You can access the Adabas Event Replicator Subsystem from AOS. Using the Adabas Event Replicator Subsystem, you can create, modify, and delete replication definitions in the Event Replicator Server. For more information about maintaining replication definitions, read *Replication Definition Overview and Maintenance*, elsewhere in this guide.

To access the Adabas Event Replicator Subsystem from AOS:

- Select R on the Replicator Management screen.

  The Main Menu of the Adabas Event Replicator Subsystem appears.

```
16:00:57      ***** A D A B A S  EVENT REPLICATOR SUBSYSTEM *****    2010-05-28
Vers 3.3.1                         Main Menu                          M-RP0010
DBID 1234  File 89

                  Code    Function
                  ----    --------------------------------
                    A     Administrator Functions
                    D     Destination Definitions
                    F     Transaction Filter Definitions
                    G     Global Format Buffer Definitions
                    I     Initial-State Definitions
                    Q     Input Queue Definitions
                    R     Resend Buffer Definitions
                    S     Subscription Definitions
                    ?     Help
                    .     Exit
                  ----    --------------------------------

           Code ... _

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit
```

Read *About the Adabas Event Replicator Subsystem*, in *Adabas Event Replicator Subsystem User's Guide*, for more information.

Index

F

G

I

L

M

N

O

P

R

S

T

U

W