



AMDOCS CONVERGENT CHARGING
RELEASE 8.1

Turbo Charging 8.1

Run Book

Document Information

Release: **8.1**
Publication Date: **November 2011; updated August 2015 for FB 8.1.2.30**
Catalog Number: **1779788**
Information Security: **Level 1 - Confidential**

Copyright © 2015 Amdocs. All Rights Reserved. Reproduction or distribution other than for intended purposes is prohibited, without the prior written consent of Amdocs. The trademarks and service marks of Amdocs, including the Amdocs mark and logo, Intentional Customer Experience, CES, Clarify, Ensemble, Enabler, Return on Relationship, Intelecable, Collabrent, XACCT, DST Innovis, Stibo Graphic Software, Qpass, Cramer, SigValue, JacobsRimell, ChangingWorlds, jNetX, OpenMarket Inc., MX Telecom Inc., MX Telecom Ltd, Streamezzo, and Bridgewater Systems are the exclusive property of Amdocs, and may not be used without permission. All other marks are the property of their respective owners.

Table of Contents

1	Introduction.....	1
2	ADJ1ICWFlow – Implementation Compiler Workflow	19
3	ADJ1EVENTSRV – Event Server.....	72
4	ADJ1FILE2ESRV – File2E	181
5	ADJ1UH – Update Handler in Full and Incremental Mode.....	208
6	ADJ1UH – Update Handler in Synchronous Mode	232
7	ADJ1UHMARK4RE – Update Handler in Mark for Rerate Mode.....	244
8	ADJ1RCN – Update Handler in Reconciliation Mode	249
9	ADJ1UHCUG – Update Handler in CUG Mode	265
10	ADJ1UHMINFULL – Update Handler in Mini-Full Mode	268
11	ADJ1DB2E – DB2E.....	271
12	ADJ1GAT2ESRV – GAT2E.....	320
13	ADJ1RRPSRV – Rerate Prepare.....	334
14	ADJ1RRPOPREP – Rerate Population Report	356
15	ADJ1RRPEOC – Rerate Prepare for End of Cycle	367
16	ADJ1DISPENTFR – Rerate Prepare for End of Cycle Finish Notification.....	387
17	ADJ1ERRREP – Rerate Error Report.....	390
18	ADJ1CLNBLACK – Blacklist Table Clean-up	401
19	ADJ1RER – Rejected Event Recycler	404
20	ADJ1NTFSRV – Notification.....	420
21	ADJ1DISPSRV – Dispatcher.....	448
22	ADJ1DISPEOC – Dispatcher for End of Cycle	482
23	ADJ1ELASRV – ELA Collector	492
24	ADJ1RORC – Read-Only Rater Client.....	505
25	ADJ1UQPROXY – Usage Query Proxy	519

26	ADJ1UQSRV – Usage Query Server	529
27	ADJ1SETDBACT – Set Database Activities	548
28	ADJ1CEEXT – Cycle Events Extract.....	552
29	ADJ1CPEXT – Cycle Accumulators Extract.....	566
30	ADJ1CAEXT – Cycle Allowances Extract.....	580
31	ADJ1OCEEXT – Optimized Customer Group Event Extract	594
32	ADJ1POCEEXT – Optimized Customer Group Event Extract Post Rerate	612
33	ADJ1RCPEXT – Customer Group Accumulators Extract.....	618
34	ADJ1PRCPEXT – Customer Group Accumulators Extract Post Rerate.....	636
35	ADJ1RCAEXT – Customer Group Allowances Extract.....	641
36	ADJ1PRCAEXT – Customer Group Allowances Extract Post Rerate.....	659
37	ADJ1BODEXT – Bill on Demand Extract	664
38	ADJ1EXTCLEANE – Extract Clean-up	679
39	ADJ1OUTCOL – Outcollect Prepare	683
40	ADJ1_APB_CycleBillRun_Sh – Turbo Charging Cycle Bill Run.....	697
41	ADJ1MAPCREAT – Create Map	701
42	ADJ1ACGRPLSTN – Accumulator Group Listener.....	705
43	ADJ1EVGRPLSTN – Event Group Listener	709
44	ADJ1PEFNSHNTF – Accumulator Extract Finish Notification	713
45	ADJ1CYCLESTAT – Cycle State	716
46	ADJ1CYCLEMNT – Cycle Maintenance	721
47	ADJ1CYCMNTEOD – Cycle Maintenance EOD	731
48	ADJ1UROPS – Update Refresh at Standby Site	736
49	ADJ1_ConfirmCycle_sh – Turbo Charging Confirm Cycle	739
50	ADJ1_EOC_USG_CTRL_Sh – EOC Usage Control.....	742
51	ADJ1TRUNC – EOD Truncate Usage	748
52	ADJ1COMPUSG – Compress Usage	752

53	ADJ1MOVEUSG – Move Usage	755
54	ADJ1TRNCUSG – Truncate Usage.....	758
55	ADJ1ROLLACCUM – Roll Accumulators.....	761
56	ADJ1CRA – Copy Rolling Accumulators.....	776
57	ADJ1CYCMEMCLN – Cycle Memory Clean-up.....	784
58	ADJ1CPCYCUSG – Copy Cycle Usage.....	789
59	ADJ1CCUCU – Copy Cycle Usage Clean-up.....	804
60	ADJ1SETENVAR – Set Environment Variables.....	811
61	ADJ1IC_DependCheck – Dependency Checker for Implementation Compiler Libraries	816
62	UTL1MULTILEXT– Multiple Library Extract.....	821
63	UTL1LIBEXT – Library Extract	829
64	UTL1POSTLIBEX – Library Post-Extract.....	836
65	memdbdump – AIMOS Dump.....	840
66	metainfo – AIMOS MetalInfo.....	847
67	ERT – Error Entity Recovery Tool.....	859
68	High Availability	880
	Appendix A Reconciliation Requests	889
	Appendix B Configuration Parameter Mechanism.....	903



1 Introduction

An event occurs whenever a customer makes use of any communications service provider (CSP) services, such as making a cellular or wireline phone call, transferring data, sending a message, connecting to the Internet, or purchasing content via the Internet.

Turbo Charging provides full event processing capabilities that can be employed to manage subscriber events across various lines of business.

Target Audience

This document is intended for the following teams:

- Testing
- Delivery accounts
- Data Center operators
- Infra experts

Scope of This Document

This document describes the batch jobs, daemons, and scripts of Turbo Charging.

Process Information

Each process is described in a separate chapter. The following information is provided if relevant for the particular process.

Subject	Description
Name	Both the Operational job name and full name of the process (this is also the name of the chapter).
Description	A description of the process, including its purpose, what it does, and the results of a successful run. This section also includes the process type and its run frequency.
Participation in Generic Maps	A description of the generic maps in which the process participates.
Activation and Shutdown	The command line, script, and executable that activate and shut down the process, and the utility used, such as Amdocs Monitoring & Control.
Preceding and Dependent Processes	A list of processes that must run before the activation of the process (preceding processes) and the list of processes that can be run only after its completion (dependent processes).
Affected Applications	A list of applications that are affected by the process results.
Log, Input, and Output Files	The name, location, format, and detailed content of all files.
Process Flow	The process flow step by step.
Environment Variables	The environment variables that impact on the process results, including their default values and the method for changing the values if necessary.
Parameters	The parameters that affect the process results, including their default values.
Configuration Parameters	The configuration and fine-tuning parameters of the process.
Configuration Files	The names of configuration files that affect the process results.
Database Connections	A list of database tables to which the process connects during processing.
Admin Commands	A list of admin commands that the process accepts.
Distribution	The distribution method for the process results.
Screen Messages	The messages that appear on user screens indicating a failure, a warning, or a successful run.

Subject	Description
Troubleshooting	Common error messages with brief solutions.
Recovery Instructions	Instructions for the operator on how to rerun the process if it fails, or if the whole system fails.
Important Remarks	Any additional relevant information for the process that is not included in other sections.

Languages

The programming languages in which the processes are written are:

- Java
- C++

Input and Output Directories

The input and output files are placed in the following permanent directories:

- Interface input files are placed in \$ABP_APP_ROOT/interfaces/input.
- Interface output files are placed in \$ABP_APP_ROOT/interfaces/output.

Log and Trace Files

This section describes the naming conventions used for logs and traces.

Log Files

This section describes log files.

C++ Processes

All Turbo Charging processes written in C++ use the Logger mechanism of Amdocs C++ Foundation.

Logs of Turbo Charging processes written in C++ can be enabled or disabled by setting the ADJ.Logger.Enable parameter in the ADJ1_CONF_SECTION_PARAM table. The default value of this parameter is the ADJ_LOGGER_ENABLE environment variable defined in the ADJ1_PROC_PARAMS table.

Log severity can be set using the ADJ.Logger.Severity parameter in the ADJ1_CONF_SECTION_PARAM table. The default value of this parameter is the ADJ_LOGGER_SEVERITY environment variable defined in the ADJ1_PROC_PARAMS table.

For more information about these configuration parameters and environment variables, see the corresponding sections in the “ADJ1EVENTSRV – Event Server” chapter.

Turbo Charging processes written in C++ use the following naming conventions for logs:

- *Log files:*

`${GN1_TASK_NAME}_${APPLICATION_ID}_%D_%T_%n.log`

where:

- `GN1_TASK_NAME` is the job's name.
- `APPLICATION_ID` is the process instance ID (for example, ES100).
- `D` is the current date.
- `T` is the current time.
- `n` is the serial number of the file.

- *Console logs:*

`<Inner script name>_${APPLICATION_ID}_<Date>_<Time>.log`

- *Operational logs:*

`${GN1_TASK_NAME}_${APPLICATION_ID}_${ABP_MARKET}_<Date>_<Time>.log`

The names and locations of log files can be changed via the configuration parameters of the Logger mechanism in the ADJ1_CONF_SECTION_PARAM table. For more information, see *C++ Foundation Programmer Guide*.

Java Processes

Logs of Turbo Charging Java processes cannot be disabled.

Log severity can be set using the log4j environment variables in the .w.ini.pre.env file in the home directory. After making any changes in the file, the operator must save it, open a new session for the same environment, and run the process in the new session. For more information about the log4j variables, see the “Environment Variables” sections in the chapters describing Java processes.

Turbo Charging processes written in Java use the following naming convention for logs:

`${GN1_TASK_NAME}_<Time Stamp>.log`

Trace Files

Trace files capture low-level trace information in a text format. Tracing can be dynamically turned on or off and is required for special investigations.

In non-production environments, all Turbo Charging processes written in C++ use the Logger mechanism of Amdocs C++ Foundation.

In the out-of-the-box implementation (defined in the ADJ1_CONF_SECTION_PARAM table):

- Severity of traces is set to 0 (traces are active).
- The location and naming convention for traces are as follows:
 `${ABP_APLOG}/ ${GN1_TASK_NAME}_TRACE_${APPLICATION_ID}_%
D_%T_%n.log`

Both definitions can be changed using the rules of the Logger mechanism. For more information, see *C++ Foundation Programmer Guide*.

There are no trace files for processes written in Java.

Light Tracer Files

The Light Tracer is the mechanism that replaces the tracing functionality of the Logger in a production environment. Both mechanisms keep working in parallel while all the tracing activities are performed by the Light Tracer. The Light Tracer does *not* replace the regular logging functionality.

Light Tracer files use the following naming convention:

`<Application Name>_<Instance Name>_<Process ID>_<Date>_<Time>[_<Sequence Number>].trace|toe`

Example:

`TC_ES100_24795_20110406_075919.trace`

where:

- *Application Name* – The name of the application, such as ‘TC’.
- *Instance Name* – The process instance as defined in the PROCESS_INSTANCE_ID (or sometimes PROCESS_EXEC_ID) field in the routing (GN1_SYS_) tables of the Availability Manager. This instance name equals the value of the APPLICATION_ID parameter in the start-up command line for each process.
- *Process ID* – The ID of the process in the operating system.
- *Date* – The date on which the file was written.
- *Time* – The time at which the file was written.
- *Sequence Number* – An optional sequence number that is used when trace messages are divided into several files. It is added to the end of the file name for every part after the first one.
- *trace* – The extension for regular trace files.
- *toe* – The extension for trace per event (flow tracing) files.

The Light Tracer produces files in binary format. The Light Tracer formatting utility reads the binary files and converts them into readable output files in the ASCII (text) format. For more information about this utility, see *Amdocs Billing Utilities Run Book*.

The binary file has the following structure:

- *File header* – Sixty-four bits that indicate the endianness used, followed by sixty-four bits that indicate the version of the Light Tracer (for compatibility issues)
- *Flow trace beginning indicator* – Only for .toe files
- *Traces*
 - *Header* – Sixty-four bits that start every trace. There are two header types (depending on a specific bit in the header):
 - *Internal message of the Light Tracer* – The number of traces that the Light Tracer received but did not write to the log because the buffer was full
 - *Regular trace header*
 - ◆ Module ID
 - ◆ Submodule ID
 - ◆ File ID
 - ◆ Message sequence number
 - ◆ Line number of the trace in the source code file
 - ◆ The number of parameters
 - ◆ Indicators

Optional header extensions include:

- ◆ *Subheader* – The size of large-scale, dynamic parameters (such as dynamic string) or indicators related to dynamic parameters (such as whether the message is a log)
- ◆ *Time stamp*
- ◆ *Thread ID*
- ◆ *Context ID*
- *Parameters* – Variables included in the trace message whose type is known but whose value is not known before run time, for example, the session_id parameter in the following message: Starting session <session_id>.
- *Flow trace ending indicator* – Only for .toe files
- *File ending indicator* – An indicator that shows that the Light Tracer was shut down properly

Information that is written into Light Tracer files includes:

- Traces
- Traces per event (flow traces)

- Traces of SQL queries, including their bind variables
- Log messages with the severity of INFO and DEBUG, if the merging of log and trace messages is enabled, including the following logs:
 - File2E event log
 - Dispatcher event log
 - Various statistics, such as the statistics of the Persistence Writer

Connection Parameters

The following connection parameters are used to connect the component's processes to various resources (application database, security database, URLs, ports, and so on):

- *APPL_DB_USER* – Amdocs Billing Customer Manager Applicative database user name
- *APPL_DB_PASSWORD* – Amdocs Billing Customer Manager Applicative database password
- *APPL_DB_INSTANCE* – Amdocs Billing Customer Manager Applicative database instance
- *APPL_DB_HOST* – Amdocs Billing Customer Manager Applicative database host
- *APPL_DB_TYPE* – Amdocs Billing Customer Manager Applicative database type ('ORA')
- *ADJAPPL_DB_USER* – Applicative Usage database user name
- *ADJAPPL_DB_PASSWORD* – Applicative Usage database password
- *ADJAPPL_DB_INSTANCE* – Applicative Usage database instance
- *ADJAPPL_DB_HOST* – Applicative Usage database host
- *ADJAPPL_DB_TYPE* – Applicative Usage database type ('ORA')
- *ADJREF_DB_USER* – Reference database user name
- *ADJREF_DB_PASSWORD* – Reference database password
- *ADJREF_DB_INSTANCE* – Reference database instance
- *ADJREF_DB_HOST* – Reference database host
- *ADJREF_DB_TYPE* – Reference database type ('ORA')
- *ADJAC_DB_USER* – Audit & Control database user name
- *ADJAC_DB_PASSWORD* – Audit & Control database password
- *ADJAC_DB_INSTANCE* – Audit & Control database instance
- *ADJCONFIG_DB_USER* – Configuration database user name
- *ADJCONFIG_DB_PASSWORD* – Configuration database password
- *ADJCONFIG_DB_INSTANCE* – Configuration database instance

- *ADJCONFIG_DB_HOST* – Configuration database host
- *ADJCONFIG_DB_TYPE* – Configuration database type ('ORA')
- *ADJCUSTUSG..._DB_USER* – Customer Usage database user name
- *ADJCUSTUSG..._DB_PASSWORD* – Customer Usage database password
- *ADJCUSTUSG..._DB_INSTANCE* – Customer Usage database instance
- *ADJCUSTUSG..._DB_HOST* – Customer Usage database host
- *ADJCUSTUSG..._DB_TYPE* – Customer Usage database type
- *ADJOP_DB_USER* – Operational database user name
- *ADJOP_DB_PASSWORD* – Operational database password
- *ADJOP_DB_INSTANCE* – Operational database instance
- *ADJOP_DB_HOST* – Operational database host
- *ADJOP_DB_TYPE* – Operational database type ('ORA')
- *ADJRESOURCE_DB_USER* – Resource Usage database user name
- *ADJRESOURCE_DB_PASSWORD* – Resource Usage database password
- *ADJRESOURCE_DB_INSTANCE* – Resource Usage database instance
- *ADJRESOURCE_DB_HOST* – Resource Usage database host
- *ADJRESOURCE_DB_TYPE* – Resource Usage database type ('ORA')
- *ADJUSG..._DB_USER* – Usage database user name
- *ADJUSG..._DB_PASSWORD* – Usage database password
- *ADJUSG..._DB_INSTANCE* – Usage database instance
- *ADJUSG..._DB_HOST* – Usage database host
- *ADJUSG..._DB_TYPE* – Usage database type
- *ADJTRBCL_DB_USER* – Transaction Broker Client database user name
- *ADJTRBCL_DB_PASSWORD* – Transaction Broker Client database password
- *ADJTRBCL_DB_INSTANCE* – Transaction Broker Client database instance
- *ADJTRBCL_DB_HOST* – Transaction Broker Client database host
- *ADJTRBCL_DB_TYPE* – Transaction Broker Client database type ('ORA')
- *ADJTRBPB_DB_USER* – Transaction Broker Publisher database user name
- *ADJTRBPB_DB_PASSWORD* – Transaction Broker Publisher database password
- *ADJTRBPB_DB_INSTANCE* – Transaction Broker Publisher database instance

- *ADJTRPB_DB_HOST* – Transaction Broker Publisher database host
- *ADJTRPB_DB_TYPE* – Transaction Broker Publisher database type ('ORA')



Note: Connect codes for Usage databases (both home and alternate) are specified in the ADJ1_CYCLES table. These connect codes must be also defined at least for the ADJ1EVENTSRV task name in the GN1_TASK_CONNECT table.

Connection for Session-Based Guiding

By default, the APR1_SESSION_GUIDING table, which is used for session-based guiding, is located in the applicative database.

If this table is in a different database, a new record must be added to the ADJ1_POLICY_CFG table with the following data:

- POLICY_CODE = 'SBG_DB_CONN_CODE'
- POLICY_VALUE = <Prefix of the environment variables for the database connection>

For example, if the POLICY_VALUE field for the SBG_DB_CONN_CODE is set to 'ADJSBG', the system looks for the following environment variables:

- ADJSBG_DB_USER
- ADJSBG_DB_PASSWORD
- ADJSBG_DB_INSTANCE

General Troubleshooting

The following shows how to troubleshoot various errors.

I/O Errors

If a process is aborted because it fails to open, read, write, or close any file, analyze and handle the status that appears in the error message. For example, if the disk quota is exceeded, clear disk space, and rerun the job.

If the problem cannot be solved, contact the software component representative.

Database Access Errors

Contact the Database Administrator if the aborted process displays error messages such as the following:

- Maximum number of processes exceeded
- Failed to connect to Oracle
- Table or view does not exist

Missing or Invalid Data Errors

Contact the software component representative if the aborted job displays errors messages such as the following:

- No relevant data
- Invalid data
- Key not found

Other Errors

If the error cannot be classified as belonging to one of the above groups, contact the software component representative.

Debugging

The following environment variables enable debugging, if they are defined before the process activation script is run.

Name	Description	Default Value	Method of Changing the Variable
ADJ1_DEBUGGER	The name of the debugger	None	export ADJ1_DEBUGGER=<debuggerName> <i>Example:</i> export ADJ1_DEBUGGER=wdb
DISPLAY	The display of the debugger	None	export DISPLAY=<The user's display>

Environment Variables

Turbo Charging processes use the following environment variables.

Name	Description	Location	Default Value
ABP_ADJ_ROOT	The root directory for Turbo Charging	\${ABP_BIN}/w.ini	\$HOME/var/m3g/projs/adj
ABP_APE_ROOT	The root directory for the Rating Engine	\${ABP_BIN}/w.ini	\$HOME/var/m3g/projs/ape
ABP_APB_ROOT	The root directory for Turbo Charging processes	\${ABP_BIN}/w.ini	\$HOME/var/m3g/projs/apr

In addition, the \${RTPRIO} environment variable determines the runtime priority of every Turbo Charging process. It is used in the general ADJ1_Process_Run_Sh.sh script that is involved in process activation, as part of the exe command (\$RTPRIO exe <parameters>).

This environment variable has a considerable effect on performance. Therefore, it is important to ensure that the Event Server is given the highest priority of all Turbo Charging processes. For example, the priorities can be set as follows:

- All File2E instances – 110
- All Event Server instances – 100

Shared Memory Size

The MEMORY_SIZE_IN_BYTES parameter in the APR1_CONF_SECTION_PARAM table is used to define the requested size of the shared memory. It is important to configure this parameter according to the processing requirements of the system.

Because the Event Server requires more memory than other processes, this parameter has two layers:

- *For the Event Server* – The PARAM_CLASS is ‘ES’. For more information, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.
- *For other processes* – The PARAM_CLASS is ‘APR’. Any change to this parameter affects the size of the shared memory of all the processes that connect to the APR1_CONF_SECTION_PARAM table.

Required Resources and Services

For a complete list of third-party software, public software, and Amdocs tools and utilities, see *Amdocs Billing Server Pre-Installation Guide*.

Scripts

This section describes the script naming conventions for Turbo Charging processes. It also explains how to work with the op_adj_env_sh script.

Script Naming Conventions

Turbo Charging processes use the following script naming conventions.

- For each daemon process:
 - ADJ1_<Unique process name>_Daemon_Shell_Sh – This script is called from the UNIX command line or Amdocs Monitoring & Control/Availability Manager.
 - ADJ1_<Unique process name>_Daemon_Sh – This script is in the OPPROG table and is called by RunJobs. It enables the operator to run daemons as jobs in Amdocs Batch Job Manager.
 - ADJ1_<Unique process name>_Daemon_inner_Sh – This script is called internally by each of the two scripts above.
- For each job process:
 - ADJ1_<Unique process name>_Job_Shell_Sh – This script is called from the UNIX command line or Amdocs Monitoring & Control/Availability Manager.
 - ADJ1_<Unique process name>_Job_Sh – This script is in the OPPROG table and is called by RunJobs.
 - ADJ1_<Unique process name>_Job_inner_Sh – This script is called internally by each of the two scripts above.

Amdocs Batch Job Manager Application Script

This section describes the op_adj_env_sh script of Amdocs Batch Job Manager.

Description

The op_adj_env_sh script of Amdocs Batch Job Manager is called in either of the following ways:

- By the ADJ1_<Unique process name>_Daemon_Shell_Sh or ADJ1_<Unique process name>_Job_Shell_Sh script, when the flow is triggered from Amdocs Monitoring & Control or the command line
- Through Amdocs Batch Job Manager (RunJobs), when the flow is triggered from Amdocs Batch Job Manager or the command line

The op_adj_env_sh script initializes the environment variables that each process uses at runtime (these variables appear in the .ccf configuration file). The variables are initialized in the init files of the process in the order in which they are passed to the script.



Note: As there can be a core op_adj_env_sh file and a number of customization op_adj<n>_env_sh files, there is an order in which these files are activated. This order determines the initialization order for the variables. For more information, see the “Initialization Order” section in this chapter.

Customization Scripts

The core op_adj_env_sh can be customized. While the script has no number, each customized script has a sequence number after _adj, for example, op_adj9_env_sh.

To execute a customization op_adj<n>_env_sh script, perform one of the following sample procedures:

- Procedure 1:
 - a. Copy the core script to the \$ABP_PRIVATE_BIN directory.
 - b. Change the name of the core script to op_adj<n>_env_sh.
 - c. Modify the original core script, and save the changes.
- Procedure 2
 - a. Create a customized op_adj<n>_env_sh script.
 - b. Place the customized script in one of the following locations:
 - \$ABP_CUSTOM_BIN/op_adj<n>_env_sh
 - \$ABP_ETC/ op_adj<n>_env_sh
 - \$ABP_PRIVATE_ETC/ op_adj<n>_env_sh

Initialization Order

The initialization order for the core and customization scripts is as follows:

- Core layer:
 - a. \$ABP_BIN/op_adj_env_sh
 - b. \$ABP_CUSTOM_BIN/op_adj_env_sh
 - c. \$ABP_ETC/op_adj_env_sh
 - d. \$ABP_PRIVATE_ETC/op_adj_env_sh
- Each subsequent layer in the same sequence (where <n> is the layer number, such as 2, 3, 4, 5, 6, 7, 8, 9, and so on):
 - a. \$ABP_BIN/op_adj<n>_env_sh
 - b. \$ABP_CUSTOM_BIN/op_adj<n>_env_sh
 - c. \$ABP_ETC/op_adj<n>_env_sh
 - d. \$ABP_PRIVATE_ETC/op_adj<n>_env_sh

Because the customization script is executed after the core script, it can override core definitions.

Process Start-up Orchestration

The operator can use the Command Map mechanism of the Availability Manager to configure a set of commands for activating a list of processes in a particular, predefined order.

To do so, the operator must perform the following:

1. Define a new map in the GN1_SYS_COMMAND_MAP table. The entries in this table indicate to the Availability Manager that a specific command is not a regular admin command but is included in a set of commands.
2. Define the set of commands for starting the processes in the GN1_SYS_COMMAND_MAP_DETAILS table.
3. Define the start-up order by creating dependencies between the commands in the GN1_SYS_COMMAND_DEPENDENCIES table.

For more information about these tables, see *Availability Manager High Availability Business Parameter Configuration Guide*.

After configuring the Command Map tables and bouncing both Availability Managers, the operator can start the processes in one click via Amdocs Monitoring & Control. For more information, see *Amdocs Billing Operator Guide*.

In addition, this mechanism can be used for planned maintenance activities, such as refreshing libraries, upgrading the software, and upgrading the hardware. For more information, see *Turbo Charging High Availability Maintenance Guide*.

Related Documents

- *Amdocs Billing Availability Manager Data Model*
- *Amdocs Billing Availability Manager Run Book*
- *Amdocs Billing Introduction*
- *Amdocs Billing Introduction*
- *Amdocs Billing Operator Guide*
- *Amdocs Billing Server Pre-Installation Guide*
- *Amdocs Billing Utilities Run Book*
- *Amdocs Invoicing Run Book*
- *Amdocs Service Platform Operation Manual*
- *Audit & Control Run Book*
- *Availability Manager High Availability Business Parameters Configuration Guide*
- *C++ Foundation Programmer Guide*
- *Rating Logic Configurator Implementation Best Practices*
- *Rating Logic Configurator User Guide*
- *Turbo Charging Architecture*
- *Turbo Charging Configurator User Guide*
- *Turbo Charging Data Model*
- *Turbo Charging Extension Functions Programmer Guide*
- *Turbo Charging High Availability Maintenance Guide*
- *Turbo Charging Implementation Compiler XML Configuration Guide*
- *Turbo Charging Reconciliation Tool User Guide*
- *Turbo Charging Update Handler Implementation Best Practices*
- *Turbo Charging Update Handler XML Configuration Guide*

Terms and Definitions

The following specialized terms are used in this document.



Note: For a complete list of terms, see Amdocs Billing Glossary.

Term	Definition
accumulator	A state container used for a variety of purposes, such as storing usage, balances, limits, usage rights, and so forth. The system relates an accumulator to a specific cycle instance and when required, copies the accumulator data from one cycle instance to another according to the relevant initiation rules defined in implementation. (Formerly known as performance indicator or PI.)
agreement	In the customer model, each organizational unit may have an agreement. Each agreement can contain a set of offers. A subscriber is related to a unit in the customer model, and receives, by default, all the offers that are assigned to one of the agreements belonging to the units above it in the organizational structure.
AIMOS	Amdocs in-memory object storage. Embedded storage that caches the data of a persistent database, such as Oracle and others (it is not limited to relational databases). It caches the data in customized object data structures that best suit the application, to achieve high performance, low latency, high efficiency, and in-memory survivability.
Amdocs Acquisition & Formatting	An optional module whose function is to collect the incoming event records and format them for use by other modules for guiding the records to the appropriate customer. Amdocs Acquisition & Formatting collects the Event Detail Records (EDRs) from the network switches and processes them.
Amdocs Monitoring & Control	A tool for monitoring and manipulating Amdocs applications, third-party tools, and network elements. It provides centralized control of all required applications while increasing application availability via a simple generic GUI. Amdocs Monitoring & Control is provided with an extensive out-of-the-box implementation (in the form of logical applications) that supports the Amdocs Billing processes and daemons.
Availability Manager	The Availability Manager (AVM) provides the mechanism supporting high availability. The Availability Manager has the responsibility of making decisions about failovers. Such decisions are based on at least a cross-check between data gathered from all critical processes in the system. The Availability Manager senses a server failure by sending ICMP echo requests to the server IP.
CSP	Communication service provider.
customer	The main entity with which the communication service provider conducts its business. A customer may be an individual or a company. A customer may have several subscribers and several global agreements for all or some of its subscribers. In addition, a customer may possess several accounts organizing its subscribers in a specific financial structure.
customer group	A range of customer segments.

Term	Definition
customer segment	A logical collection of customers. The customer segment is a number resulting from a function applied to a customer ID. Such division of customers into segments enhances performance.
cycle	A cyclic period with a frequency and a close day. A cycle is identified by a cycle code. A population's bill cycle means the period (usually one month long) between bills for that population (cycle population).
Cycle Maintenance	A Turbo Charging process that does the following: <ul style="list-style-type: none"> ■ Receives commands from external sources pertaining to changes in the cycle area ■ Updates the Turbo Charging Cycle States table ■ Informs all relevant parties of the new state of the Turbo Charging Cycle States table
DB2E	Database to Event Server. A Turbo Charging process that extracts data from the Turbo Charging Customer database and sends the data as events to Event Server processes.
event	Any use of the network by a customer. Events can originate from various sources, including wireless circuit switches, packet data switches, content provider servers, Internet, and fixed-net elements, and standard inputs from clearinghouses. Internal events, including changing parameter values, can originate from other system modules.
Event Interface Module	A part of the Event Server process that receives requests through sockets and sends responses to these requests. The Event Interface Module also performs routing of resource data.
Event Processing Module	A part of the Event Server process that executes the main body of processing, including session management, rating, and balance management.
Event Server	A Turbo Charging process that groups all functions to process usage and other events using data from the Turbo Charging Customer and Usage databases. Some of these functions are active, while others are idle. The set of active functions within the process defines its logical identity.
event type	The name of the structure of an event data record.
File2E	File to Event Server. A Turbo Charging process that extracts event data from the files prepared by mediation and then sends the events contained in these files to the relevant Event Servers.
frame event	The envelope containing all the information coming from the network event, consisting of several smaller services, according to the number of MSCCs handled by the primary session (or their equivalent in other protocols). This event represents the primary session, as defined by the PDP context and the session ID.
Geo Redundancy	A configuration of service sites, set up in pairs, at geographically separate locations. Each geo-redundant pair consists of an active and a standby site. In the event of failure of the active site, the standby site takes over with a minimum of service downtime.
Guiding to Customer Module	A part of the Event Server process that performs guiding to customer according to resource data.

Term	Definition
Implementation Compiler	A Turbo Charging process that is responsible for converting the data that is created and maintained in the Rating Logic Configurator into a set of compiled libraries that execute the rating logic and support the communication with the network by means of messages adhering to the Diameter protocol in their structure.
Implementation Repository	A subsystem of the Rating Logic Configurator used by technical personnel to maintain entities that serve as building blocks for pricing packages, service packages, discount and penalty packages, and offers. In essence, the Implementation Repository is a container for the structural definitions of business object types, together with the business rules that define all possible pricing item types and qualification criteria. In addition, the Implementation Repository can be used to define the mapping between the messages in the Diameter protocol and Rating Logic Configurator (PC) events.
MSCC	Multiple Services Credit Control. One of the functions supported by the 3GPP Diameter protocol.
notification	A message created by the rating implementation logic when it is executed by the Event Server as a result of changes to the accumulators.
Notification process	A daemon that reads notifications created by the Event Server from the Notifications table, formats them, and sends them to their relevant targets, such as the Transaction Broker or files.
performance indicator	See <i>accumulator</i> .
Persistence Writer Function	A part of the Event Server process that persists all permanent data to the Usage database.
primary session	The main session, associated with a single PDP context and session ID coming from the network. The primary session handles the frame event.
Rating Logic Configurator	A graphical user interface that enables the definition of the technical elements and rating rules, providing support for new services that the communication service provider may introduce in the future. This enables the communication service provider to implement different pricing strategies for the same event type (for example, mobile call, wireline call, and content transaction). In addition, the Rating Logic Configurator can be used to define and modify network protocols via implementation. Part of the product catalog.
Replication Target Function	A part of the Event Server process that is responsible for duplication of critical data on another server.
Rerate Prepare	A Turbo Charging process that is responsible for preparing rerate requests for rerating, which is performed by the Event Server.
resource	Any logical or physical item that is provisioned to the network or used for guiding. This includes such items as telephone numbers, SIM Cards, IP addresses, Internet user names, GPS services, and the like.
revenue recovery files	Files that contain unanswered events and events that were not handled because the Event Server was busy. These files are created by the network element, which sends them to Turbo Charging.

Term	Definition
segment	A logical collection of customers or resources. The segment is a number resulting from a function applied to a customer or resource ID. Such division of population into segments enhances performance.
service event	An event associated with one of the services provided under the frame event.
service session	A session representing a specific service given within the framework of the primary session. Each of these sessions is treated as a child of the primary session and is identified by the session ID and the rating group or service.
subscriber	A user with a clear identification that plays a key role in the customer model. The exact definition of a subscriber is implementation-dependent.
Update Handler	In Turbo Charging, a process that is responsible for the standard transfer of changes from the Amdocs Billing Customer Manager database to the internal Customer database.
Usage Extract	In Turbo Charging, a batch process that extracts rated events or accumulators from the database and writes the output into files.
Usage Query	In Turbo Charging, a mechanism used for viewing customer usage records (events and accumulators) after they have been rated by the Event Server, for updating event states, and for sending events to the Event Server.



Note: When describing business definitions in the product catalog, this document uses Rating Logic Configurator terminology. The terminology used in Enterprise Product Catalog – Portfolio Edition for these business definitions is different. For a mapping between the terminology used in the business subsystem of the Rating Logic Configurator and in Enterprise Product Catalog – Portfolio Edition, see Amdocs Enterprise Product Catalog Portfolio Edition Functional Overview.

2 ADJ1ICWFlow – Implementation Compiler Workflow

This chapter describes the Implementation Compiler Workflow process.

Description

The Implementation Compiler Workflow manages the complex sequence of operations that is performed from the release of a new implementation of the rating logic or of the messages compliant with the Diameter protocol to a product that can be used by the Event Server or other dependent processes.

One of these operations is to execute the Implementation Compiler to generate the rating or protocol logic code, or to compile the generated code to libraries. The Implementation Compiler Workflow supports different kinds of environments, such as a developer account or a runtime environment. Each environment determines the way the Implementation Compiler Workflow is initialized (for example, the location of third-party software, resources, and so on). It also dictates a set of parameters for the execution of operations (for example, for refreshing a manifest file, cleaning the working environment, and so on), which is referred to as a profile. The user can override a profile if necessary.

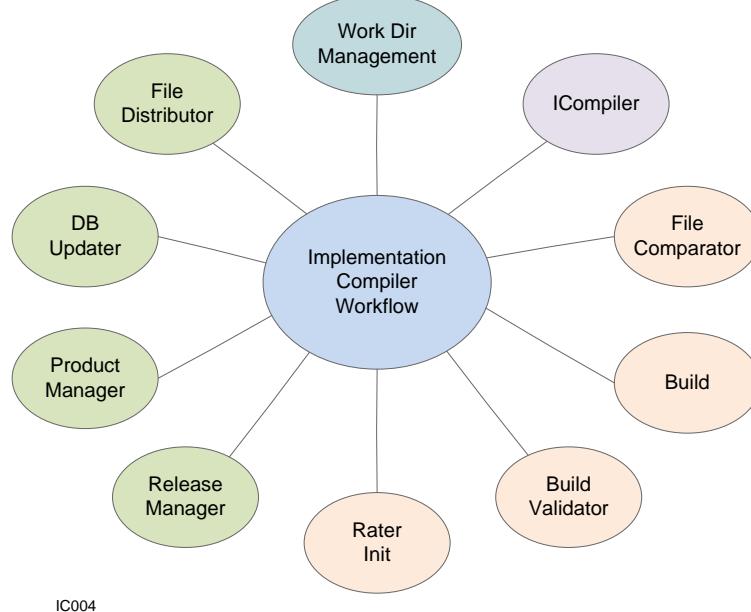
The Implementation Compiler Workflow can be executed as part of an operational map, manually (in a development/system test account), or by a trigger (Reference Table Synchronization process). The operational name of the Implementation Compiler Workflow process is ADJ1ICWFlow.

Implementation Compiler Workflow Tasks

Figure 2.1 shows the Implementation Compiler Workflow responsibilities and tasks, grouped by category:

- *Light purple* – Generation task
- *Light orange* – Build-related tasks
- *Light green* – Distribution-related tasks
- *Aqua* – Work dir management

Figure 2.1 Implementation Compiler Workflow Tasks



Following are the main tasks and responsibilities of the Implementation Compiler Workflow:

- *Work Dir Management* – Creates the Workflow environment (mandatory for the first time only). This environment is divided into the following main areas:
 - *Build area* – This directory is used for source code generation and construction.
 - *Generation area* – This area is used for keeping the input files, the registry, and the Implementation Compiler logs. In addition, the generated SQL and XML files are copied to this area for backward compatibility.
 - *Release area* – This directory is used for keeping the constructed product for later distribution.
- *ICompiler* – Runs the Implementation Compiler to generate implementation products:
 - Source code
 - Textual resource files, such as XML files or SQL scripts
 - Reports
- *File Comparator* – Handles group comparison and compares the include files when option 1 is running in an existing Implementation Compiler Workflow environment. For more information about the options, see the “Options” section in this chapter.

- *Build* – Compiles the generated files into a set of C++ libraries containing the rating or protocol logic. These libraries are loaded into the Event Server for processing incoming events.

The Build component includes the Group Generator, which organizes the generated files into groups by library directory. By default, a group contains 20 generated files. The grouped files are located under the `${build dir}/ ${variant}/ ${product type}/ sources/mpr_generated/cpp/src_grp` directory.
- *Build Validator and Rater Init* – Validate the compiled libraries by running validation tools and the Rater Init process, which loads the generated libraries. These tasks also load the updated reference data that was generated by the Implementation Compiler into the master database.
- *Release Manager* – Collects the various generated or compiled files and creates a release directory with the required content.
- *Product Manager* – Stores the release in the database. If the `MPRFLOW_STORE_PRODUCTS` environment variable is set to ‘false’, the libraries are not stored in the database. If the environment variable is set to ‘true’, the libraries are also written to the `ADJ1_MPR_PRODUCTS` table.
- *DB Updater* – Uses the generated SQL files to update the database structure (both reference and applicative), to support entities such as events and accumulators, which are dynamically defined in the Rating Logic Configurator.
- *File Distributor* – Distributes the compiled libraries to a set of predefined target locations.

For more information on the tasks, see the “Flow” section in this chapter.

The operation of the Implementation Compiler Workflow is always in the context of the release version, which is determined by the `MPRFLOW_RELEASE_VERSION` variable. This variable is automatically managed by the Implementation Compiler Workflow for the STANDARD and EXTENDED presets; however, if an Implementation Compiler Workflow operation fails, the operator may be required to populate this variable.

Working Environment

The Implementation Compiler Workflow performs its work in a working environment (referred to as the work directory – `<workdir>`). The location of the work directory is always passed as a command line to the Implementation Compiler Workflow process.

The work directory is built from three main directories representing the three main areas:

- *Mpr* – Generation area
- *Build* – Build area
- *Distribution* – Release area

These areas are described in subsequent sections.

Generation Area

This area represents the Implementation Compiler environment. It holds the resources needed for invoking the Implementation Compiler and the generated outputs.

The structure of this environment is defined in the configuration file of the Implementation Compiler (see the “Configuration Files” section in this chapter) and contains all the required elements for its execution. This environment is either built by the Implementation Compiler or created before execution.

The following is a common environment structure, defined in the default Implementation Compiler configuration:

- `${rootDir}` – The root directory.
- `${rootDir}\input\rlc\` – Contains input files from the Rating Logic Configurator.
- `${rootDir}\input\manifest\` – Contains input manifest files.
- `${rootDir}\output\sql\` – Contains output files of the SQL type. These files are moved to the build area before the next generation so that existing files can be removed or modified as necessary.
- `${rootDir}\output\xml\` – Contains output files of the XML type, including XML files generated for the Light Tracer, which are moved to the storage so that they can be used by the Light Tracer formatting utility. These files are moved to the build area before the next generation so that existing files can be removed or modified as necessary.
- `${rootDir}\logs\` – Contains the logs of the Implementation Compiler Workflow.
- `${rootDir}\registry\<mapper_registry.xml>\` – Contains the registry file of the Implementation Compiler.
- `${rootDir}\<configuration_file>.icc\` – Contains the configuration file of the Implementation Compiler.



Note: This is the structure of the runtime environment of the Implementation Compiler. This structure is located under the <workdir>/mpr directory, where workdir is the directory representing the environment of the Implementation Compiler Workflow. The default name of the rootDir in this environment is mpr_runtime.

The root directory is provided as a mandatory argument in the command line for the Implementation Compiler (`-rootDir`). The Workflow manager passes this argument to the Implementation Compiler. The directory can be configured via the corresponding environment variable (see the “Environment Variables” section in this chapter).

The output directory (in the common environment, \${rootDir}\output) does not have to be into the root directory; it can be located anywhere on the disk. Even if this location is defined in the configuration file of the Implementation Compiler, it can be overridden using the -o command line argument.



Caution: Although the Workflow enables customizing any command line argument of the underlying processes, the directories must be configured with caution. This is because such customization impacts on the underlying processes, and this impact must be fully understood before any changes are made.

Build Area

The build area contains the generated source code as well as the resources and scripts needed to compile the generated source code.

The build area is located under the <workdir>/build directory, with the following structure:

- \${build dir}/resources – A directory that contains internal resources needed for compilation (from the build kit).
- \${build dir}/include – A directory that contains the include files.
- \${build dir}/include_tmp – A temporary directory for the build. This directory is used to check which include files must be copied to the include directory.
- \${build dir}/status/compile – An internal directory for the compilation phase of the build process. It must not be deleted while the process is running. This is a temporary directory for parallel compilation that is used to determine whether a specific library is ready for linkage in parallel with the compilation.
- \${build dir}/status/linkage – An internal directory for the library phase of the build process. It must not be deleted while the process is running. This is a temporary directory for parallel linkage that is used to determine whether a group of dependent libraries has completed the linkage step.
- \${build dir}/\${variant}/\${product type}/build_info – A directory that contains generated build information for the build process.
- \${build dir}/\${variant}/\${product type}/logs – A directory that contains the log of the build and its validations.
- \${build dir}/\${variant}/\${product type}/sources/mpr_generated – A directory that contains the source files generated by the Implementation Compiler.
- \${build dir}/\${variant}/\${product type}/sources/mpr_generated/cpp/src_grp – An optional directory, created when the File Grouping mechanism is enabled. The File Grouping mechanism improves reduces the compilation time and library sizes.
- \${build dir}/\${variant}/\${product type}/products/lib – A directory that contains the compiled libraries.

Release Area

The release area holds the compiled libraries, compressed source files used to generate the libraries, and other resource files needed to set up an environment for the execution of the generated libraries.

The release area is located under the `<workdir>/distribution` directory, with the following structure. Each distribution is stored in a subfolder for each release date.

- `/${workdir}/distribution/${release_number}/${variant}/${product type}/sources/` – Contains the compressed source files, which have been generated by the Implementation Compiler and are used for building the library
- `/${workdir}/distribution/${release_number}/${variant}/${product type}/products/lib` – Contains the compiled libraries

Workflow Presets

The Implementation Compiler Workflow supports several environment types and uses a specific preset for each type. A preset is a set of predefined values.

The Implementation Compiler Workflow works with the following types of presets:

- Resource location presets (such as the location of .jar files and C++ libraries)
- Command line argument preset for managed processes

These presents are also referred to as profiles.

Presets are currently managed as environment variables in a .ksh or .bat file.

The user can change the behavior of managed processes by setting custom values in the preset file. This is done via the `ADJ1ICWFlow_setenv.ksh` file (or an `ADJ1ICWFlow_setenv.bat` file for Windows®) located in the working directory. This file is usually referred to as the local preset file.

The Implementation Compiler Workflow generates the local preset file when the user requests to create a working environment. The generated file contains all the command line arguments of the managed processes as comments. The user can remove the comment mark-up ('#' in Unix and 'rem' in Windows).

Example 1:

In the runtime environment, the default preset file for the command line arguments contains the option to import an external manifest file from the storage. The user can choose to add or remove this option by configuring the appropriate environment variable in the `workdir/ADJ1ICWflow_setenv.ksh` file.

Example 2:

In an environment in which the Operational database is available, the connection parameters for the reference database can be retrieved using the `-db` option. These connection parameters are used both by the Implementation Compiler process and the process that updates the database with the products of the generation. If the Operational database is not available, the user can use the local preset file to set up the `MPR_DB_*` variables and configure the connection to the reference database.

At the time of execution, the Workflow performs the following:

1. Determines the type of the environment in which it is working
2. Executes the default preset for the configuration of resources
3. Executes the default preset for the configuration of processes
4. Executes the local preset (if one exists)
5. Invokes the appropriate Ant task for the chosen option

Process Type

Batch process

Run Frequency

In general, the Implementation Compiler Workflow must run every time one of the following happens:

- A change in the Rating Logic Configurator implementation is distributed
- A core library fix is provided
- A core include file is changed (as part of the include kit)

The Dependency Checker for Implementation Compiler Libraries tool can perform a more fine-grained check to determine whether the Implementation Compiler Workflow must run. For more information about this tool, see the “[ADJ1IC_DependCheck – Dependency Checker for Implementation Compiler Libraries](#)” chapter.

Activation

This section describes how to run the Implementation Compiler Workflow.

Command Line:	ADJ1ICWFlow –workdir <Directory> -option <Option number or a preset> -db [-antoption <Ant option>]
Amdocs Monitoring & Control:	No
Script Name:	ADJ1ICWFlow.ksh
Executable Name:	ADJ1ICWFlow.ksh

Preliminary Actions

In the following cases, it is recommended that the operator invoke the Implementation Compiler Workflow in Clean mode (using the CLEAN preset; see the “[Options](#)” section in this chapter) prior to executing it in the STANDARD or EXTENDED mode:

- The work directory contains generated or compiled products produced by a different implementation.
- The reference tables populated by the Implementation Compiler Workflow contain data produced by a different implementation.

Command Line Arguments

The Implementation Compiler Workflow supports the following command line arguments that affect its execution:

- **-workdir** – This argument specifies the working directory for the Implementation Compiler Workflow. This directory marks the root of the Workflow environment, which is created by the Workflow on demand.
- **-option** – The Workflow manager exposes more than twenty tasks, from the creation of the Workflow environment, generation, and compilation, to distribution to the Event Servers. In a common scenario, the Workflow exposes these tasks by logical groups or by presets. Valid option numbers range from 1 to 6 and from 10 to 22. Valid presets are ‘STANDARD’, ‘EXTENDED’, ‘CLEAN’, and ‘INFO’. For more information, see the “[Options](#)” section in this chapter.
- **-db** – This option instructs the Implementation Compiler Workflow to get the database definition from the operational task. This option is available only on Unix with access to the Operational database.
- **-antoption** – This argument enables the operator to pass an Ant option to the Ant process that executes the tasks. One of the most commonly used options is **-debug**, which enables the Ant task to print information on the execution of the targets. Another possible value is **-verbose**. These two options represent different granularities for debugging the Ant scripts.

The processes invoked by the Workflow also require command line arguments. The Workflow supports them via its environment variables. For example, the Implementation Compiler can retrieve external manifest files by using the **-m <Path to the manifest directory>** argument. This option can be set via the **MPR_RT_CMD_ARG_MANIFEST** environment variable (see the “[Environment Variables](#)” section in this chapter). The environment variables can be set in the **ADJ1ICWFlow_setenv.ksh** file in the work directory. This file contains all the configurable parameters and is created by the Workflow when it is requested to create the working environment. If the file does not exist, it must be created by the operator.

Options

The options are organized in the order of their logical dependencies (for example, option 3 cannot be executed if option 2 has never been run). They are divided into the following categories representing different granularity:

- *Regular* – Includes options from 1 to 6. Each option can include several steps. These are medium granularity options.

Example:

Option 1 creates the work environment, and installs the include kit and the build kit. These tasks are grouped in one option because they are commonly used.

- *Detailed* – Includes options from 10 to 22. These are lower granularity options. The user can use a detailed option to skip a step of a higher granularity option.
Example:
Option 3 (build) compares files, compiles them, validates the compiled object, and invokes the Rater Init. If a user wishes only to perform compilation (and not file comparison), option 15 (compile) can be used instead of option 3.
- *Preset* – Includes the STANDARD, EXTENDED, CLEAN, and INFO presets. The STANDARD and EXTENDED presets are higher granularity options. They enable the user to handle common scenarios, such as generate, build, and prepare for release.
 - *STANDARD* – Runs options 2 to 5.
 - *EXTENDED* – Runs options 13 to 22.
 - *CLEAN* – Cleans the specified area of the runtime environment before starting. Valid values are:
 - *SOURCES* – Removes the sources from the build area
 - *PRODUCTS* – Removes the products from the build area
 - *BUILD* – Cleans the build area (removes the sources and products)
 - *DB* – Cleans the reference tables
 - *INFO* – Displays information about the runtime environment, for example, prints the variables used by the Implementation Compiler Workflow.

For more information on the various options, see the “Flow” section in this chapter.

Run Modes

A run mode defines the way in which the Workflow finds the resources for its execution. The Workflow process can run in the following modes:

- *RT* – Runtime environment for a system test or production account
- *RT_ADJ* – Runtime environment for the Turbo Charging testing environment (which differs in the definition of the storage from the regular testing environment)
- *STANDALONE* – For a self-installed environment (usually a manually configured or a Windows environment)

The run mode is specified by setting the MPRFLOW_RUN_MODE environment variable. If it is not set, the Workflow manager determines the run mode as follows:

- If the CCPROJECT_HOME environment variable does not exist:
 - If the Workflow resource files are present in the directory specified in the ABP_CONFIG environment variable, this is a runtime environment (RT).
 - If the Workflow resource files are not present in the directory specified in the ABP_CONFIG environment variable, this is a Turbo Charging runtime environment (ADJ_RT).



Note: It is usually not necessary to specify the run mode to the Implementation Compiler Workflow; it is determined automatically.

The user can request the Implementation Compiler Workflow to print information including its work mode by using the `info` option in the activation command line.

First-Time Activation

When the Implementation Compiler Workflow is used for the first time in an account, the work directory must be created in a standalone step, and then other steps can be invoked.

When the work directory exists, there is no need to perform this step unless the operator wants to create another work directory.

Activation with File Grouping

The File Grouping mechanism reduces compilation time and library size. This mechanism uses the Group Generator to organize the generated files into groups. The default group size is 20 files.

To customize the group size for a particular library:

1. Open `ADJ1ICWFlow_setenv.ksh` or `.bat` file, and export the corresponding `MPRFLOW_BUILD_NUM_OF_FILES_IN_GROUP_<name of the library directory>` environment variable.

Example:

To have a group of 100 files for the AIMOS formatting library, populate the environment variable as follows:

```
export  
MPRFLOW_BUILD_NUM_OF_FILES_IN_GROUP_aimosFormatting=100
```

If you customize the grouping for a specific library, you must delete the products for the specific library before running the build.

2. Run the Implementation Compiler (generated code) build.

File grouping is activated by default. However, if you are working with a build kit and `workdir` created without file grouping, you must perform additional steps to start using the File Grouping mechanism.

To activate file grouping:

1. Clean the product area of the Implementation Compiler build under `workdir/build/<build variant>/<build type>/product` (for example, `workdir/build/64/trace/product`).
2. Create a back-up of the `ADJ1ICWFlow_setenv.ksh` or `ADJ1ICWFlow_setenv.bat` (for Windows) file in the `workdir` directory.
3. Run option 1 or options 10, 11, and 12 of the Implementation Compiler Workflow.
4. Merge the contents of the back-up `ADJ1ICWFlow_setenv.ksh` or `.bat` file with the new `ADJ1ICWFlow_setenv.ksh` or `.bat` file, which is created by the Workflow in options 1 or 10.

5. To customize the group size for a particular library, open the merged ADJ1ICWFlow_setenv.ksh or .bat file, and export the corresponding MPRFLOW_BUILD_NUM_OF_FILES_IN_GROUP_<name of the library directory> environment variable.

Example:

To have a group of 100 files for the AIMOS formatting library, populate the environment variable as follows:

```
export  
MPRFLOW_BUILD_NUM_OF_FILES_IN_GROUP_aimosFormatting=100
```

If you customize the grouping for a specific library, you must delete the products for the specific library before running the build.

6. Run the Implementation Compiler (generated code) build.

Activation after a Private Fix

Private fixes are supported for .ksh scripts and for Java .class and .jar files.

The location of private files for RT and ADJ_RT environments is as follows:

- *.ksh files* – The pbin directory
- *.jar files* – The regular location for private .jar files for the application in the storage
- *Classes* – The regular location for private classes in the storage or in the workdir/private/classes directory
- *Core h file* – The appropriate location in the work directory:
<workdir>/build/include/<area>

Activation of the Implementation Compiler Workflow after a private fix depends on the type of the fix. The following steps must be performed:

- *Core library fixes (not Implementation Compiler fixes) and core include files* – Recompilation followed by later steps (no need to regenerate the code). In the case of an .h file change, it is recommended that the operator run a clean build (using the CLEAN preset), followed by regular execution.
- *Implementation Compiler fixes* – Generation followed by later steps (compilation, distribution, and so on)
- *Implementation fixes* – Generation with refreshing Rating Logic Configurator data followed by later steps
- *Manifest fixes* – Generation with refreshing Implementation Compiler files and copying the custom manifest followed by later steps.

Activation of Rater Init

Normally, Rater Init runs as part of the Implementation Compiler Workflow (option 17 and option 20). However, it is also possible to run it separately after the build phase or after updating reference tables, before starting the Event Server.



Note: When a new pricing item type and offer are added to the implementation, Rater Init must run only after the reference data SQL queries have been executed.

Rater Init checks that the code that was generated by the Implementation Compiler is at least in minimal synchronization with the Rating Engine. This synchronization ensures that the initialization stage of the Rating Engine is successful. In addition, because Rater Init performs validation checks on the implementation, it can reveal implementation issues before the Event Server is run.

To start Rater Init manually, run the following command:

```
ADJ1RaterInit_Sh -d <DB_CONNECTION_STRING> [-w <y/n>] [-x <y/n>] [-r <y/n>] [-b <y/n>]
```

Example:

```
ADJ1RaterInit_Sh -d adisc/adisc@TC750A3 -w y -x y -r y -b y
```

This command takes the following arguments, which affect the execution of Rater Init:

- **-d** – The connection string to the database in the following format:
user_name/password@instance.
- **-w** – Determines the version of the Implementation Compiler libraries to be used. There are two types of versions:
 - *IN WORK* – Libraries that are unapproved because the flow has not yet finished successfully
 - *RELEASED* – Libraries that are approved because the flow has finished successfully

If the value of this argument is ‘n’ (default), the RELEASED version is used. Otherwise, the IN WORK libraries are taken.

- **-x** – Determines whether the RaterInit.xml file must be generated and placed under \$ABP_LOG. This file contains the data of all events, offers, and packages. The default value is ‘y’ (generate the XML file).
- **-r** – Determines whether reference data must be validated. Because the Implementation Compiler generates SQL files that update the reference database, the validation checks that the commands in these files ran, and that their effect is aligned with the code generated by the Implementation Compiler. The default value is ‘n’ (no validation).
- **-b** – Determines whether business builders, which bring the business data from the product catalog, are loaded and checked. The default value is ‘y’ (business builders are loaded).

The output of the Rater Init executable is written into the following file:

`${ABP_APP_ROOT}/log/RaterInit/<date>/ADJ1_RaterInit_<date>.log,`

where the `<date>` is in the following format: `%Y%m%d_%H%M%S`.

Preceding Processes

The following processes must run successfully before this process is activated:

- *Rating Logic Configurator's technical data distribution* – Provides the Implementation Repository and auxiliary files. The Rating Logic Configurator also distributes business data (such as offers and pricing packages) that are usually frequently updated. However, the Implementation Compiler only requires the technical data for its operations.
- *Reference Table Synchronization (RTS) process* – The rules for copying the contents of the PC1_XML_DISTRIB table into the APE1_XML_DISTRIB table are defined by implementation. They are used to specify for the Reference Table Synchronization process how to handle the information distributed by the Rating Logic Configurator.

Dependent Processes

The following processes can be run only after the successful completion of this process:

- Distribution of updated libraries to the Event Servers. The distribution process ensures that the reference tables and the libraries are synchronized.
- Update of database data and structure required as a result of the changes in the Rating Logic Configurator implementation. This update is made by running SQL scripts that are generated during the execution of the Implementation Compiler.

Affected Applications

The process affects the following applications:

- *Event Server* – Loads the C++ libraries that are created from the code generated by the Implementation Compiler. These libraries contain the rating logic and, optionally, Diameter-related data (if the protocol was implemented using the Rating Logic Configurator). The Event Server uses Diameter-related code in the following areas:
 - *Selection cases* – Based on the message, the selection case the types of service events to be created from the frame event.
 - *Parsing mapping cases* – Based on the message and the specific service event type, the parsing mapping case maps the message content to the Rating Logic Configurator (PC) event (populates the event attributes and session variables, checks for partial charge, and so on).

- *Construction mapping cases* – Construction mapping cases prepare Diameter-compliant outgoing messages to the network:
 - Diameter Credit Control Application (DCCA) answer messages (for example, Credit Control Answer (CCA))
 - Server-initiated commands (for example, Re-Authorization Request (RAR))
- *Update Handler* – Reads data from the database tables that are updated with information generated by the Implementation Compiler.
- *Event Extract* – Loads the C++ libraries that are created from the code generated by the Implementation Compiler.
- *Usage Query Server* – Loads the C++ libraries that are created from the code generated by the Implementation Compiler.
- *Usage Query Proxy* – Uses data in the database tables that are populated with information generated by the Implementation Compiler.
- *Notification process* – Loads the generated formatting functions that format the notification data according to a compliant transaction format (defined in output format manifest files).
- *Dispatcher* – Loads the generated formatting functions that format the dispatching record (external record) data according to a compliant format.
- *Reconciliation* – Uses the Implementation Compiler output as follows:
 - Reads the layout of the data from the generated ADJ1_METADATA table
 - Reads the accumulator buffer data (parsed by the Implementation Compiler) from Amdocs In-Memory Object Storage (AIMOS) and compares it to the data from the database
- *Rejected Event Recycler* – Uses the APE1_RATED_EVENT_MAP mapping table generated by the Implementation Compiler to get information about the mapping between fields and attribute details, such as the attribute type, length, and so on.
- *Rerate Prepare* – Uses the SQL scripts generated by the Implementation Compiler to select events from the database.
- *Read-Only Rater client* – Uses unique SQL scripts that enable it to select single or multiple rated events, and single rejected events.
- *Copy Cycle Usage* – Uses unique SQL scripts that enable it to select multiple records from a source database and insert them into the target database.

Log Files

The Workflow executes several sub-processes; therefore, a number of logs are created. In addition, the Workflow has its own log.

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Regular log files:*
 - *Implementation Compiler Workflow logs* – icwflow.<Date>_<Time>.log
 - *Implementation Compiler logs* – mpr.<Date>_<Time>.log
 - *Compilation logs* – log.build.<Date>_<Time>.txt
 - *Validation logs* – log.buildCheck.<Date>_<Time>.txt
 - *Rater Init logs* – log.raterInit.<Date>_<Time>.txt
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Regular log files:*
 - *Implementation Compiler Workflow logs* – <workdir>/logs
 - *Implementation Compiler logs* – <workdir>/mpr/mpr_runtime/log
 - *Compilation logs* – <workdir>/build/sources/<variant>/<product type>/log
 - *Validation logs* – <workdir>/build/sources/<variant>/<product type>/log
 - *Rater Init logs* – <workdir>/build/sources/<variant>/<product type>/log
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter)

Contents

The log files contain the following:

- *Regular log files* – Information about the execution of each process. This information includes debug, info, warning, and error messages (depending on the logging level defined in the *log4j.properties* file).
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Input Files

This section describes input files that are mainly used by the Implementation Compiler. They are required for the successful execution of the relevant Implementation Compiler Workflow tasks.

Implementation Compiler Registry File

This section describes the Implementation Compiler registry file.

Name

mapper-registry.xml

Location and Format

The Implementation Compiler uses the registry file by default. This file is the only file under the `<workdir>/mpr/mpr_runtime/registry` directory.

It is possible to define a custom registry via the `MPR_RT_CMD_ARG_REGISTRY` environment variable.

In this case, the Implementation Compiler creates a back-up of any registry file (`mapper-registry.xml`) in the registry folder and then copies the file specified via the command line to the same directory. The new file is renamed `mapper-registry.xml`.

It is also possible to protect the registry via the `MPR_RT_CMD_WORKFLOW_MODE` environment variable:

- When the variable is set to ‘REG’ (default), if no registry file is found, the Implementation Compiler creates a new file.
- When the variable is set to ‘PROTECTED’, if no registry file is found, the process throws a designated exception.

Contents

The registry file contains all the data that must be persisted between code generation activities (for example, the IDs assigned to events and accumulators).



Caution: This file must never be deleted, to ensure consistency from one code generation activity to the next.

Implementation Repository File

This section describes the Implementation Repository file.

Name

Implementation_repository.xml

Location and Format

The process receives the Implementation Repository XML file from the distribution table. The file is extracted from the database, unzipped, and placed in the file system in the `<workdir>/mpr/mpr_runtime/input/rlc` directory.

Contents

The Implementation Repository file is an output file of the Rating Logic Configurator. It contains the definitions of such rating-related entities as events, accumulators, pricing item types, elementary types, extension functions, and more. The file also contains the rating logic that is used to rate incoming events.

Protocol Dictionary File

This section describes the Protocol Dictionary file.

Name

Protocol_dictionary.xml

Location and Format

The process receives the Protocol Dictionary XML file from the distribution table. The file is extracted from the database, unzipped, and placed in the file system in the `<workdir>/mpr/mpr_runtime/input/rcl` directory.

Contents

The Protocol Dictionary file is an output file of the Rating Logic Configurator. It contains the definitions of information elements used for creating attribute-value pairs (AVPs) that comprise the Diameter protocol dictionary. The AVPs participate in the construction of messages intended for communicating with the network by means of the Diameter protocol. In addition to the AVPs, the Protocol Dictionary file contains the structure of Diameter messages.

The Protocol Dictionary file is validated by the Implementation Compiler in the generation phase and by the Event Server in the initialization phase.

Auxiliary Files

This section describes auxiliary files.

Location and Format

The number of auxiliary files varies according to the specific Rating Logic Configurator implementation.

The process receives the auxiliary XML files from the distribution table. The files are extracted from the database, unzipped, and placed in the file system in the `<workdir>/mpr/mpr_runtime/input/rcl` directory.

Contents

There is a file for each auxiliary object, such as the value list, mapping tables, period sets, and the special day set, that is defined in the Rating Logic Configurator. The files contain the auxiliary object structure and data. The Implementation Compiler only makes use of the structure information.

Manifest Files

The manifest input files are XML files with information required for code generation. They are used to:

- Provide information that cannot be defined in the Rating Logic Configurator
- Supplement the implementation data of the Rating Logic Configurator

The manifest directory is flat (does not contain subdirectories).

Manifest files follow this naming convention:

<Manifest type><Customization level>_<File name>.xml, where:

- The manifest type is one of the following:
 - *Extension functions* – EFM
 - *Output formats* – OFM
 - *Auditing counters* – ACM
 - *Duplicate check* – DCM
 - *Mapping tables* – MTM
 - *Pre-emptive events* – PSSM
- The customization level is ‘1’ (for core files).

Manifest files can be stored in the database, related to a specific configuration, and refreshed (via the `-refresh` option).

The manifest files are located in the `<workdir>/mpr/mpr_runtime/input/manifest` directory.

For more information about manifest files, see *Turbo Charging Implementation Compiler XML Configuration Guide*.

Extension Function Manifest Files

This section describes the extension function manifest files.

Name

`EFM1_<free text>.xml`

Location and Format

The process reads the extension function manifest XML files from the root manifest folder. The location of the root manifest folder is specified in the configuration file of the Implementation Compiler.

The manifest folder may contain multiple files with the same structure. For example, extension function manifests are divided into several files according to the type of function.

Contents

Extension function manifest files are:

- Created by the function developer
- Used to define the physical implementation data of extension functions, including the physical function name, name space, header file, list of parameters, and so on
- Used by the generated code to call the ‘real’ function, since the data in the Rating Logic Configurator only contains logical information about the function

Following is an example of an extension function manifest file.

Example:

```
<Function logicalName="Apply allowances" type="EntityMethod" returnDataType="Void"
generationType="Normal">
  <Description>Text</Description>
  <ImplementationInfo>
    <HeaderFile>adj/pe/ApplyRoleExtFunc.h</HeaderFile>
    <NameSpace>adj::pe</NameSpace>
    <PhysicalName>applyAllowances</PhysicalName>
  </ImplementationInfo>
  <Parameters>
    <Parameter name="i_event"      dataType="Event"      order="1" isContext="true"/>
    <Parameter name="inputContext" dataType="InputContext" order="2" isContext="false"/>
    <Parameter name="appContext"   dataType="AppContext"   order="3" isContext="false"/>
  </Parameters>
```

Output Format Manifest Files

This section describes the output format manifest files.

Name

OFM1_<*free text*>.xml

Location and Format

The process reads the output format manifest XML files from the root manifest folder. The location of the root manifest folder is specified in the configuration file of the Implementation Compiler.

Contents

The output format manifest files define the possible formats required for event extracts and usage queries. The formats are used to format the output data of events and accumulators that are extracted from the database by the Usage Extract or Usage Query processes.

Auditing Counter Manifest Files

This section describes the auditing counter manifest files.

Name

ACM1_<*free text*>.xml

Location and Format

The process reads the auditing counter manifest XML files from the root manifest folder. The location of the root manifest folder is specified in the configuration file of the Implementation Compiler.

Contents

The auditing counter manifest files define the event or accumulator attributes to be monitored by each auditing counter. It is possible to monitor their values before the rating phase, after it, or both. The manifest file also includes the target unit of measurement of the counter. The Implementation Compiler generates a formatting function for each event containing the requested information in the target unit of measurement and precision.

Duplicate Check Manifest Files

This section describes the duplicate check manifest files.

Name

DCM1_<*free text*>.xml

Location and Format

The process reads the duplicate check manifest XML files from the root manifest folder. The location of the root manifest folder is specified in the configuration file of the Implementation Compiler.

Contents

The duplicate check manifest files define the attributes to be used by the Duplicate Check mechanism. They contain the events that are marked for a duplicate check and their attributes. An event can inherit these duplicate check attributes from its base event. These definitions can be overridden at the event or attribute level.

The following is an example of a duplicate check manifest file.

Example:

```
<dupcheck-descriptors>
  <dupcheck Descriptor name="Event" inherit-base="true">
    <attributes-info>
      <attribute-info duplicate-check-ind="true" name="Event type ID"/>
    </attributes-info>
  </dupcheck Descriptor>
  <dupcheck Descriptor name="Basic event" inherit-base="true">
    <attributes-info>
      <attribute-info duplicate-check-ind="true" name="Resource value"/>
      <attribute-info duplicate-check-ind="true" name="Resource type"/>
    </attributes-info>
  </dupcheck Descriptor>
...

```

Mapping Table Manifest Files

This section describes the mapping table manifest files.

Name

MTM1_<*free text*>.xml

Location and Format

The process reads the mapping table manifest XML files from the root manifest folder. The location of the root manifest folder is specified in the configuration file of the Implementation Compiler.

Contents

The mapping table manifest files define the features that are specific to mapping tables, such as the partial key or best match search columns.

The following is an example of a mapping table manifest file.

Example:

```
<partial-key-manifest xmlns:jaxb="http://java.sun.com/xml/ns/jaxb"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="partial-
    keys.xsd" schema-version="String" description="Example for partial key">
    <mapping-tables>
        <mapping-table auxiliary-type-name="Balance threshold policy">
            <partial-keys>
                <partial-key key-name="PartialKeyOne">
                    <key-columns>
                        <column column-name="DIRECTION"/>
                        <column column-name="CURRENCY"/>
                    </key-columns>
                    <order-by-columns>
                        <column column-name="EFFECTIVE_DATE"/>
                    </order-by-columns>
                </partial-key>
            </partial-keys>
        </mapping-table>
    </mapping-tables>
</partial-key-manifest>
```

Pre-Emptive Event Manifest Files

This section describes the pre-emptive event manifest files.

Name

PSSM1_PreemptiveServicesManifest.xml

Location and Format

The process reads the pre-emptive event manifest XML files from the root manifest folder. The location of the root manifest folder is specified in the configuration file of the Implementation Compiler.

Contents

The pre-emptive event manifest files define the events that the Event Server creates and rates even though they do not arrive from the network. These events cause the creation of service sessions in those cases where the primary session does not contain any.

The following is the XSD for the pre-emptive manifest file.

```
<?xml version="1.0" encoding="UTF-8"?>
<xss:schema xmlns:xss="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xss:element name="PreemptiveDefinitions">
    <xss:complexType>
      <xss:sequence>
        <xss:element minOccurs="0" ref="ProtocolID"/>
      </xss:sequence>
    </xss:complexType>
  </xss:element>
  <xss:element name="ProtocolID">
    <xss:complexType>
      <xss:sequence>
        <xss:element ref="ServiceSessionPreemptiveInfos"/>
      </xss:sequence>
      <xss:attribute name="Id" use="required" type="xs:integer"/>
    </xss:complexType>
  </xss:element>
  <xss:element name="ServiceSessionPreemptiveInfos">
    <xss:complexType>
      <xss:sequence>
        <xss:element minOccurs="1" maxOccurs="unbounded" ref="ServiceSessionPreemptiveInfo"/>
      </xss:sequence>
    </xss:complexType>
  </xss:element>
  <xss:element name="ServiceSessionPreemptiveInfo">
    <xss:complexType>
      <xss:attribute name="PcEventName" use="required" type="xs:string"/>
      <xss:attribute name="ServiceId" use="required" type="xs:integer"/>
    </xss:complexType>
  </xss:element>
</xss:schema>
```

The following is an example of a pre-emptive event manifest file.

Example:

```
<PreemptiveDefinitions>
  <ProtocolID Id="7">
    <ServiceSessionPreemptiveInfos>
      <ServiceSessionPreemptiveInfo ServiceId="7" PcEventName="GPRS authorization" />
    </ServiceSessionPreemptiveInfos>
  </ProtocolID>
</PreemptiveDefinitions>
```

Verification Identification Information File

This section describes the verification identification information file.

Name

VerIdentInfo.ini

Location and Format

A sample file is located in the `<workdir>/build/resources/config` directory.

The file is divided into sections marked by square brackets. The following section types are customizable:

- `[COMMAND]` – Specifies the command that must be run to embed metadata in the library
- `[ENVIRONMENT]` – Provides the name of the environment variable that retrieves the value to be embedded in the library

Contents

The file is used by the build system of the Implementation Compiler to insert metadata into the generated libraries (in Unix only).

The user can use `VerIdentInfo.ini` file to provide the custom values to be embedded in a library.

The following is the default `VerIdentInfo.ini` file:

```
[COMMAND]
Date = date +%Y%m%d_%H%M%S
[GENERAL]
[ENVIRONMENT]
Product_Name = CCPRODUCT
Product_Version = CCPRODUCTVER
Variant = VARIANT_MODE
Host = HOST
User_Name = LOGNAME
ORACLE_HOME = ORACLE_HOME
ACE_HOME = ACE_HOME
BOOST_HOME = BOOST_HOME
XALANCPP_HOME = XALANCPP_HOME
ANT_HOME = ANT_HOME
MPR_DB_USER = MPR_DB_USER
MPR_DB_INSTANCE = MPR_DB_INSTANCE
MPR_DB_SERVER = MPR_DB_SERVER
[DATABASE]
deprecated.
[LAYOUT]
Core_Product = abp
[OUTPUT_LAYOUT]
[EXCEPT_BBS]
[BBS_64_BIT]
```

The metadata that was added to the libraries can be retrieved using the following command:

what <library name> | grep CC

Output Files

The Implementation Compiler generates thousands of output files, which mainly contain generated source code in various syntaxes, such as C++, SQL, and XML.

Folder Structure

All output files are generated in several subfolders under one output folder, which by default is located under the `workdir/mpr/mpr_runtime/output` directory.

Under the root output folder, the Implementation Compiler automatically creates the following folder structure:

<Root output folder>

- flavor (production/trace)
 - cpp
 - ◆ pub
 - ◆ mapper
 - ◆ ...
 - src
 - ◆ ...
 - sql
 - ◆ ...
 - xml
 - ◆ ...



Note: These folders contain additional subfolders that are not specified here.

Reports

The Implementation Compiler Workflow generates reports that analyze the Rating Logic Configurator implementation and display statistics on the sizes of events and accumulators, the use of overlap fields, and more. These reports can be used to estimate the required database size and check the effectiveness of attribute persistence.

The reports are located in the `workdir/mpr/mpr_runtime/output/xml/general` directory. They include the following information:

- *EntityInfo.xml* – This report shows information about events and accumulators. It has two sections:
 - A summary of database details for all the events and accumulators, per entity type:
 - Average entity size (without the size of the parent entities)
 - Name of the entity with the minimum size

- Minimum entity size (without the size of the parent entities)
- Name of the entity with the maximum size
- Maximum entity size (without the size of the parent entities)
- Average size of a column in the database
- Minimum size of a column in the database
- Maximum size of a column in the database
- Number of columns in the database (without the spare columns)
- Total OCI buffer (without the spare columns)
- Information about each event or accumulator entity:
 - Is leaf (whether or not the entity is defined on the event or accumulator leaf)
 - Entity type (event or accumulator)
 - ID
 - Name
 - Number of persistent attributes in the database and their size (without the size of the parent attributes)
 - Number of serialized attributes in memory and their size (without the size of the parent attributes)
- *EntityInfoPI.csv* and *EntityInfoEvent.csv* – These reports show information about the persistence columns of accumulators and events. Each row in the CSV file contains the following data:
 - Attribute name
 - Attribute type (simple or complex)
 - Raw attribute type (such as String or Numeric)
 - Name of the column in the database for this attribute
 - Is leaf (whether or not the attribute is defined on the event or accumulator leaf)
 - Name of the base (parent) accumulator or event, if one exists
- *AccumPersistenceAttrInfo.html* – This report shows information about the attributes of each accumulator:
 - Accumulator name
 - Total number of attributes, followed by the number of persistent attributes in parentheses
 - Number of attributes that are updated in the initialization phase
 - Number of attributes that are updated in the computation phase
 - Number of attributes that are not updated at all

- Number of attributes that are not updated at all but are persisted to the database
- Names of attributes that are not updated at all but are persisted to the database
- *Validation-Report.xml* – This report shows information about each validation message:
 - *Message header* – Including the following elements:
 - *Severity* – The severity of the problem (fatal, error, warning, or information)
 - *Category* – The category to which the validation message belongs
 - *Type* – The type of the message
 - *Message body* – The actual text of the message that is based on a dynamic template and includes the arguments that the message gets
 - *Optional additions* – Such optional information as a URL that leads to the entity associated with the error message or the description of an exception if applicable

Event Log Attribute Dictionary

The event log contains fields with useful statistical information per event. There is a common unique identifier for lines *produced by the same event* in any module on any machine. This enables the user to construct a sequential history of an event as it progressed from module to module. The event log is produced for the Guiding to Customer (FR), Event Processing (RB), and outgoing Network Interface (CR) reporting points.

The implementation can add attributes to the event log produced at the Event Processing point using the *Event log customization* property in the Rating Logic Configurator. The Implementation Compiler generates the *Attribute_print_customization.xml* file, which provides users with a dictionary of the customized attributes to be printed to the log.

Flow

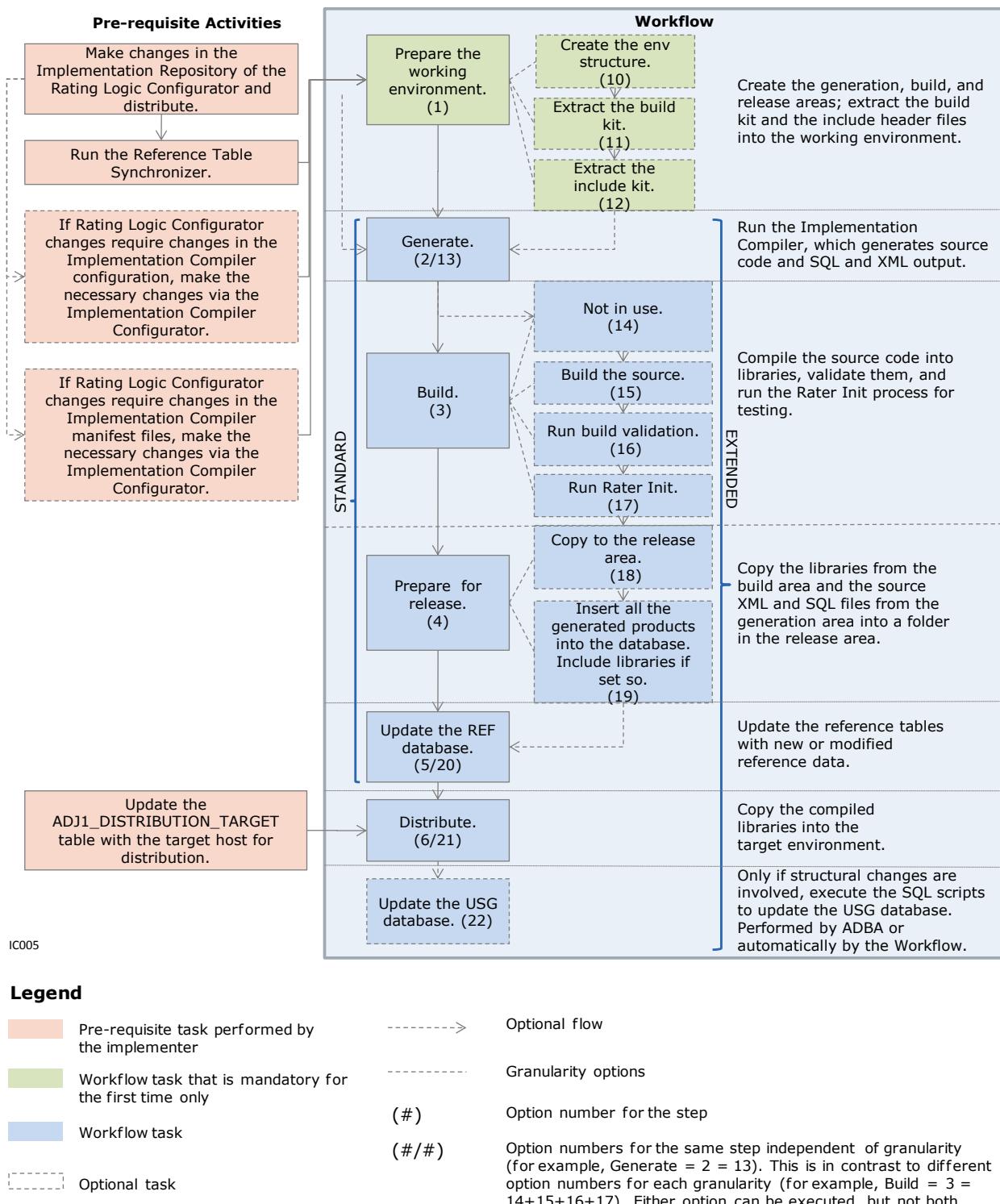
This section describes the general flow and behavior of the Implementation Compiler Workflow.

General Flow

The Implementation Compiler Workflow is internally implemented as Ant tasks, wrapped by a shell script. The shell script simplifies working with the Ant tasks, and enables running the flow in a single command or executing each task separately. The Workflow is designed to handle the full processing flow, from running the Implementation Compiler to updating the reference data. Additional tasks, such as creating an environment and distributing the libraries, can be performed upon request.

Figure 2.2 shows the steps involved in the Implementation Compiler Workflow as well as the steps necessary for its successful execution (for example, getting the libraries and the environment to run the Event Server with the new implementation result). For more information about options, granularities, and presets, see the “Options” section in this chapter.

Figure 2.2 Implementation Compiler Workflow



The following table details the steps of the Implementation Compiler Workflow.

Regular Option #	Detailed Option #	Step	Description	Performed by	Comments
Prerequisite	Prerequisite	Making changes in the Rating Logic Configurator	This step involves implementation changes made to the Implementation Repository in the technical subsystem (Rating Logic Configurator) of the product catalog.	Implementer using the Rating Logic Configurator GUI	Changes to the business subsystem of the product catalog, such as the addition of offers or pricing packages, are not relevant for this flow as they do not affect the code generated by the Implementation Compiler.
		Executing the Reference Table Synchronization (RTS) process	This step involves running the Reference Table Synchronization process, which handles the data distributed from the Rating Logic Configurator and prepares it for other applications, such as the Implementation Compiler Workflow, the Event Server, and more.	Manually by the implementer	N/A
		Creating and editing manifest files (optional)	This step involves creating and editing manifest files for extension functions, output formats, and more.	Developer or implementer	This step is optional because in many cases, the changes to the Rating Logic Configurator do not require adding or modifying manifest files.
		Modifying the Implementation Compiler's configuration (optional)	This step involves editing the configuration of the Implementation Compiler to achieve the required generated code.	Implementer	This step is optional because in many cases, the changes to the Rating Logic Configurator do not require modifying the configuration of the Implementation Compiler.
		Configuring the target environment to which the data is to be distributed	This step involves updating the ADJ1_DISTRIBUTION_TARGET table to indicate to which host the libraries are to be distributed.	Implementer	This step is needed if the distribution process of the Implementation Compiler Workflow is used to distribute the compiled code.
1	10, 11, 12	Creating a build environment (mandatory for the first time only)	This step involves creating a build environment, which includes build scripts, header files, and utilities.	Manually or automatically as a task in the Workflow	For more information, see the “Working Environment” section in this chapter.

Regular Option #	Detailed Option #	Step	Description	Performed by	Comments
2	13	Executing the Implementation Compiler	This step involves executing the Implementation Compiler, which analyzes the Rating Logic Configurator implementation, manifest files, and the configuration, and generates source code to match this input data.	Automatically by running the Implementation Compiler as a task in the Workflow	<p>Code generation is automatically performed only for new or modified objects. The operator can configure the number of threads used for the generation, to enable parallel source generation.</p> <p>The output of the generation process resides in the build area of the working environment.</p>
3	14	Not in use	N/A	N/A	N/A
	15	Creating a build	This step involves a build process, which compiles the generated code into a set of libraries.	Automatically as a task in the Workflow	<p>The build process uses the following strategies:</p> <ul style="list-style-type: none"> ■ <i>Grouping mechanism</i> – Enables grouping a configured number of .cpp files into a single file. ■ <i>Parallel mechanism</i> – Enables parallelism at the compilation stage, when a configured number of files are compiled concurrently. ■ <i>Parallel dependent mechanism</i> – Enables running a linkage process on a library that has already been compiled, according to the dependency configuration. <p>This mechanism enables the linkage process to run in parallel with the compilation stage rather than separately.</p>
	16	Validating the libraries created	This step involves checking any unresolved symbols to determine whether the libraries were built correctly.	Automatically as a task in the Workflow	N/A

Regular Option #	Detailed Option #	Step	Description	Performed by	Comments
	17	Running Rater Init	After the new libraries have been created, this step involves executing a test to make sure that they are working correctly.	Automatically as a task in the Workflow	N/A
4	18	Copying the new libraries into the release area	This step involves copying the newly generated libraries into a special folder from which the libraries are distributed to their final targets.	Automatically as a task in the Workflow	A new subfolder is created in this folder for each execution of the flow. The name of the subfolder includes the version of the flow and the time of execution.
	19	Inserting the generated products into the database	<p>This step involves inserting the generated products into the database. The types of information that is inserted depend on the value of the MPRFLOW_STORE_PRODUCTS environment variable. By default, all generated products except for libraries are inserted to the database.</p> <p>If the value of the MPRFLOW_STORE_PRODUCTS environment variable is set to ‘true’, the libraries are compressed and restored in the database (the original files are not affected). Before compression, a checksum is calculated for each library to be later compared with the checksum of the file extracted to the file system.</p> <p>For more information, see the “Environment Variables” section in this chapter.</p>	Automatically as a task in the Workflow	For example, an XML file describing the layout of the accumulator buffer that is passed between the Usage Query Server and the Usage Query Proxy is inserted into the database.

Regular Option #	Detailed Option #	Step	Description	Performed by	Comments
5	20	Applying the Reference database updates	This step involves executing the SQL scripts for updating reference tables (such as the mapping tables), which were generated by the Implementation Compiler, to update the master database. The ADJ1_MPR_PRODUCTS table is also updated in this step.	Automatically as a task in the Workflow	The database that is updated is the master database that served as the source for the input data. On Windows NT, it is possible to apply changes to the database in parallel with the build and distribution steps by setting the MPRFLOW_SQL_IN_PARALLEL environment variable.
6	21	Distributing the libraries	This step involves distributing the newly generated libraries to the defined target locations. If the locations were not defined in the ADJ1_DISTRIBUTION_TARGET table as part of the prerequisite steps, this step does not do any work.	Automatically as a task in the Workflow	The target location can be on the current machine or on a remote machine. The distribution process reports the status of the distribution to each target. Alternatively, the libraries can be distributed automatically through the Release Distribution map. For more information, see the “UTL1MULTILEXT– Multiple Library Extract” chapter.
N/A	22	Applying the Usage database updates (optional)	This step involves executing the SQL scripts for upgrading the structure of the Rated Events, Rejected Events, and Accumulators tables.	Manually or automatically as a task in the Workflow	On Windows NT, it is possible to apply changes to the database in parallel with the build and distribution steps by setting the MPRFLOW_SQL_IN_PARALLEL environment variable.

Implementation Compiler Workflow Behavior

This section describes various aspects of the behavior of the Implementation Compiler Workflow.

Automated Prerequisite Check

The Implementation Compiler Workflow enables the operator to invoke custom prerequisite scripts before running the Workflow itself. For example, such scripts can be used to ensure that specific third-party software is installed in the environment.

By default, the Implementation Compiler Workflow calls the `INF1_IRC_prerun_validation.pl` script. In addition to performing other initialization steps, this script validates the environment in which the Implementation Compiler Workflow is to run prior to its execution, for all the running options. This script is invoked in Unix environments only.

The script returns the status of the validation (exit 1 in the case of an error) to the main Workflow. The flow continues only if the validation was successful; otherwise, it is terminated like in other error cases.

The validation script can be disabled by setting the `RUN_VALIDATION_MODE` environment variable to ‘false’ in the `ADJ1ICWFlow_setenv.ksh` or `.bat` file in the working directory.

Auto-Retry Mechanism

If a build failure occurs in an NT environment, the Auto-Retry mechanism checks the logs to see whether the failure occurred because of a common IncrediBuild or Visual Studio error. If it did, the mechanism reruns the build step automatically.

Automated Data Validation

The Implementation Compiler Workflow lets the operator enable or disable data validation for complex attributes before running the Event Server.

This validation calculates the actual size of the String representation of a complex attribute (including delimiters) and compares it to the persistence size for the complex attribute (the maximum size defined for it in the database). If the actual size is larger, the Event Server throws an exception, rejects the event, and continues processing other events as usual.

Because this validation affects performance, it can be disabled by setting the `COMPLEX_SIZE_VALIDATION_ENABLED` environment variable to ‘false’ in the `ADJ1ICWFlow_setenv.ksh` or `.bat` file in the working directory (recommended for production environments). After the value of the parameter is changed, the Implementation Compiler Workflow must be run again.

Environment Variables

The following environment variables affect the process.



Note: This table does not include environment variables that define the classpath of the application.

(*) All of these environment variables can be set in the ADJ1ICWFlow_setenv.ksh file located in the working directory that is passed in the command line to the Workflow.

Most of the environment variables below have default values, which are set automatically according to the running environment type. Therefore, in most cases, they must not be modified or customized.

Name	Description	Default Value	Method of Changing the Variable
Environment variables related to the set-up of the working environment			
GAJTMPPR_FLOW_RESOURCES	The resource directory of the Implementation Compiler Workflow, which holds properties files and other static input needed to run the Workflow.	N/A	See (*) above.
MPR_ANT_SCRIPT_DIR	The directory that contains the generic target Ant build files of the Implementation Compiler.	N/A	N/A
MPR_FLOW_ROOT_DIR	The location of the Implementation Compiler Workflow environment.	N/A	See (*) above.
MPRFLOW_BUILD_INCLUDE_DIR	The directory that contains the include tar files (headers for C++ include files).	Set automatically, according to the running environment type (for example, RT)	See (*) above.
MPRFLOW_BUILD_KIT_DIR	The directory that contains the build kit.	Set automatically, according to the running environment type (for example, RT)	See (*) above.

Name	Description	Default Value	Method of Changing the Variable
Environment variables that determine to which database the process must connect to refresh the Rating Logic Configurator and Implementation Compiler files (input files for the Implementation Compiler). These variables are automatically set when the –db option is used in the command line. They can be customized via the local setenv script.			
MPR_DB_INSTANCE	The database instance.	N/A	See (*) above.
MPR_DB_PASS	The database password.	N/A	See (*) above.
MPR_DB_PORT	The database port.	1521	See (*) above.
MPR_DB_SERVER	The database server.	N/A	See (*) above.
MPR_DB_SERVICE	The service name (taken from the tnsnames.ora). This environment variable must be specified when connecting to the Oracle RAC. It can also be used when connecting to a regular database. If the service is defined, the values of the environment variables defining the server, instance, and port are ignored.	N/A	See (*) above.
MPR_DB_USER	The database user name.	N/A	See (*) above.
Environment variables for configuring the execution of the Implementation Compiler			
MPR_GEN_TYPE	The generation type. Valid values: <ul style="list-style-type: none">■ Production■ Trace■ All	ALL	See (*) above.
MPR_LIB_NAMING_POLICY	The naming policy for generated libraries (indicates whether the version is appended to the name). Valid values: <ul style="list-style-type: none">■ <i>dev</i> – The library name does not contain the version.■ <i>prod</i> – The library name contains the version.	In the runtime environment, the default value is ‘prod’; in the development environment, the default is ‘dev’. For libraries that are to be loaded in a production environment, the naming policy must always be set to ‘prod’.	See (*) above.
MPR_OUTPUT_DIR	The location of the generated files.	< <i>workdir</i> >/mpr/mpr_runtime/out	See (*) above.

Name	Description	Default Value	Method of Changing the Variable
MPR_RT_CLEAN_METADATA	Determines whether to clean the metadata table before running the reference SQL scripts. This variable is to be set to ‘true’ only in Windows.	false	See (*) above.
MPR_RT_CLEAN_TYPE	The area of the runtime environment that is to be cleaned before execution. Valid values: <ul style="list-style-type: none"> ■ <i>RT</i> – All runtime data ■ <i>REG</i> – Registry ■ <i>RLC</i> – Rating Logic Configurator data ■ <i>MF</i> – Manifest files ■ <i>OUTPUT</i> – The output directory for generated files More than one value can be specified, if necessary.	In Full mode, the value is set to ‘OUTPUT’ by default.	See (*) above.
MPR_RT_CMD_ARG_MANIFEST	The location of additional manifest files to be imported into the Implementation Compiler runtime environment. The directories are imported in the order of appearance.  <i>Note: This variable is not required if “Procedure 2” is used to generate and compile an implementation with a custom extension function.</i>	N/A	See (*) above.
MPR_RT_CMD_ARG_REFRESH	Indicates which input files for the Implementation Compiler are to be refreshed from the database. Valid values: <ul style="list-style-type: none"> ■ <i>RLC</i> – Only the Rating Logic Configurator implementation files ■ <i>IC</i> – Only the Implementation Compiler files ■ <i>CONFIG</i> – Only the Implementation Compiler configuration file ■ <i>ALL</i> – All files 	ALL	See (*) above.

Name	Description	Default Value	Method of Changing the Variable
MPR_RT_CMD_ARG_REGISTRY	The location of the external registry to be imported into the Implementation Compiler runtime environment.	N/A	See (*) above.
MPR_RT_CMD_GEN_THREADS_COUNT	The number of threads used in the generation phase. If this variable is set to 1, generation is performed in the normal, non-parallel mode. Recommended values are: <ul style="list-style-type: none">■ <i>NT</i> – 2■ <i>Unix</i> – 8	<ul style="list-style-type: none">■ <i>NT</i> – 2■ <i>Unix</i> – 8	See (*) above.
MPR_RT_CMD_WORKFLOW_MODE	Indicates whether a new registry is created if a registry file is not found in the file system. Valid values: <ul style="list-style-type: none">■ <i>REG</i> – A new empty file is created.■ <i>PROTECTED</i> – An exception is thrown.	REG	See (*) above.
MPR_RT_DIR	The root directory of the Implementation Compiler runtime environment (the generation environment). This variable is <i>mandatory</i> .	< <i>workdir</i> >/mpr/mpr_runtime	See (*) above.
MPR_RT_LOG_DIR	The location of the log directory.	< <i>workdir</i> >/mpr/mpr_runtime/logs	See (*) above.
MPR_RT_PKG_ID	The specific package ID for which to refresh the data.	If this variable is not specified, and a refresh is requested, the active package ID is used.	See (*) above.
MPRFLOW_SQL_IN_PARALLEL	For Windows NT only. Indicates whether the SQL scripts must run in parallel with the build stage. Valid values (lowercase): <ul style="list-style-type: none">■ true■ false	false	See (*) above

Name	Description	Default Value	Method of Changing the Variable
Environment variables for the build and its validation			
CPF_TC_MAIN_SLEEP_INTERVAL	The sleep time for the Rater Init.	0	See (*) above.
IGNORE_ERRORS_IN_INIT	The number of errors in the initialization process to be displayed in the log.	1	See (*) above.
MPR_FLOW_DEBUG_PORT	The port for the various other flows in the Implementation Compiler Workflow, such as the File Comparator flow or the Release Info flow. This port is used for debugging from a remote machine.	N	See (*) above.
MPR_IC_JVM_ADD_ARGS	Additional JVM arguments for the Implementation Compiler runtime process.	N/A	See (*) above.
MPR_IC_JVM_ARG_HEAP_SIZE_MB	The heap memory size of the Implementation Compiler JVM in megabytes.	512	See (*) above.
MPR_IC_JVM_DEBUG_PORT	The port for the Implementation Compiler runtime process. This port is used for debugging from a remote machine.	N	See (*) above.
MPR_USE_CPPCOMB	Indicates whether the Implementation Compiler must generate the object and the dependency file for each CPP source in one compilation. This option reduces the time required for a build.	Y	See (*) above. To remove this flag, change export MPR_USE_CPPCOMB=Y to export MPR_USE_CPPCOMB=.
MPRFLOW_BUILD_MIGRATION	Indicates whether Migration libraries (libmpr_piMigration.so and libmpr_eventMigration.so) must be included in the build. Accumulator and Event Migration is a one-time process that converts accumulators and events from earlier Amdocs Billing versions that did not include Turbo Charging to the Turbo Charging system.	false	See (*) above.

Name	Description	Default Value	Method of Changing the Variable
MPRFLOW_BUILD_NUM_OF_FILES_IN_GROUP_<name of the library directory>	The number of files in the specified group.	20	See (*) above.
MPRFLOW_BUILD_PRODUCT_TYPE	The product type. Valid values (lowercase): <ul style="list-style-type: none"> ■ trace ■ production 	trace	See (*) above.
MPRFLOW_BUILD_VARIANT	The build variant.	64	See (*) above.
MPRFLOW_COMPARISON_THREADSCOUNT	Enables customizing the number of threads used by the File Comparator.	10	See (*) above.
MPRFLOW_COMPILE_PARALLEL_PROCESSES	The number of concurrent processes in the compilation stage.	8	See (*) above.
MPRFLOW_LIB_NAME	The name of the library to invoke for Rater Init.	Mpr_dd	See (*) above.
MPRFLOW_TRACE_FLAG	Indicates whether the code required for event and protocol tracing must be generated. Valid values (lowercase): <ul style="list-style-type: none"> ■ true ■ false 	true	See (*) above.
MRPFLOW_RATER_INIT_INSTANCE	The instance of the database connection used to execute Rater Init.	Points to the MPR_DB_INSTANCE environment variable.	See (*) above.
MRPFLOW_RATER_INIT_PASS	The password for the database connection used to execute Rater Init.	Points to the MPR_DB_PASS environment variable.	See (*) above.
MRPFLOW_RATER_INIT_USER	The user name for the database connection used to execute Rater Init.	Points to the MPR_DB_USER environment variable.	See (*) above.
MPRFLOW_USE_INCREDBUILD	Indicates whether the Implementation Compiler Workflow must use InrediBuild.	true	See (*) above.

Name	Description	Default Value	Method of Changing the Variable
Environment variables related to custom extension functions			
CUST_INCLUDE_PATH	<p>The path to the header of the custom extension function. The environment variable can contain several paths. For the complete syntax, see the “Using Custom Extension Functions” section in this chapter.</p> <p> Note: <i>This variable is not required if “Procedure 2” is used to generate and compile an implementation with a custom extension function.</i></p>	None	See (*) above.
Environment variables related to the distribution process			
MPR_DISTRIB_INCLUDE_LIB_AND_SRC	Defines whether the libraries and sources of the Implementation Compiler must be packed in the distribution phrase.	true	See (*) above.
MPRFLOW DISTRIBUTION_DB_INSTANCE	The instance of the database that contains information about the target for delivering the products.	MPRFLOW_USG_DB_INSTANCE	See (*) above.
MPRFLOW DISTRIBUTION_DB_PASS	The password for the database that contains information about the target for delivering the products.	MPRFLOW_USG_DB_PASS	See (*) above.
MPRFLOW DISTRIBUTION_DB_PORT	The port of the database server for the database that contains information about the target for delivering the products.	MPRFLOW_USG_DB_PORT	See (*) above.

Name	Description	Default Value	Method of Changing the Variable
MPRFLOW_DISTRIBUTION_DB_SERVER	The name of the database server for the database that contains information about the target for delivering the products.	MPRFLOW_USG_DB_SERVER	See (*) above.
MPRFLOW_DISTRIBUTION_DB_SERVICE	The service name (taken from the tnsnames.ora). This environment variable must be specified when connecting to the Oracle RAC. It can also be used when connecting to a regular database. If the service is defined, the values of the environment variables defining the server, instance, and port are ignored.	MPRFLOW_USG_DB_SERVICE	See (*) above.
MPRFLOW_DISTRIBUTION_DB_USER	The user name for the database that contains information about the target for delivering the products.	MPRFLOW_USG_DB_USER	See (*) above.
MPRFLOW_DISTRIB_LIB_DIR_FROM	A custom location of the libraries to be copied to the distribution area.	<workdir>/mpr/mpr_runtime/output/	See (*) above.
MPRFLOW_DISTRIB_LIB_DIR_TO	A custom location in the distribution area to which the libraries must be copied.	<workdir>/distribution/\${MPRFLOW_RELEASE_VERSION}/\${MPRFLOW_BUILD_VARIANT}/\${product type}/products/lib	See (*) above
MPRFLOW_DISTRIB_LIB_PATTERN	The pattern that is used to recognize compiled libraries.	‘*’ – All files.	See (*) above.
MPRFLOW_DISTRIB_RESOURCES_DIR_FROM	A custom location of the resources (in SQL and XML format) to be copied to the release area.	<workdir>/mpr/mpr_runtime/output/	See (*) above.
MPRFLOW_DISTRIB_RESOURCES_DIR_TO	A custom location in the release area to which the resources must be copied.	<workdir>/distribution/\${MPRFLOW_RELEASE_VERSION}/\${MPRFLOW_BUILD_VARIANT}	See (*) above.
MPRFLOW_DISTRIB_SOURCES_DIR_FROM	A custom location of the sources used for the release to be copied to the distribution area.	<workdir>/build/\${variant}/\${product type}/sources/mpr_generated	See (*) above.

Name	Description	Default Value	Method of Changing the Variable
MPRFLOW_DISTRIB_SOURCES_DIR_TO	A custom location in the distribution area to which the sources must be copied.	<workdir>/distribution/\${MPRFLOW_RELEASE_VERSION}/\${MPRFLOW_BUILD_VARIANT}/\${product type}/sources	See (*) above.
MPRFLOW_RELEASE_VERSION	A custom release version to use. This variable must be populated by the operator if the STANDARD or EXTENDED preset failed or if the operator manages the operations executed by the Implementation Compiler Workflow.	A sequential number beginning from 1.	See (*) above.
Environment variables related to inserting the generated products			
MPRFLOW_NUMBER_OF VERSIONS_TO_KEEP	The number of versions for which the library files must be kept in the ADJ1_MPR_PRODUCTS table. After inserting or updating the files in the table, the Implementation Compiler Workflow deletes the records that contain old libraries. The ADJ1_MPR_PRODUCTS table is cleaned up even if libraries are not stored in it. Therefore, in environments for which MPRFLOW_STORE_PRODUCTS=false and for which clean-up is not needed, the MPRFLOW_NUMBER_OF VERSIONS_TO_KEEP variable must be set to 0. Otherwise, clean-up is enabled, and only two previous versions are retained.	2	See (*) above.
MPRFLOW_PRODUCT_DB_INSTANCE	The instance of the database where the ADJ1_MPR_PRODUCTS table is to be updated.	MPR_DB_INSTANCE	See (*) above.
MPRFLOW_PRODUCT_DB_PASS	The password for the database where the ADJ1_MPR_PRODUCTS table is to be updated.	MPR_DB_PASS	See (*) above.
MPRFLOW_PRODUCT_DB_PORT	The port of the database server of the database where the ADJ1_MPR_PRODUCTS table is to be updated.	MPR_DB_PORT	See (*) above.

Name	Description	Default Value	Method of Changing the Variable
MPRFLOW_PRODUCT_DB_SERVER	The name of the database server of the database where the ADJ1_MPR_PRODUCTS table is to be updated.	MPR_DB_SERVER	See (*) above.
MPRFLOW_PRODUCT_DB_SERVICE	The service name (taken from the tnsnames.ora). This environment variable must be specified when connecting to the Oracle RAC. It can also be used when connecting to a regular database. If the service is defined, the values of the environment variables defining the server, instance, and port are ignored.	MPR_DB_SERVICE	See (*) above.
MPRFLOW_PRODUCT_DB_USER	The user name for the database where the ADJ1_MPR_PRODUCTS table is to be updated.	MPR_DB_USER	See (*) above.

Name	Description	Default Value	Method of Changing the Variable
MPRFLOW_STORE_PRODUCTS	<p>Determines the types of data that is stored in the ADJ1_MPR_PRODUCTS table. Valid values:</p> <ul style="list-style-type: none"> ■ <i>false (default)</i> – The following files are stored in the table: <ul style="list-style-type: none"> • Generated XML and SQL files (one file per record) • Generated source code (all the code in one record) • Manifest files (all the files in one record) • Registry file • Configuration file (.icc) ■ <i>true</i> – Generated libraries (one file per record) are stored in addition to the files that are stored in the table by default. <p>The ADJ1_MPR_PRODUCTS table is cleaned up even if libraries are not stored in it. Therefore, in environments for which MPRFLOW_STORE_PRODUCTS=false and for which clean-up is not needed, the MPRFLOW_NUMBER_OF VERSIONS_TO_K EEP variable must be set to 0. Otherwise, clean-up is enabled, and only two previous versions are retained.</p>	false	See (*) above.

Environment variables related to changes in the reference database

MPRFLOW_REF_DB_INSTANCE	The instance of the database where the reference table is to be updated.	MPR_DB_INSTANCE	See (*) above.
MPRFLOW_REF_DB_PASS	The password for the database where the reference table is to be updated.	MPR_DB_PASS	See (*) above.
MPRFLOW_REF_DB_PORT	The port of the database server of the database where the reference table is to be updated.	MPR_DB_PORT	See (*) above.
MPRFLOW_REF_DB_SERVER	The name of the database server of the database where the reference table is to be updated.	MPR_DB_SERVER	See (*) above.

Name	Description	Default Value	Method of Changing the Variable
MPRFLOW_REF_DB_SERVICE	The service name (taken from the tnsnames.ora). This environment variable must be specified when connecting to the Oracle RAC. It can also be used when connecting to a regular database. If the service is defined, the values of the environment variables defining the server, instance, and port are ignored.	MPR_DB_SERVICE	See (*) above.
MPRFLOW_REF_DB_USER	The user name for the database where the reference table is to be updated.	MPR_DB_USER	See (*) above.
MPRFLOW_SQL_REF_DIR	A custom directory that contains the SQL files to be run on the reference database. By default, this variable is set to the corresponding folder in the release area.	<workdir>/distribution/\${MPRFLOW_RELEASE_VERSION}/\${MPRFLOW_BUILD_VARIANT}/	See (*) above.
Environment variables related to changes in the usage tables			
MPRFLOW_USG_DB_INSTANCE	The instance of the database where the usage tables are to be updated.	N/A	See (*) above.
MPRFLOW_USG_DB_PASS	The password for the database where the usage tables are to be updated.	N/A	See (*) above.
MPRFLOW_USG_DB_PORT	The port of the database server of the database where the usage tables are to be updated.	N/A	See (*) above.
MPRFLOW_USG_DB_SERVER	The name of the database server of the database where the usage tables are to be updated.	N/A	See (*) above.
MPRFLOW_USG_DB_SERVICE	The service name (taken from the tnsnames.ora). This environment variable must be specified when connecting to the Oracle RAC. It can also be used when connecting to a regular database. If the service is defined, the values of the environment variables defining the server, instance, and port are ignored.	N/A	See (*) above.
MPRFLOW_USG_DB_USER	The user name for the database where the application tables are to be updated.	N/A	See (*) above.

Name	Description	Default Value	Method of Changing the Variable
General environment variables			
COMPLEX_SIZE_VALIDATION_ENABLED	Defines whether the complex size validation must run. Valid values: <ul style="list-style-type: none">■ <i>true</i> – Recommended for non-production environments■ <i>false</i> – Recommended for production environments	true	See (*) above.
GAVM_PATH	The path to the libraries of the Availability Manager.	ADJUST_BIN_LOCATION	See (*) above.
MPR_MEMORY_PRINT_FREQUENCY	The rate, in seconds, at which memory usage is sampled and printed to the log.	By default, the value is not set (nothing is printed).	See (*) above.
MPR_PRINT_OCI_BUFFER	Defines whether the OCI buffer must be printed to the log in the case of a problem.	N/A	See (*) above.
RUN_VALIDATION_MODE	Defines whether the validation script must run. Valid values: <ul style="list-style-type: none">■ <i>true</i>■ <i>false</i>	true	See (*) above.
TCL_HOME	The path to the third-party TCL libraries.	%THIRD_PARTIES_LOCATION%\\tcl8.4.16_VS2005	See (*) above.
TMPDIR	The directory for temporary files.	If the value of this environment variable is not defined, the default temporary directory of the system is used.	See (*) above.

Parameters

Parameters are set via the environment variables described in the “Environment Variables” section in this chapter.

Configuration Parameters

The process uses Light Tracer configuration parameters, which are defined in the ADJ1_CONF_SECTION_PARAM table. For more information, see [Light Tracer Parameters](#) in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Configuration Files

The configuration file of the Implementation Compiler is one of the input files required for the successful execution of the Workflow. It is an XML file with an .icc extension that is located in the root folder of the generation area (defined in the MPR_RT_ROOT_DIR_ARG environment variable).

This configuration file is obtained at runtime from the file system. If this file does not exist, the Implementation Compiler creates and uses a default one.

For more information about the .icc file, see *Turbo Charging Implementation Compiler XML Configuration Guide*.

In addition, the Implementation Compiler uses a number of properties files that contain some configuration. These files must not be changed.

Database Connections

The process connects to the following table during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Reference	APE1_XML_DISTRIB (configurable through the Implementation Compiler’s properties file)	Read-only

Admin Commands

The Implementation Compiler Workflow supports the following admin commands. For information on how to execute the commands, see the “[Handling Admin Commands](#)” section in the “[High Availability](#)” chapter.

Command	Description	Parameters
SET_LIGHT_TRACER_COMMAND	Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters	<ul style="list-style-type: none"> ▪ Activation parameters: <ul style="list-style-type: none"> • <i>turnTracer “on/off”</i> – Activates or stops the Light Tracer • <i>--turnAssert “on/off”</i> – Activates or stops Assertions • <i>--turnLogOnTrace “on/off”</i> – Activates or stops writing log messages in Light Tracer files • <i>--turnFlowTracer/turnAll “on/off”</i> – Activates or stops tracing a single flow or event that is transferred between machines • <i>--turnAll “on/off”</i> – Activates or stops all of the above. ▪ The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter). ▪ One of the following sets of parameters specifying the module and the submodule: <ul style="list-style-type: none"> • Module ID only: <ul style="list-style-type: none"> ▫ <i>--moduleId 63</i> – Applies the activation command to the Implementation Compiler Workflow, including all libraries • Both module ID and submodule ID: <ul style="list-style-type: none"> ▫ <i>--moduleId 63</i> – Applies the activation command to the Implementation Compiler Workflow ▫ <i>--subModuleIDs “<n>”</i> – Applies the activation command to a specific library. The name of each library and the corresponding subModuleID appears in the XML file generated for the Light Tracer and in the registry. ▪ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • <i>--threadInfo “Y/N”</i> – Specifies whether trace or assertion messages include the thread information. • <i>--timestampInd “Y/N”</i> – Specifies whether trace or assertion messages include the time stamp.

Command	Description	Parameters
		<ul style="list-style-type: none"> • <code>--contextInd "Y/N"</code> – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • <code>--noOfMessages "<number>"</code> – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down. • <code>--period "5/10/20/60/120/480/1440"</code> – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated. <p> Note: If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached.</p> <ul style="list-style-type: none"> • <code>--assertAbortInd "Y/N"</code> – Specifies whether the application is to be aborted if the assertion condition fails. ▪ <code>--commandType "changeBufferSize"</code> – Resizes the Light Tracer buffer at run time. • <code>--bufferSize "<number>"</code> – The new size for the Light Tracer buffer.
SWITCH_LIGHT_TRACER_ON ALL	Activates the Light Tracer without the ability to specify the mode or send parameters	N/A
SWITCH_LIGHT_TRACER_OFF ALL	Stops the Light Tracer	N/A

Examples:

- `ADJ1_Send_Admin_Command_Sh hpbl820 ADJ1ICWFlow SET_LIGHT_TRACER_COMMAND --turnTracer "on" --moduleID 63 --threadInfo "y"`
- `ADJ1_Send_Admin_Command_Sh hpbl820 ADJ1ICWFlow SWITCH_LIGHT_TRACER_ON ALL`

Distribution

The output files of the Implementation Compiler are processed by a build process into runtime compiled libraries. Those libraries are distributed to the Event Servers.

If the libraries are stored in the same data source (the ADJ1_MPR_PRODUCTS table) as reference data (depending on the MPRFLOW_STORE_PRODUCTS environment variable), the libraries are extracted from the updated data source as part of the Release Distribution map. The runtime environment starts using the updated data source only after the extraction is finished and validated successfully. For more information, see the “UTL1MULTILEXT– Multiple Library Extract” chapter.

Screen Messages

There are no special screen messages. Log message are also sent to the console, but they are best viewed in the log file.

Troubleshooting

The following important or frequent errors can occur during the process run.

Error	Preventing Action
Missing manifest files	Make sure that you have the latest manifest files from the Configuration Control system.
No database connection	Verify that the database details that are passed as environment variables point to a working database that contains the required table. Make sure that the user name and password are correct.

Recovery Instructions

After the problem is fixed, rerun the process with the same configuration.

Important Remarks

This section explains how to deploy customization extension functions and provide private fixes.

Using Custom Extension Functions

To generate and compile an implementation with a custom extension function correctly, perform one of the following procedures:

- Put the customization headers and manifest files in a folder and use environment variables to point the Implementation Compiler Workflow to them. For more information, see “[Procedure 1](#).”
- Create JAR files that enable option 1 of the Implementation Compiler Workflow to deploy the customization extension functions. For more information, see “[Procedure 2](#).”

The procedure to be used depends on the change that is to be deployed. For example, if a manifest file or a header file must be added, there is no need to repack the JAR files, and the first procedure is preferable.

Procedure 1

To generate and compile an implementation with a custom extension function using environment variables, perform the following:

1. Copy the customized *.h files to any location (for example, <workdir>/build/include/custom/<your custom folder>).
2. Add the path to the new header files by setting the CUST_INCLUDE_PATH environment variable.

- *Unix*

```
export CUST_INCLUDE_PATH="-I${custom location of headers}"
```

- *Windows*

```
set CUST_INCLUDE_PATH=<include path 1>;<include path 2>
```

3. Add the path and the new shared library.

- *Unix*

```
export CUST_LINKDLL_DIRS="-L${LIBPATH1} -L${LIBPATH2}"
```



Note: Only the directories are set.

```
export CUST_LINKDLL_POST_LIBS="-l${LIB_NAME1} -l${LIB_NAME2}"
```



Note: Only the library name without the extension is set.

- *Windows*

```
set CUST_LINKDLL_DIRS=libpath1;libpath2
```



Note: Only the directories are set, separated by semicolons.

```
set CUST_LINKDLL_POST_LIBS=libname1.lib libname2.lib
```



Note: The library names here include the extension and are separated by spaces.

4. Perform the export in the ADJ1ICWFlow_setenv.ksh script (or the extension .bat file in Windows) in your work directory.

For more information on environment variables, see the “[Environment Variables](#)” section in this chapter.

Procedure 2

To generate and compile an implementation with a custom extension function using JAR files, perform the following:



Note: Use this procedure only in a Unix environment per customization level. The procedure below uses customization layer 9 as an example.

1. Prepare two JAR files:
 - *ADJ9IC_CPP_IncFiles.jar* – This file must include the C++ header files and the relative folder path.
 - *ADJ9IC_manifestFiles.jar* – This file must include the customization manifest files and the relative folder path.
2. Place these JAR files in one of the following locations:
 - \${ABP_PRIVATE_BIN}
 - \${ABP_APP_CUSTOM_BIN}
 - \${ABP_CUSTOM_BIN}
3. To deploy the JAR files, run the Implementation Compiler Workflow with option 1 or 12.

For more information about the options of the Implementation Compiler Workflow, see the “[Flow](#)” section in this chapter.

For information on how to create a custom extension functions, see *Turbo Charging Extension Functions Programmer Guide*.

Providing Private Fixes

The following procedure creates a copy of the Implementation Compiler Workflow environment with the same input data and sources, so that developers can modify the sources and compile new libraries without modifying the original Implementation Compiler Workflow environment.

To provide compiled private fixes for the Implementation Compiler Workflow without affecting the master database:

1. Run the following script:

PCI1_Create_MapperPackage <Source Path> <Destination Path>

Example:

```
CreatePackage_ksh /mprhome2/mpr/Mapper_1013  
/vfrinffhome2/vfr/aim/for_irc/vfrwrk43
```

This script creates a TAR package for the Implementation Compiler Workflow.

2. Run the following script:

**PCI1_Modify_ADJ1ICWFlow_setenv_ksh <ICWFlow Home> <Build Number>
<CC Version> <CC Home>**

Example:

```
PCI1_Modify_ADJ1ICWFlow_setenv_ksh/vfrinffhome2/vfr/aim/for_irc/vfrwrk43  
179 1013/vfrhome/vfr/ccvfr(mb_ccvfr
```

This script extracts the TAR file and prepares the ADJ1ICWFlow_setenv.ksh file, which defines a new working directory that can be used to create private libraries.

3 ADJ1EVENTSRV – Event Server

This chapter describes the Event Server process.

Description

The Event Server is the main processing daemon of the Turbo Charging event processing platform. It is comprised of a daemon running a set of functions, and is defined and deployed according to specific installation requirements.

Separate instances of the Event Server can be configured differently without any additional coding.

Specific functions can be customized. As long as the relationships among the various functions remain the same, customization does not alter the flow or functionality of other functions.

Process Type

Daemon

Run Frequency

Runs continuously

Participation in Generic Maps

This section contains information on the generic maps in which the process participates.

Map Name

- End-of-Cycle (EOC), as part of Rerate
- Undo, as part of Rerate

For more information about these maps, see *Amdocs Billing Operator Guide*.

Severity Level

Severity Level 2 – The map must hold until the process is fixed.

Activation

Command Line:	ADJ1_ES_Daemon_Shell_Sh -n <APPLICATION_ID> -c "-f <CCF_FILE_NAME>" -e <EWD_EXE>  Note: The -e parameter is optional. It is used in environments with Availability Manager.
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_ES_Daemon_Shell_Sh
Executable Name:	gcpf1fwcApp

Example:

ADJ1_ES_Daemon_Shell_Sh -n ES100 -c "-f complete_ES_TCP_20.ccf" -e gn1avm_ewd

For more information about the activation scripts, see the “[Scripts](#)” section in the “[Introduction](#)” chapter.

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> ES \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

```
ADJ1_Send_Admin_Command_Sh hpbl820 ES 100
CPF_GracefulShutdownCommand
```

Preceding Processes

In the End-of-Cycle and Undo maps, the Event Server and the Rerate Prepare for End of Cycle processes function as a pair that is responsible for rerating. Therefore, in these maps, the Event Server shares the dependencies of the Rerate Prepare for End of Cycle process.

In addition, each process group (active and shadow processes) works with standalone shared memory. The shared memory key for each process is allocated dynamically using the memdbkey utility, which is called from the start-up script. This utility is described in the following section.

Shared Memory Space Allocation

The memdbkey utility creates a unique key that is used for shared memory space allocation. This ID is known to both active and shadow processes through an environment variable set by the utility. Distinct shared memory IDs enable the following:

- Several Event Server processes in the same environment
- Different process types in the same environment
- Several environments on the same machine

The utility is activated as follows:

```
memdbkey <Option> [{<Parameter>}]
```

The following options and their related parameters are available:

- **-h** – Show help.
- **-v** – Show the version of the utility.
- **-n <process name>** – Create the shared memory key according to the process name, for example, ES100.

- `-n <process name> -c` – If the key already exists for a particular process, clean it up. This option is to be used when both the active and the shadow processes are down.
- `-g <group name>` – Create the shared memory key according to the group name, for example, ES100.
- `-g <group name> -c` – If the key already exists for a particular group, clean it up. This option is to be used when both the active and the shadow processes are down.
- `-k <shared memory key>` – Use a user-defined key.
- `-k <shared memory key> -c` – Remove the key provided by the user. This option is to be used when both the active and the shadow processes are down.
- `-a` – Remove all shared memory arenas of the user. This option returns the number of removed shared memory arenas. If deletion failed, the value of -1 or 255 is returned. Because the number of arenas for a user is usually less than 200, the value of 255 always means failure and not the number of removed arenas.



Note: If any Event Server shared memory segments exist, the removal of \$ABP_APARTMENT_ROOT/shm_keys_holder directory or its contents is not allowed.

The memdbkey utility executes the following flow:

1. Retrieves the value of the \$ADJ1_SHMEM_PATH environment variable. If this environment variable is not defined or the directory does not exist, uses \$ABP_APARTMENT_ROOT. If this environment variable is not defined or the directory does not exist, uses \$HOME. The user must have write permissions to the folder defined by the variable.
2. Retrieves the value of the \$SHM_KEYS_HOLDER environment variable.
3. Retrieves the user name of the environment where the script is run (for example, adjwrk5) from the \$USERNAME or \$USER environment variable.
4. Does the following:
 - If the process group is empty, retrieves the process group ID from the \$PROCESS_GROUP_ID environment variable and process type ID from the \$PROCESS_TYPE_ID environment variable.
If at least one of these environment variables is not found, memdbkey tries to retrieve the data from the database:
 - i. Retrieves the following environment variables:
 - ◆ ADJCONFIG_DB_USER
 - ◆ ADJCONFIG_DB_PASSWORD
 - ◆ ADJCONFIG_DB_INSTANCE

If one of these variables does not exist, memdbkey exits with failure.

- ii. Retrieves the data from the GN1_SYS_PROC_INSTANCE_CFG table using the process name from the PROCESS_CODE column as input.
If the data does not exist in the database, memdbkey exits with failure.
 - If the process group is supplied, uses ‘1’ as the process type ID.
5. Concatenates the results of the previous steps to create a file name.

Example:

If the results of steps 1–4 are:

- (1) = “\usr\abp_home”
- (2) = “shm_keys_holder”
- (3) = “adjwrk5”
- (4) = 1000 (group) and 1 (Event Server)

the file name is

\usr\abp_home\shm_keys_holder\TC_SHMTOK1_adjwrk5_1_1000.

6. Checks whether the file exists, and creates it if it does not.
7. Generates a unique key using an ftok system call.



Note: If a file is deleted and then re-created, ftok may return a different key than the original one.

8. Sets the internal \$ADJ1_SH_MEM_ID environment variable to be equal to the generated key.
9. The Event Server performs GETENV on the \$ADJ1_SH_MEM_ID environment variable and sends the result to the memory module. If this variable is not defined, the Event Server works with virtual memory.

Dependent Processes

In the End-of-Cycle and Undo maps, the Event Server and the Rerate Prepare for End of Cycle processes function as a pair that is responsible for rerating. Therefore, in these maps, every process that retrieves updated data from the Events or Accumulators table depends on this process.

In addition, the following utilities can be used for operations on the shared memory of the Event Server.

Memory Dump

The AIMOS dump utility enables moving the contents of the shared memory to a series of files and later loading these files into AIMOS for analysis or reuse. For more information on this utility, see the “memdbdump – AIMOS Dump” chapter.

Memory Analyzer

Due to the nature of AIMOS, its data cannot be viewed by external tools. The AIMOS MetaInfo tool enables the user to view information regarding AIMOS as well as the information stored within AIMOS. For more information on this tool, see the “[metainfo – AIMOS MetaInfo](#)” chapter.

Memory Pool Definitions

The ADJ1_AllocatedSharedMemoryBlocks_Sh script calculates the shared memory pool inflation sizes and writes them into a file. It is run by the Availability Manager on the failed host during a host failover, prior to cleaning the shared memory. When the replication host assumes master responsibilities, the Event Server performs the following:

- Under the following conditions, reads the pool inflation sizes from the file:
 - The file exists.
 - The value of the SHM_POOL_INFILATION_SET_OVERRIDE_FROM_FILE parameter in the APR1_CONF_SECTION_PARAM table is set to ‘TRUE’.
 - The calculated pool inflation size from the file is greater than the threshold defined in the SHM_POOL_INFILATION_LOW_WATER_MARK_THRESHOLD parameter in the APR1_CONF_SECTION_PARAM table.
- In the following cases, the Event Server uses the memory pool inflation sizes defined in the SHM_POOL_INFILATION_SET parameter in the APR1_CONF_SECTION_PARAM table with the relevant PARAM_CLASS:
 - The file does not exist.
 - The file exists but cannot be read.
 - The value of the SHM_POOL_INFILATION_SET_OVERRIDE_FROM_FILE parameter in the APR1_CONF_SECTION_PARAM table is set to ‘FALSE’.
 - The value of the SHM_POOL_INFILATION_SET_OVERRIDE_FROM_FILE parameter in the APR1_CONF_SECTION_PARAM table is set to ‘TRUE’, but the calculated pool inflation size from the file is smaller than the threshold defined in the SHM_POOL_INFILATION_LOW_WATER_MARK_THRESHOLD parameter in the APR1_CONF_SECTION_PARAM table.

The name of the file containing shared memory pool definitions is unique per process group. It uses the following naming convention:

PoolInfSet_<process group>.txt

The file path is configured using the SHM_POOL_INFILATION_SET_FILE_PATH parameter in the APR1_CONF_SECTION_PARAM table.

Depending on the expected pool inflation, it is possible to configure the fraction of the shared memory to pre-allocate at Event Server start-up using the SHM_POOL_INFILATION_SCRIPT_FACTOR environment variable.

Memory Clean-up

The ADJ1_ES_Cleanup_Shell_Sh script is responsible for cleaning the shared memory of a process. It is used by the Availability Manager to automatically delete the shared memory of a bundled process group (that is, the Event Servers) at the time of failover, when the responsibility is moved to a different server on a different host. The script must be activated when all the processes (active and shadow) that use the particular shared memory are down; otherwise, the memory is not cleaned.

ADJ1_ES_Cleanup_Shell_Sh receives the name of the process as a parameter (for example, ADJ1_ES_Cleanup_Shell_Sh -n ES100). It checks whether the shared memory key already exists, and if so, deletes it as follows:

1. Creates a directory under \$CRASH_LOG_DIR for the relevant Event Server (if the \$CRASH_LOG_DIR variable is not set, the /tmp directory is used)
2. Produces a mini-report of the shared memory state (utilization and last actions) and puts it in the created directory
3. Removes the shared memory
4. Moves all the logs of the relevant Event Server into the created directory
5. Creates a list of all files in the ‘IU’ status in File2E and puts it in the same directory
6. Reports the shared memory removal status upon exit

The ADJ1_ES_Cleanup_Shell_Sh script produces log files with the following naming convention:

arena_report_<ARENA_KEY>_<TIME_STAMP>.log

These log files are located in the \${ABP_LOG} directory.

Clean-up of Oracle Sessions

The ADJ1_KillOracleSession_Sh script is responsible for deleting the Oracle sessions of a process that does not exist anymore. It is called by the Availability Manager to automatically close those stale sessions in the case of a process or host failure. The script requires that an appropriate stored procedure is created in each Usage database instance and appropriate permissions are granted.

The ADJ1_KillOracleSession_Sh script receives the host name and the process ID of the failed process as command-line parameters.

Affected Applications

The process affects the following processes:

- *DB2E* – The Turbo Charging process that extracts data from the Turbo Charging database tables and sends the data as events to Event Server processes.
- *File2E* – The Turbo Charging process that extracts event data from files and then sends the events contained in these files to the relevant Event Servers
- *Rerate* – The Turbo Charging process that handles rerating of events in the postpaid mode

- *Notification* – The Turbo Charging process that extracts notification records from the APE1_NOTIFICATIONS table and sends them to the relevant targets
- *Dispatcher* – The Turbo Charging process that writes external records into files according to their targets or sends real-time notifications to Web services
- *ELA Collector* – The Turbo Charging process that collects and aggregates Event-Level Auditing and Service Level Management data

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*
ADJ1EVENTSRV_<APPLICATION_ID>_%D_%T_%n.log
Example:
ADJ1EVENTSRV_ES100_20081109_130927_1.log
- *Console log files:*
ADJ1_ES_Daemon_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log
Example:
ADJ1_ES_Daemon_inner_Sh_ES100_20081109_130927.log
- *Operational log files:*
ADJ1EVENTSRV_<APPLICATION_ID>_\${ABP_MARKET}_<Date>_<Time>.log
Example:
ADJ1EVENTSRV_ES100_M3G_20081109_130927.log
- *Environment variable log files:*
ADJ1_ES_Daemon_Shell_Sh_<APPLICATION_ID>_<Date>_<Time>.log
Example:
ADJ1_ES_Daemon_Shell_Sh_ES100_20081109_130927.log
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

- *Event log files:*

TC_<EVENT_LOG_SUFFIX>_<APPLICATION_ID>_<PROCESS_ID>_IVER_<IM PL_VER ('0' for the Guiding to Customer (FR) and Network Interface (CR) reporting points)>_<Date>_<Time>_<SEQUENCE_NUMBER>.<FILE_EXT>

Examples:

- TC_EVENT_LOG_CR_ES100_21336_IVER_0_20140225_110436_0.log
- TC_EVENT_LOG_CRIN_ES100_21336_IVER_0_20140225_110436_0.log
- TC_EVENT_LOG_FR_ES120_32092_IVER_0_20140225_090924_0.log
- TC_EVENT_LOG_RB_ES140_9779_IVER_8_20140305_155253_0.log

- *KPI log file:*

KPITrace.log

Location

Log files are located in the following directories:

- *Log files and KPI log file* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG
- *Environment variable log files* – \$ABP_LOG
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in this chapter)
- *Event log files* – As defined by the EVENT_LOG_CRIN.FILE_PATH, EVENT_LOG_FR.FILE_PATH, EVENT_LOG_RB.FILE_PATH, and EVENT_LOG_CR.FILE_PATH configuration parameters (see the “Configuration Parameters” section in this chapter)

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in this chapter.

Contents

The log files contain the following:

- *Log files* – The process error information.
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).
- *Operational log files* – The output of operational scripts.
- *Environment variable log files* – Environment variables used by the process.

- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.
- *Event log files* – Fields with useful statistical information per event. There is a common unique identifier for lines *produced by the same event* in any module on any machine. This enables the user to construct a sequential history of an event as it progressed from module to module.

The following attributes are included at each reporting point in the String format:



Note: Each file includes a header that describes the format of the file (for core fields only).

- Incoming Network Interface (CRIN) reporting point:



Note: This log is printed only for the normal flow.

- Unique ID



Note: For a specific event, this ID is the same across the reporting points.

- Module = CRIN
 - Session ID
 - Event category:
 - ◆ *O* – One-time
 - ◆ *I* – Init
 - ◆ *U* – Update
 - ◆ *T* – Terminate
 - ◆ *N* – None
 - Request number
 - Incoming Event Type ID
 - Protocol ID
 - Resource Type
 - Resource Value
 - Event Date-Time
 - Time of arrival at the Network Interface Module (with millisecond resolution)
 - Duration
 - Internal error code
 - Status
 - Source ID

- Hop By Hop
- End To End

Example:

UniqueID,Module,SessionID,EventCategory,RequestNumber,IncomingEvent
TypeID,ProtocolID,ResourceType,ResourceValue,EventDate,EnterTime,Dur
ation,InternalErrorCode,Status,SourceId,HopByHop,EndToEnd

8614454380000000,CRIN,31 30 32 ,I,1,2,10,1,268735,1370082000,2014-11-
12 13:00:31.107,5,2001,PROC,0,128,128

8621454381000000,CRIN,31 30 32 ,U,2,3,10,1,268735,1370082600,2014-
11-12 13:01:00.292,4,2001,PROC,0,128,128

8615454381000000,CRIN,31 30 32 ,T,3,4,10,1,268735,1370086200,2014-
11-12 13:01:33.537,3,2001,PROC,0,128,128

8620454384000000,CRIN,31 30 37 ,I,1,2,8,1,937165,1370162400,2014-11-
12 13:04:36.230,5,2001,PROC,4,-1,-1

- Guiding to Customer (FR) reporting point:
 - Unique ID
 -  Note: *For a specific event, this ID is the same across the reporting points.*
 - Module = FR
 - Session ID
 - Event category:
 - ◆ *O* – One-time
 - ◆ *I* – Init
 - ◆ *U* – Update
 - ◆ *T* – Terminate
 - ◆ *N* – None
 - Request number
 - Time of arrival at the Guiding to Customer Module (with millisecond resolution)
 - Duration of processing by the Guiding to Customer Module (with millisecond resolution)
 - Status of processing by the Guiding to Customer Module (success or failure)
 - Error code
 - Resource type
 - Resource value
 - Resource segment
 - Customer segment

- Customer ID
- Subscriber ID
- Cycle code
- Source ID

Example:

UniqueId,Module,SessionID,EventCategory,RequestNumber,EnterTime,Dura
tion,Status,ErrorCode,ResourceType,ResourceValue,ResourceSegment,Cus
tomerSegment,CustomerID,SubscriberID,CycleCode,SourceId

2262091818000000,FR,39 32 30 32 33 ,O,3, 2014-05-11
18:18:06.297,1,OK,2001,1,268735,638,643,274,117,1,0

- Event Processing (RB) reporting point:

- Unique ID



*Note: For a specific event, this ID is the same across the reporting
points.*

- Module = RB
 - Token ID
 - Session ID
 - Event category:
 - ◆ *O* – One-time
 - ◆ *I* – Init
 - ◆ *U* – Update
 - ◆ *T* – Terminate
 - ◆ *N* – None
 - Request number
 - Customer ID
 - Subscriber ID
 - Cycle code
 - Time of arrival at the Event Processing Module (with millisecond resolution)
 - Duration of processing by the Event Processing Module (with millisecond
resolution)
 - Frame internal error code
 - Frame external error code
 - Frame event type ID
 - Event source (network element or File2E)
 - Source ID (network element ID or file ID)

- Index in the file (line number)
- Frame status
- Single service index
- Service internal error code
- Service external error code
- Service status
- Service Rating Logic Configurator ID
- Single event index
- Event ID
- Event type ID
- Event internal error code
- Event external error code
- Event status
- IMPL1
- IMPL2
- IMPL3

IMPL1, IMPL2, and IMPL3 are additional fields that can be added to this event log per event type by the implementation. These fields are concatenated at the end of the log, after the core fields.

Example:

```
UniqueId,Module,TokenID,SessionID,EventCategory,RequestNumber,Custo  
merID,SubscriberID,CycleCode,EnterTime,Duration,FrameInternalErrorCode  
,FrameExternalErrorCode,FrameEventTypeId,SourceOfEvent,Sourceld,Inde  
xInFile,FrameStatus,SingleServiceIndex,ServiceInternalErrorCode,ServiceEx  
ternalErrorCode,ServiceStatus,ServiceRLCID,SingleEventIndex,EventID,Eve  
ntTypeID,EventInternalErrorCode,EventExternalErrorCode,EventStatus#IMP  
L1,IMPL2,IMPL3  
2236091737000000,RB,3193236716425052160,00 00 00 00 00 00 00 00 08  
,O,3,274,117,1, 2014-05-11  
16:57:52.583,406,2001,2001,4,NE,-1,-1,PROCESSED,1,2001,2001,SUCCE  
SS,0,1,2258091737000000,40,2001,2001,SUCCESS#102400,0,  
2236091737000001,RB,3193237403619819520,00 00 00 00 00 00 00 02  
,I,1,4,6,1, 2014-05-  
1117:03:13.755,286,2001,2001,2,NE,-1,-1,PROCESSED,1,2001,2001,SUC  
CESS,0,1,2258091737000001,74,2001,2001,SUCCESS#0,0,  
0,RB,3193238228295583391,00 00 00 00 00 00 02 ,I,1,4,6,1, 2014-05-11  
17:06:15.201,36,10086,2001,25,ES,-1,-1,PROCESSED,1,10086,2001,SUCC  
ESS,0,1,2258091737000001,86,10086,2001,IMPL_DROP#0,0,
```

```
2236091737000002,RB,3193240839593656320,00 00 00 00 00 00 00 00 06
,T,1,4,4,1, 2014-05-11
17:25:29.604,80,2001,2001,2,NE,-1,-1,PROCESSED,1,2001,2001,SUCCE
S,0,1,2256091765000000,80,2001,2001,SUCCESS#,,
```

- Outgoing Network Interface (CR) reporting point:



Note: This log is printed only for the normal flow.

- Unique ID



Note: For a specific event, this ID is the same across the reporting points.

- Module = CR
- Session ID
- Event category:
 - ◆ *O* – One-time
 - ◆ *I* – Init
 - ◆ *U* – Update
 - ◆ *T* – Terminate
 - ◆ *N* – None
- Request number
- Customer ID
- Subscriber ID
- Cycle code
- Resource type
- Resource value
- Time of arrival at the Network Interface Module (with millisecond resolution)
- Duration
- Internal error code
- Status
- Source ID

- Hop by hop
- End to end

Example:

UniqueId,Module,SessionID,EventCategory,RequestNumber,CustomerID,SubscriberID,CycleCode,ResourceType,ResourceValue,EnterTime,Duration,InternalErrorCode,Status,SourceId,HopByHop,EndToEnd

2182091672000000,CR,31 32 33 ,I,1,1593,687,1,1,42323423, 2014-05-11 15:52:52.926,575,2001,PROCESSED,0,128,128

2182091672000001,CR,31 32 33 31 ,U,1,1593,687,1,1,42323423, 2014-05-11 15:54:16.718,77,2001,PROCESSED,0,128,128

2182091672000002,CR,31 32 33 31 32 ,U,1,1593,687,1,1,42323423, 2014-05-11 15:56:33.776,83,2001,PROCESSED,0,128,128

2182091672000003,CR,31 32 33 31 32 32 ,T,1,1593,687,1,1,42323423, 2014-05-11 16:00:25.799,58,2001,PROCESSED,0,-1,-1

- *KPI log files* – Performance KPIs per event. See the “[KPI Mechanism](#)” section in this chapter.

Output Files

This section describes the output files of the Event Server.

Event Trace XML File

If defined by the EVENT_TRACING_MODE configuration parameter, the Event Server produces an Event Trace XML file. This file enables testing the calculation performed on an event and its accumulators and investigating related problems. While it is used mainly to check the execution of the implementation written in the Rating Logic Configurator for such calculation, the file also assists in tracing additional problems with event processing, such as the handling of the rating entities and the way the Implementation Compiler translates the instructions from the Implementation Repository of the Rating Logic Configurator.

The Event Trace XML file is produced only by those Event Servers that use Implementation Compiler libraries compiled in the mode that enables printing trace information. These libraries are not used by regular Event Servers in production.

Name

At least one trace file is produced for each Rating Logic Configurator event. The name of the file specifies the event to which it belongs.

If the protocol was implemented using the Diameter framework, session information is added as a prefix to the names of all trace files related to the same frame event. If the event participated in one of the flows for cross-customer shared allowances, the following information is added as a postfix to the name of the relevant file:

- *REDIRECT* – If the event was redirected from the original customer to the shared allowance customer
- *REDIRECT_BACK* – If the event was redirected back to the original customer after having been redirected to the shared allowance customer
- *REDIRECT_BACK_AFTER_PARTIAL_CHARGE* – If the event was redirected back to the original customer after having been redirected to a shared allowance customer and partially charged there

Event Trace files use the following naming convention:

[<source ID>_<session ID>_<request number>_]Event<event ID>_<thread ID>_<year>_<month>_<day>_<hh>_<mm>_<ss>[_<postfix>].xml

Example:

0_1_2_Event0_1258891584_2012_01_01_00_00_00.xml

The Event Server can be configured to produce an additional file with the same information as the regular file, but in a different format. To meet the different usability needs of users who read the file, it can be customized using XSLT. For example, some of the data can be removed, and other data can be reformatted. The name of this file corresponds to the name of the regular file, but is appended with the word “formatted” before the extension:

[<source ID>_<session ID>_<request number>_]Event<event ID>_<thread ID>_<year>_<month>_<day>_<hh>_<mm>_<ss>[_<postfix>]_formatted.xml

Example:

0_1_2_Event0_3172_1258891584_2012_01_01_00_00_formatted.xml

Location

The file is located under \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/.

Contents

Each of the two Event Trace XML files includes the following data:

- Entities
 - Calculated entities, such as events and accumulators. Information is printed before and after the calculation.
 - *Reservation information:*
 - Request type
 - Session ID
 - Session expiration date
 - *Event*
All attributes, including control fields (name, type, and value)
 - *Accumulators*
 -  Note: For accumulators that were changed during rating, the <Updated> tag is set to 'Yes'.
 - *Subscriber information*
 - Subscriber offers
 - Subscriber parameters
 - Subscriber data
 - Qualification information (for Pre-rating and Rating stages), such as offers, packages, and items selected, with additional catalog information on those entities
 - Information on how the selected pricing item types where executed (handler and statement execution descriptions)
 - Dispatching information, including the external record mapping that was executed and notifications created during execution

The Event Trace XML file contains following tags:

- <*EntityDbgBefore*> – Information about the event, accumulators , subscriber, and event variables before the event was processed
- <*EntityDbgAfterPrerate*> – Information about the event and event variables after the Prerating stage
- <*EntityDbgAfter*> – Information about the event, accumulators, event variables, notifications, and dispatching records after the event was processed
- <*Qualification*> – Trace information related to Guiding to Service (can appear twice: before and after event processing)
- <*Evaluation*> – Trace information related to Implementation Compiler, such as activation of qualification cases activation and handlers (can appear twice: before and after event processing)

Protocol Trace XML File

If the protocol was implemented using the Diameter framework, and if defined by the PROTOCOL_TRACING_MODE configuration parameter, the Event Server produces a Protocol Trace XML file for each frame event. This file enables testing the frame event and investigating problems with the parsing of input Diameter messages, creation of Rating Logic Configurator events, and mapping to output Diameter messages. While it is used mainly to check the execution of the implementation written in the Rating Logic Configurator, the file also assists in tracing additional problems with event processing, such as the handling of the rating entities and the way the Implementation Compiler translates the instructions from the Message area in the Implementation Repository of the Rating Logic Configurator.

The Protocol Trace XML file is produced only for those Event Servers that use Implementation Compiler libraries compiled in the mode that enables printing trace information. These libraries are not used by regular Event Servers in production.

Name

At least one trace file is produced for each frame event. The name of the file specifies the session to which it belongs. If the event participated in one of the flows for cross-customer shared allowances, the following information is added as a postfix to the name of the relevant file:

- *REDIRECT* – If the event was redirected from the original customer to the shared allowance customer
- *REDIRECT_BACK* – If the event was redirected back to the original customer after having been redirected to the shared allowance customer
- *REDIRECT_BACK_AFTER_PARTIAL_CHARGE* – If the event was redirected back to the original customer after having been redirected to a shared allowance customer and partially charged there

The Protocol Trace XML files have the following naming convention:

<source ID>_<session ID>_<request number>_Protocol_<year>_<month>_<day>_<hh>_<mm>_<ss>[_<postfix>].xml

Example:

0_6_12_Protocol_2012_09_05_17_34_12.xml

Location

The file is located under \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/.

Contents

The Protocol Trace XML file contains following tags:

- *<MessageIn>* – The input Diameter message
- *<Selection>* – The selection cases that determine which Rating Logic Configurator events to create

- <Parsing> – The mapping of attributes from the frame event to the Rating Logic Configurator events that were created
- <Construction> – The construction of a Diameter response
- <MessageOut> – The output Diameter message

Flow

The Event Server flow depends entirely on configuration. For general information about the flow, see *Turbo Charging Architecture and Amdocs Billing Introduction*.

Environment Variables

The Event Server uses the following environment variables:

- The ADJ1_SHMEM_PATH environment variable determines the path to the files used for generating shared memory keys. By default, it equals ABP_AP_Root.
- The internal CCR_DEMO environment variable determines the path for protocol files (for example, CCR_DEMO = ~/temp). Alternatively, the location of the protocol files to be used by the Event Server can be specified via the PROTOCOL_FILES_LOCATION parameter in the APR1_CONF_SECTION_PARAM table (see the “Configuration Parameters” section in this chapter).

The Event Server uses the values of these parameters when it looks for a specific protocol file, as follows:

- If the CCR_DEMO environment variable exists, the Event Server searches for the relevant file in the database or in the file system, according to the value of the variable.
- If the CCR_DEMO environment variable does not exist, but the PROTOCOL_FILES_LOCATION parameter exists in the APR1_CONF_SECTION_PARAM table, the Event Server searches for the relevant file in the database or in the file system, according to the value of the parameter.
- If neither CCR_DEMO nor PROTOCOL_FILES_LOCATION parameter exists, or if the file is not located in the file system, the Event Server searches for the relevant file in the database. If it cannot find the file, a fatal error occurs.

- The Event Server uses the following environment variables that are initialized by the op_adj_env_sh script:
 - Variables that control the high and low watermarks:

Variable Name	Description	Valid Values/ Default Value
APR_ES_FR_HWM	The high watermark for the Guiding to Customer (also known as FR) task queue, which indicates when the queue is full	16000
APR_ES_FR_LWM	The high watermark for the Guiding to Customer (also known as FR) task queue, which indicates that the queue is ready to receive messages	16000

- Variables that control the number and priority of threads:



Note: Out-of-the-box, the number of threads cannot be set to '0'. To enable setting it to '0' (and thereby disable the corresponding functionality of the Event Server), the Allow Zero Threads attribute of the relevant task in the .ccf file must be changed to 'true'.

Variable Name	Description	Valid Values/ Default Value
APR_ES_CR_IN_NUM_OF_THREADS	The number of incoming Event Interface (also known as CR In) threads in the Event Server.	1
APR_ES_CR_OUT_NUM_OF_THREADS	The number of outgoing Event Interface (also known as CR Out) threads in the Event Server.	1
APR_ES_EAGERLOAD_NUM_OF_THREADS	The number of threads to be used in the Eager Load process.	1
APR_ES_FR_NUM_OF_THREADS	The number of Guiding to Customer (also known as FR) threads in the Event Server.	1
APR_ES_PW_MAX_TOTAL_CHILD_MESSAGES	The maximum number of pending messages for a specific customer group, only if the group is suspended. When this limit is reached, the application may drop events.	50000
APR_ES_PW_MAX_TOTAL_MESSAGES	The maximum number of pending messages on the queue. When this limit is reached, the application may drop messages.	50000
APR_ES_PW_MAX_TOTAL_PARENT_MESSAGES	The maximum number of pending messages on a database connection. When this limit is reached, the application may drop events.	50000
APR_ES_PW_NUM_OF_THREADS	The number of Persistence Writer threads in the Event Server.	1
APR_ES_PW_OVERRIDE_PRIORITY	Indicates whether the Persistence Writer threads have a non-default priority.	false

Variable Name	Description	Valid Values/ Default Value
APR_ES_PW_PRIORITY	The priority of the Persistence Writer threads if the APR_ES_PW_OVERRIDE_PRIORITY variable is set to ‘true’.	Default
APR_ES_RATER_ENABLE_BLOCKING_THREADS	Indicates whether the blocking threads functionality is enabled. Blocking threads are blocked while waiting for an I/O response; they are only at work when the database must be accessed. Non-blocking threads perform all other work. After a host failover, blocking threads work against the database, thereby improving the latency of events.	true
APR_ES_RATER_NUM_OF_THREADS	<p>The number of Rating threads in the Event Server.</p> <p>If the APR_ES_RATER_ENABLE_BLOCKING_THREADS environment variable is set to ‘true’, the threads defined for the Rater task are divided into two groups: blocking and non-blocking.</p> <p>The number of non-blocking threads is the same as the number of logical CPUs on the machine (calculated automatically). The other threads are blocking. If the defined number of Rating threads is less than the number of logical CPUs, only one thread is defined as blocking and the rest are non-blocking.</p> <p>Therefore, it is important to configure enough Rating threads for the blocking functionality. For example, the recommended value for 12 core CPUs is 78 threads.</p>	1
APR_ES_RATER_OVERRIDE_PRIORITY	Indicates whether the Rating threads have a non-default priority.	false
APR_ES_RATER_PRIORITY	The priority of the Rating threads if the APR_ES_RATER_OVERRIDE_PRIORITY variable is set to ‘true’.	Default
APR_ES_RERATE_NUM_OF_THREADS	The number of Rerate threads in the Event Server.	1

- Variables that define the number of elements in queues:

Variable Name	Description	Valid Values/ Default Value
APR_ES_FR_QUEUE_ELEMENTS	The number of elements in the Guiding to Customer (FR) queue	-1
APR_ES_PW_QUEUE_ELEMENTS	The number of elements in the Persistence Writer queue	-1
APR_ES_RATER_QUEUE_ELEMENTS	The number of elements in the Event Processing (Rater) queue	-1

- Variables that are related to connections:

Variable Name	Description	Valid Values/ Default Value
APR_CONN_NETWORK_RECV_MAX_WAIT_SEC	The maximum number of seconds to try to read a buffer from a stream, as long as at least one byte is received per second.	60
APR_CONN_NETWORK_SEND_MAX_WAIT_SEC	The maximum number of seconds to try to send a buffer to a stream, as long as at least one byte is sent per second.	60
NUM_OF_CUG_RECONNECT_ATTEMPTS	The number of attempts to reconnect to the resource database.	3
NUMBER_OF_CONNECTIONS_PER_CYCLE	The number of connections to the Usage database per cycle.	1
RECONNECTATT	The maximum number of reconnection attempts to the current database.	3
SLEEP_BTWN_CUG_RECONNECT_ATTEMPTS	The sleep time between attempts to reconnect to the resource database.	3
SLEEPBETCONN	The sleep time (in seconds) between two attempts to reconnect to the database. If n > 0, the value of the SLEEP_TIME_BETWEEN_RECONNECTION_ATTEMPTS is taken from this environment variable (currently, 30 seconds).	A positive number representing seconds
SLEEPBETSERVCONN	The sleep time between reconnection attempts when trying to connect to another server, in nanoseconds.	30000000000

- Variables related to the Rater Init process:

Variable Name	Description	Valid Values/ Default Value
CPF_TC_MAIN_SLEEP_INTERVAL	The sleep time for the Rater Init	0
IGNORE_ERRORS_IN_INIT	The number of errors in the initialization process to be displayed in the log	1

- Variables that define various buffer sizes:

Variable Name	Description	Valid Values/ Default Value
APR_ACCUM_INSERT_BUFFER_SIZE	For the Persistence Writer Function. The size of the OCI Oracle buffer for keeping accumulators.	10000
APR_CONN_EXTERNAL_RECV_BUF_SIZE_CONNECTION	The size of the TCP buffer that is used for receiving messages through an external connection (to an external network element).	25165824
APR_CONN_EXTERNAL_SEND_BUF_SIZE_CONNECTION	The size of the TCP buffer that is used for sending messages through an external connection (to an external network element).	25165824

Variable Name	Description	Valid Values/ Default Value
APR_CONN_INTERNAL_CLIENT_RECV_BUF_SIZE	The size of the buffer that is used for receiving messages through a connection to an Event Server client, such as File2E.	25165824
APR_CONN_INTERNAL_CLIENT_RECV_BUF_SIZE_CONNECTION	The size of the TCP buffer that is used for receiving messages through a connection to an Event Server client, such as File2E.	25165824
APR_CONN_INTERNAL_CLIENT_SEND_BUF_SIZE	The size of the buffer that is used for sending messages through a connection to an Event Server client, such as File2E.	25165824
APR_CONN_INTERNAL_CLIENT_SEND_BUF_SIZE_CONNECTION	The size of the TCP buffer that is used for sending messages through a connection to an Event Server client, such as File2E.	25165824
APR_CONN_INTERNAL_RECV_BUF_SIZE_CONNECTION	The size of the TCP buffer that is used for receiving messages through an internal connection (between Event Servers).	25165824
APR_CONN_INTERNAL_SEND_BUF_SIZE_CONNECTION	The size of the TCP buffer that is used for sending messages through an internal connection (between Event Servers).	25165824
UDP_RECV_BUF_SIZE	The size of the UDP buffer that is used for receiving messages. This size is calculated as follows: UDP_SEND_BUF_SIZE * (One fourth of the number of Event Servers in one cluster)	6656000
UDP_SEND_BUF_SIZE	The size of the UDP buffer that is used for sending messages. This size is calculated as follows: (Number of Event Servers performing the Network Interface Function) * (Number of network element IDs) * 5 * (Size of an aggregation record = 64) * (Number of reporting points in one process). For example, in an independent Event Server functioning as the Event Processing Module, there are two reporting points: RB and PW) * 2 (For busy response handling)	1024000

- Variables that define whether the SUN performance library is used:

Variable Name	Description	Valid Values/ Default Value
ADJ_SUN_LIBUMEM_PATH	The path to the libumem.so library	/usr/lib/64/libumem.so
ADJ_USE_SUN_LIBUMEM	Indicates whether the libumem.so library must be used	true

- Variable that defines whether the general Duplicate Check mode is enabled:

Variable Name	Description	Valid Values/ Default Value
GENERAL_DUPCHECK_MODE_ENABLED	<p>Indicates whether the general Duplicate Check mechanism is enabled. Values are:</p> <ul style="list-style-type: none">■ <i>Y</i> – A duplicate check is performed if it is required by the Message-Based Flow or if it is enforced by the client (for example, when File2E sends Rerun files). In IOT mode, the event is inserted into the index-organized table (IOT) for duplicate checks.■ <i>N</i> – A duplicate check is not performed event if it is required by the Message-Based Flow. However, a duplicate check is still performed if it was forced by the client (for example, when File2E sends Rerun files). If a duplicate check is required by the Message-Based Flow or if it was forced by the client, the event is inserted into the IOT for duplicate checks.	Y

- Variables for Internal Watchdog configuration that define the following:
 - *Queue or Weight* – The Availability Manager calculates the weight of a blocked queue. If the cumulative weight of all the queues is greater than 100, the Availability Manager terminates the Event Server. If the value for a particular queue is set to 100 (default), the Availability Manager terminates the Event Server as soon as this queue is blocked.
 - *Queue Hang-up Threshold* – The time interval at which a queue must be checked for blockage. The Availability Manager queries the queue every 10 seconds. If this threshold is set to 30 seconds, the queue reports blockage only after 30 seconds have passed, and the Availability Manager has queried the queue three times. In other words, this threshold defines the time period from the last check for which a queue can be stuck before it is considered as blocked.
 - *Task Weight* – If (*<Number of blocked threads> / <Total number of threads>*) * Weight >= 100, the Availability Manager is notified. For example, if a task has four threads, two of which are blocked, the Availability Manager is notified only if the weight is configured as 200 or if all the threads become blocked. The default value of 100 means that only when all the threads are blocked, the Availability Manager is notified.
 - *Task Hang-up Threshold* – The time period for which the task threads can run without sending a heartbeat to the Internal Watchdog. If the task threads do not send a heartbeat to the Internal Watchdog within this time period, the task is considered blocked.

Task Name in CCF File	Queue Weight		Queue Hang-up Threshold		Thread Weight		Thread Hang-up Threshold	
	Variable Name	Default Value	Variable Name	Default Value	Variable Name	Default Value	Variable Name	Default Value
CR Outgoing	APR_ES_CR_OUT_IWD_QUEUE_WEIGHT	100	APR_ES_CR_OUT_IWD_QUEUE_HANGUP	60	APR_ES_CR_OUT_IWD_TASK_WEIGHT	100	APR_ES_CR_OUT_IWD_TASK_HANGUP	60
CR Incoming	APR_ES_CR_IN_IWD_QUEUE_WEIGHT	100	APR_ES_CR_IN_IWD_QUEUE_HANGUP	60	APR_ES_CR_IN_IWD_TASK_WEIGHT	100	APR_ES_CR_IN_IWD_TASK_HANGUP	60
Rater	APR_ES_RATER_IWD_QUEUE_WEIGHT	100	APR_ES_RATER_IWD_QUEUE_HANGUP	60	APR_ES_RATER_IWD_TASK_WEIGHT	100	APR_ES_RATER_IWD_TASK_HANGUP	60

Task Name in CCF File	Queue Weight		Queue Hang-up Threshold		Thread Weight		Thread Hang-up Threshold	
	Variable Name	Default Value	Variable Name	Default Value	Variable Name	Default Value	Variable Name	Default Value
GAT	APR_ES_GAT_IWD_QUEUE_WEIGHT	100	APR_ES_GAT_IWD_QUEUE_HANGUP	60	APR_ES_GAT_IWD_TASK_WEIGHT	100	APR_ES_GAT_IWD_TASK_HANGUP	60
Audit Aggregator	APR_ES_AUDIT_AGGREG_IWD_QUEUE_WEIGHT	100	APR_ES_AUDIT_AGGREG_IWD_QUEUE_HANGUP	60	APR_ES_AUDIT_AGGREG_IWD_TASK_WEIGHT	100	APR_ES_AUDIT_AGGREG_IWD_TASK_HANGUP	60
DispatcherSM	APR_ES_DISP_SM_IWD_QUEUE_WEIGHT	100	APR_ES_DISP_SM_IWD_QUEUE_HANGUP	60	APR_ES_DISP_SM_IWD_TASK_WEIGHT	100	APR_ES_DISP_SM_IWD_TASK_HANGUP	60
Cleanup	APR_ES_CLNUP_IWD_QUEUE_WEIGHT	100	APR_ES_CLNUP_IWD_QUEUE_HANGUP	3660	APR_ES_CLNUP_IWD_TASK_WEIGHT	100	APR_ES_CLNUP_IWD_TASK_HANGUP	3660
FR	APR_ES_FR_IWD_QUEUE_WEIGHT	100	APR_ES_FR_IWD_QUEUE_HANGUP	60	APR_ES_FR_IWD_TASK_WEIGHT	100	APR_ES_FR_IWD_TASK_HANGUP	60
ES_PWBS	APR_ES_PW_IWD_QUEUE_WEIGHT	0	APR_ES_PW_IWD_QUEUE_HANGUP	60	APR_ES_PW_IWD_TASK_WEIGHT	0	APR_ES_PW_IWD_TASK_HANGUP	60
Events Producer	APR_ES_EVENT_PROD_IWD_QUEUE_WEIGHT	0	APR_ES_EVENT_PROD_IWD_QUEUE_HANGUP	60	APR_ES_EVENT_PROD_IWD_TASK_WEIGHT	0	APR_ES_EVENT_PROD_IWD_TASK_HANGUP	60
PM Gateway	APR_ES_PMG_IWD_QUEUE_WEIGHT	100	APR_ES_PMG_IWD_QUEUE_HANGUP	60	APR_ES_PMG_IWD_TASK_WEIGHT	100	APR_ES_PMG_IWD_TASK_HANGUP	60
Audit Client Gateway	APR_ES_AUDIT_CLIENT_GATEWAY_IWD_QUEUE_WEIGHT	100	APR_ES_AUDIT_CLIENT_GATEWAY_IWD_QUEUE_HANGUP	60	APR_ES_AUDIT_CLIENT_GATEWAY_IWD_TASK_WEIGHT	100	APR_ES_AUDIT_CLIENT_GATEWAY_IWD_TASK_HANGUP	60

- Environment variables for controlling the recovery of event-level auditing data:



Note: By default, these environment variables are not declared or used, and they override the values in the code only if necessary.

Variable Name	Description	Valid Values/ Default Value
APR_ES_ELA_CATCH_UP_TASK_TIME_OUT	The sleep time (in seconds) of the ELACatchUp task between handling bulks of event-level auditing messages. This task reads a bulk of messages from the database, and if the message queue is below the low watermark, sends these messages to the Aggregator task.	1000
APR_ES_ELA_QUEUE_HIGH_WATER_MARK	The percentage of the size of the ELA message queue that defines when the ELA Recovery Catch-up flow is terminated. If the size of the message queue exceeds this value, the Catch-up flow stops.	85
APR_ES_ELA_QUEUE_LOW_WATER_MARK	The percentage of the size of the ELA message queue that defines when the ELA Recovery Catch-up flow is activated. If the message queue was full previously and then its size decreased below this value, the Catch-up flow starts.	70
APR_ES_ELA_RECOVERY_CLEANUP_TIME_MIN	The intervals (in minutes) at which the ELA Recovery Clean-up flow must run. The Clean-up flow checks whether there are records in the APR1_ES_ELA_RECOVERY that were sent to the ELA Collector but did not receive a response. These records are marked as out of queue, and when the Catch-up flow is activated, they are resent.	30
APR_ES_ELA_RECOVERY_LEVEL	The recovery level for event-level auditing data in the Event Server. Values are: <ul style="list-style-type: none"> ▪ DISABLE ▪ FULL_PERSISTENCE – Every record is written to the database (a high recovery level). ▪ OVERFLOW_PERSISTENCE – Records are written to the database only when the message queue for event-level auditing data is full. 	FULL_PERSISTENCE
APR_ES_ELA_RECOVERY_PARTITION_S_TO_KEEP	The number of partitions in the APR1_ES_ELA_RECOVERY table to keep when partitions are truncated.	2
APR_ES_ELA_RECOVERY_TRUNCATE_TIME_MIN	The intervals (in minutes) at which the ELA Recovery Truncate flow must run.	60

- Environment variables related to logging and tracing:

Variable Name	Description	Valid Values/ Default Value
ADJ_LIGHT_TRACER_ENABLE	The Boolean property that indicates whether to turn on the log in trace functionality and merge the log and trace messages into a single binary file. This property is valid only when the GNRL_LIGHT_TRACE parameter is defined as one of the Logger outputs.	Y
ADJ_LOGGER_ENABLE	Enables the creation of log messages via the Logger mechanism.	true
ADJ_LOGGER_LIGHT_TRACER_ENABLE	Enables the creation of Light Tracer logs.	true
ADJ_LOGGER_LIGHT_TRACER_SEVERITY	The message severity threshold for the log. The severity range is 0–50000 according to the standard Logger definitions: <ul style="list-style-type: none"> ■ 0 – Trace ■ 10000 – Debug ■ 20000 – Info ■ 30000 – Warning ■ 40000 – Error ■ 50000 – Fatal 	0
ADJ_LOGGER_MEMDB_arena_ENABLE	Enables the arena log output, which is usually printed to the application log.	TRUE
ADJ_LOGGER_MEMDB_arena_SEVERITY	The message severity threshold for the log. The severity range is 0–50000 according to the standard Logger definitions: <ul style="list-style-type: none"> ■ 0 – Trace ■ 10000 – Debug ■ 20000 – Info ■ 30000 – Warning ■ 40000 – Error ■ 50000 – Fatal 	40000
ADJ_LOGGER_SEVERITY	The message severity threshold for the log. The severity range is 0–50000 according to the standard Logger definitions: <ul style="list-style-type: none"> ■ 0 – Trace ■ 10000 – Debug ■ 20000 – Info ■ 30000 – Warning ■ 40000 – Error ■ 50000 – Fatal 	0
ADJ_TRACER_ENABLE	Enables the creation of trace messages via the Logger mechanism.	true

Variable Name	Description	Valid Values/ Default Value
ADJ_TRACER_SEVERITY	The message severity threshold for the log. The severity range is 0–50000 according to the standard Logger definitions: <ul style="list-style-type: none">■ 0 – Trace■ 10000 – Debug■ 20000 – Info■ 30000 – Warning■ 40000 – Error■ 50000 – Fatal	0
APE_EVENT_TRACING_MODE	Enables event tracing.	TRUE
APR_ENABLE_CR_EVENT_LOG	Indicates whether the event log at the Outgoing Network Interface reporting point is enabled at Event Server start-up. It can then be stopped and re-activated using an admin command.	Y
APR_ENABLE_CRIN_EVENT_LOG	Indicates whether the event log at the Incoming Network Interface reporting point is enabled at Event Server start-up. It can then be stopped and re-activated using an admin command.	Y
APR_ENABLE_FR_EVENT_LOG	Indicates whether the event log at the Guiding to Customer reporting point is enabled at Event Server start-up. It can then be stopped and re-activated using an admin command.	Y
APR_ENABLE_RB_EVENT_LOG	Indicates whether the event log at the Event Processing reporting point is enabled at Event Server start-up. It can then be stopped and re-activated using an admin command.	Y
APR_EVENT_LOG_FLUSH_SIZE	The size (in bytes) up to which an event log is accumulated before it is written to a file. Values are: <ul style="list-style-type: none">■ 0 – No buffering. Records are flushed to disk immediately.■ -1 – Default buffering of the operating system.■ >0 – The actual buffer size.	8192
APR_EVENT_LOG_MAX_FILE_SIZE	The maximum size (in megabytes) of an event log file produced at the Outgoing Network Interface reporting point.	25
APR_EVENT_TRACING_MODE	Enables event tracing.	Y

Variable Name	Description	Valid Values/ Default Value
IS_DURATION_REQUIRED	<p>Specifies whether SQL tracing must be activated for a limited amount of time only (for example, in production). If this is the case:</p> <ul style="list-style-type: none"> ■ If SQL tracing is activated at process start-up, it is limited to 60 minutes. ■ The command that activates the tracing on the fly must specify the required duration up to the allowed maximum (60 minutes). 	Y
KPI_TRACE_FREQUENCY	The interval, defined as the number of events, at which KPI data is collected and written to a log file. The KPIs are reported per event.	N/A
KPI_TRACE_LINES_PER_FILE	The maximum number of lines in a KPI trace file. When this number is reached, a new file is opened.	3000
PRINT_REPLICATED_SESSIONS	If defined, enables printing replicated sessions to the log in Debug mode.	N/A

- Other environment variables:

Variable Name	Description	Valid Values/ Default Value
ADJ1_SESSION_BASED_GUIDING_HASH_SIZE	The number of entries in the hash table that contains data for session-based guiding in AIMOS. This number must be set according to the expected number of Sy sessions.	1000
AGD_TOTAL_NUMBER_OF_SUBSCRIBERS	The total number of subscribers to be handled by all Guiding to Customer servers (used to calculate hash sizes for the Guiding segment hash table).	10000
APR_ELA_ENABLE	Enables or disables the Event-Level Auditing mechanism.	Y
APR_ES_SBG_PERSIST_TIMEOUT	The period (in milliseconds) in which the persistence task must flush its buffers into the database. The value must be greater than 1.	1000
APR_EXTERNAL_EVENT_SLA_MS	Service Level Agreement (SLA) duration in milliseconds. If the time that an event spent in the Rater queue exceeds this SLA, the event is dropped in the Event Processing Module. A null (0) value disables this feature. Disabling it affects the host failover KPI.	0
APR_FLUSH_THRESHOLD	After this number of records has been processed, the events are inserted into the Rated Events table. The value of this parameter cannot exceed 64 KB.	1
APR_MEMORY_SIZE_IN_BYTES	The requested size of the shared memory.	1073741824
APR_OCFE_PROTOCOL_ID	The ID of the Amdocs Service Platform (OCFE) protocol.	16

Variable Name	Description	Valid Values/ Default Value
APR_PW_LF_NO_COMMIT_TIMEOUT_DELTA	The timeout from the last commit, in milliseconds, after which the queue sends an expiration message to the Persistence Writer.	1000
APR_SHM_POOL_INFILATION_SET_FILE_PATH	The path to the file with the shared memory inflation pool sizes, which is created by the ADJ1_AllocatedSharedMemoryBlocks_Sh script during host failover.	\${ABP_AP_R_OOT}/config
APR_SR_DEOLDREPSESS_CHECKCUSTOMER_TIMEPERIOD	The frequency with which a subscriber's time stamp (updated every time a subscriber is accessed) under a customer is to be checked. A replication session can only be deleted when the customer is locked. To prevent unnecessary locks, the scan period is checked first. When the difference between the current time and the latest time stamp of the subscriber is greater than this parameter, the customer is locked and scanned for old replication sessions, which are subject to removal.	86400
APR_SR_DEOLDREPSESS_HOLDSESSIONS_TIMEPERIOD	The period of time for which the replication session is kept in memory (in milliseconds). When the difference between the current time and the creation time of the replication session is greater than or equal to the value of this parameter, the replication session is deleted.	7464960000
APR_SR_DEOLDREPSESS_SCANCUSTOMERS_TIMEPERIOD	The frequency with which customers are to be scanned. Customers are scanned when the difference between the current time and the time stamp of the last scan is greater than this parameter, and not after each timeout of the maintenance thread.	3600
APR_STALE_SESSIONS_MAX_SESSIONS_PER_SUBSCRIBER	The maximum number of the latest updated Sy-like sessions per subscriber to be retained in the memory of the Event Server when stale sessions are removed.	3
APR_STALE_SESSIONS_IS_EVENTETIME_GMT	Indicates whether the time stamp of Sy-like events is in GMT. If this parameter is set to 'N', the logic of Sy-like session removal does not perform conversion to local time.	N
APR_STALE_SESSIONS_PROTOCOL_IDS	A comma-separated list of Sy-like protocol IDs for which the removal of stale sessions from memory is enabled.	N/A

Variable Name	Description	Valid Values/ Default Value
CORE_DUMP_ENABLED	Indicates whether the core dump file must be created if the Event Server fails. By default, the creation of the core dump file is bypassed, and only a stack trace of the failure is written to the console log. To enable the regular flow with the core dump file, this parameter must be set to 'Y'.	N
DIAMETER_AUTH_APPLICATION_ID_IN_RESPONSE_ENABLED	Indicates whether the Auth-Application-Id attribute-value pair (AVP) is added to response messages. Possible values: <ul style="list-style-type: none"> ■ <i>Y</i> – The AVP is added to responses in all new protocols. ■ <i>N</i> – The AVP is not added to responses in any of the new protocols (not including the CEA message). ■ <i>Protocol IDs separated by commas</i> – The AVP is added to responses only in the specified protocols (the setting does not affect old protocols and the CEA message). 	Y
ES_MEMORY_SIZE_IN_BYTES	The requested size of the shared memory.	1073741824
ES_TCCLIENTS_HANDSHAKE_RCV_TIMEOUT	The timeout for receiving the message in the handshake between the Event Server and Turbo Charging clients.	30
ES_TCCLIENTS_HANDSHAKE SND_TIMEOUT	The timeout for sending the message in the handshake between the Event Server and Turbo Charging clients.	30
FR_BLOCKING_THREAD_ENABLED	Indicates whether guiding to customer in the Network thread (a blocking thread) is enabled. The value of this variable must be set in accordance with NFT results.	Y
GR_ENABLED	Indicates whether Geo Redundancy support is enabled.	N
OM_CREATE_BACKUP	Indicates whether the current file containing operational measurements must be copied to a back-up file before the current file is cleared. The back-up file can be used for investigating the causes of active-shadow or process group failover. This environment variable is relevant for all the processes that use the operational measurement framework.	N

Variable Name	Description	Valid Values/ Default Value
ROLLBACK_FULL_TRANSACTION	<p>Determines the behavior of the Persistent Writer if it fails to insert records into the database:</p> <ul style="list-style-type: none"> ■ <i>Y</i> – The entire transaction (event, accumulators, notification records, and dispatching records) is rolled back from the database, and all the records of the transaction (including the ones that did not fail) are inserted into the APE1_ERR_ENTITIES_DETAILS table. ■ <i>N</i> – Only the problematic record of a transaction is rolled back from the database and inserted into the APE1_ERR_ENTITIES_DETAILS table. The Persistence Writer attempts to insert the remaining records into the database. If the remaining records cannot be inserted into the database, the event transaction is treated as if ROLLBACK_FULL_TRANSACTION had the Y value. 	N
SEND_TO_REJECTED_RB	When all the Event Servers functioning as Guiding to Customer Modules are down, determines whether the event must be sent to the Event Processing Module that handles rejected events ('Y') or dropped with response code 3004 ('N').	Y
SHM_POOL_INFILATION_LOW_WATER_MARK_THRESHOLD	The smallest shared memory pool inflation size that can be taken from a file. If the calculated size from the file is lower than this threshold, the value of the SHM_POOL_INFILATION_SET parameter in the database is used instead.	2147483648
SHM_POOL_INFILATION_SCRIPT_FACTOR	The fraction of the shared memory to pre-allocate at Event Server start-up (for example, a factor of 2 means 50% of the shared memory).	2
USE_NEW_IOT_TABLE	Indicates whether events must be written both to the APE1_EVENT_DUP_KEYS and to the APE1_EVENT_DUP_KEYS_EXT tables for seamless upgrade without any loss of data.	N

For more information about the CCF file, see the “Configuration Files” section in this chapter.

For more information on the op_adj_env_sh script, see the “Amdocs Batch Job Manager Application Script” section in the “Introduction” chapter.

- The Event Server uses the following environment variable that is initialized in the code. For it to be used, it must be added to the op_pm_en_sh script.

Variable Name	Description	Valid Values/ Default Value
APE1_DISABLE_CONVERT	Disables the Data Upgrader of the Implementation Compiler	Y

- The Event Server has the following environment variables for monitoring queues. By default, these environment variables are not declared or used, and they override the values in the code only if necessary.

Variable Name	Description	Valid Values/ Default Value
OM_LFQ_MAX_MSG_ID	The maximum message ID used in the Queue Monitor	2048
OM_LFQ_MAX_Q_ID	The maximum queue ID used in the Queue Monitor	40
OM_LFQ_POOL_SIZE	The initial size of the pool for LFQ counter records	100

Parameters

The following parameters must be set for the process to run properly.

Parameter	Description	Default Value
APPLICATION_ID	The process instance	N/A
CCF_FILE_NAME	An XML file that contains the relevant configuration	complete_ES_TCP_20.ccf

Configuration Parameters

The configuration parameters (properties) of the Event Server are defined in the database.

The following table shows the performance tuning (configuration) parameters of the Event Server. These parameters are defined in the ADJ1_CONF_SECTION_PARAM, APE1_CONF_SECTION_PARAM, APR1_CONF_SECTION_PARAM, and AGD1_CONF_SECTION_PARAM tables using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*).

The Event Server works even when these parameters are not defined in the configuration tables. In this case, it uses default values. To change a value, add the parameter to the relevant configuration table.

In addition, the Event Server uses the parameters of the Logger and Tracer mechanisms of C++ Foundation, and of Audit & Control. For more information, see *C++ Foundation Programmer Guide* and *Audit & Control Run Book*.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ADJ	ADJ.MEMDB	ACCUMULATOR_LAYOUT_GROW_RATIO	3.5	Defines the ratio between the number of accumulators in the hash table and the size of the hash. The hash has to grow when (the size of the hash * ACCUMULATOR_LAYOUT_GROW_RATIO) is less than the number of accumulators.
ADJ	ADJ.MEMDB	ACCUMULATORS_INIT_HASH_SIZE	32 (if the parameter is not defined)	Defines the initial size of the hash table to allocate for accumulators. The size must be a power of 2 (if it is not, it is replaced with the nearest power of 2; for example, 31 is replaced with 32).
ADJ	ADJ.MEMDB	CUSTOMER_REP_SESSION_HASH_GROW_RATIO	3.5	Defines the ratio between the number of customers in the replication hash table and the size of the hash. The hash has to grow when (the size of the hash * CUSTOMER_REP_SESSION_HASH_GROW_RATIO) is less than the number of customers.
ADJ	ADJ.MEMDB	CUSTOMER_REP_SESSION_INIT_HASH_SIZE	32 (if the parameter is not defined)	Defines the initial size of the replication hash table to allocate for customers. The size must be a power of 2 (if it is not, it is replaced with the nearest power of 2; for example, 31 is replaced with 32).

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ADJ	ADJ.MEMDB	SIC_GROUP_SESSIONS_HASH_SIZE	32	The number of buckets in the hash of sessions. This parameter is not included out of the box. If it is not added, the default value is used.
ADJ	ADJ.MEMDB	SUBSCRIBERS_HASH_GROW_RATIO	3.5	Defines the ratio between the number of subscribers in the hash table and the size of the hash. The hash has to grow when (the size of the hash * SUBSCRIBERS_HASH_GROW_RATIO) is less than the number of subscribers.
ADJ	config	LOCKING_BLOCK_SIZE_IN_BYTES	32 MB	The size of an arena locking block in bytes. The arena is logically divided into blocks of this size. When memory locking is enabled (the value of the MEMORY_LOCKING_ENABLED parameter is set to ‘TRUE’), every time that the arena outgrows the current block, the Event Server checks how much free RAM is still available to determine whether it can lock the memory it requires. For more information about the checks, see the MIN_AVAILABLE_RAM_IN_MB parameter.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ADJ	config	MAX_NUM_OF_SUBSCRIBERS_FOR_BULK_PI_LOAD	2	<p>Controls the bulk load of accumulators for customers depending on their number of subscribers. The meaning of the values for this parameter depends on the role of the Event Server (master or replication):</p> <ul style="list-style-type: none"> ■ Master Event Server <ul style="list-style-type: none"> • <i>0</i> – Bulk load of accumulators is disabled. • <i>Any other value</i> – Accumulators are loaded in a bulk for all customers. ■ Replication Event Server <ul style="list-style-type: none"> • <i>0</i> – Bulk load of accumulators is disabled. • <i>Any other value</i> – The maximum number of subscribers and agreements that a customer may have to be eligible for a bulk load. In other words, if a customer has more than this number of subscribers and agreements, its accumulators are not loaded in a bulk.
ADJ	config	MEMORY_LOCKING_ENABLED	FALSE	<p>Indicates whether memory locking is enabled. In the case of a second host failure, when it is possible that the replication host does not have enough physical memory to take over the processing of two failed master hosts, the locking mechanism prevents swapping the memory to disk.</p> <p>If memory locking is enabled, the Event Server checks the free physical memory. For more information about the frequency of the checks, see the LOCKING_BLOCK_SIZE_IN_BYTES parameter. For more information about the checks that are performed and the conditions for locking, see the MIN_AVAILABLE_RAM_IN_MB parameter.</p>

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ADJ	config	MIN_AVAILABLE_RAM_IN_MB	1 GB	<p>The minimum size of the available RAM for locking (if the value of the MEMORY_LOCKING_ENABLED parameter is set to ‘TRUE’). Depending on this size, the Event Server performs the following:</p> <ul style="list-style-type: none"> ▪ If more than MIN_AVAILABLE_RAM_IN_MB is available, the Event Server checks whether the size of the memory that is already locked (in all the processes) plus the size of the locking block is less than the value of the RLIMIT_MEMLOCK Unix parameter. If so, it locks the entire block in RAM. If there are some unlocked arena pieces (due to previous lack of free RAM), all of them are locked. ▪ If less than MIN_AVAILABLE_RAM_IN_MB is available, the Event Server does not lock any memory and sends a message to the Availability Manager. The Availability Manager acts upon the message only if the Event Server is defined as a replication and has now assumed master responsibilities. An appropriate message is displayed in the application log at the INFO level.
ADJ	LIGHT_TRACER	BUFFER_SIZE	512,000*64bit	The size of the Light Tracer buffer. The Light Tracer buffer is a cyclic one-time allocated buffer that holds input messages before they are written to the output file. The buffer size must be large enough so that no messages are lost.
ADJ	LIGHT_TRACER	CONTEXT_ID	N	The Boolean property indicating whether to include the context ID in the trace messages for every module and sub-module at start-up.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ADJ	LIGHT_TRACER	EVENT_LIGHT_TRACER_CONFIG_FLOWS	N/A	<p>The event flows for which the Flow Tracer is enabled. Light Tracer messages related to events in these flows are grouped and written to .toe files if these events are sent from the network elements defined in the EVENT_LIGHT_TRACER_CONFIG_SOURCES parameter.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> ■ 1 – Main Event Server flow (Network Interface Function > Guiding to Customer Module > Event Processing Module > Replication > Network Interface Function) ■ 2 – Event Processing Module > Persistence Writer <p>The values of this parameter must be separated by commas.</p>
ADJ	LIGHT_TRACER	EVENT_LIGHT_TRACER_CONFIG_SOURCES	N/A	<p>The IDs of the network elements for which the Flow Tracer is enabled. Light Tracer messages related to events sent by these network elements are grouped and written to .toe files if these events are in a flow defined in the EVENT_LIGHT_TRACER_CONFIG_FLOWS parameter. The IDs must be separated by commas.</p> <p>The .toe files use the same naming convention as the regular trace files. Their path is defined using the LIGHT_TRACER.FLOW_TRACER.FILE_PATH parameter.</p>
ADJ	LIGHT_TRACER	FILE_PATH	\${ABP_APP_LOG}/trace/	The full path to the location where the trace file is to be written.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ADJ	LIGHT_TRACER	FLUSH_SIZE	-1	<p>The data is buffered before being flushed to the output file. The size of this buffer may influence the performance. On the other hand, crucial trace messages may be lost if the process crashes because of buffering. This parameter defines whether trace messages are buffered or flushed immediately. The following values are available:</p> <ul style="list-style-type: none"> ■ <i>-1 (default)</i> – The buffer size is the default of the operating system. ■ <i>0</i> – Each trace message is flushed immediately. ■ <i>> 0</i> – A custom buffer size is specified.
ADJ	LIGHT_TRACER	MAX_READ_ATTEMPTS	20	An internal parameter of the Light Tracer, which specifies the number of attempts to read the trace data before the reader decides that the writer is stuck and synchronizes with the current writer.
ADJ	LIGHT_TRACER	MAX_TRACE_FILE_SIZE	256	A Light Tracer parameter that defines the maximum size (in MB) of a binary trace file. When a file reaches this size, a new file is created.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ADJ	LIGHT_TRACER	SHUT_DOWN_CHECK_FREQ	1000	<p>An internal parameter of the Light Tracer, which defines the frequency at which the Light Tracer checks whether one of the limits for its shutdown has been reached. The limit can be defined as the maximum number of messages that the buffer can contain (set by the <code>--noOfMessages</code> parameter of the <code>SET_LIGHT_TRACER_COMMAND</code> admin command) or the maximum amount of time during which the tracing is activated (set by the <code>--period</code> parameter of the <code>SET_LIGHT_TRACER_COMMAND</code> admin command or by the <code>--duration</code> parameter of the <code>SET_TRACE_SQL_COMMAND</code> admin command).</p> <p>This frequency is measured in the number of messages (1000 by default). Namely, every time the Light Tracer encounters the set amount of messages, it checks whether the maximum number of messages has been reached or the maximum amount of time has passed.</p>
ADJ	LIGHT_TRACER	THREAD_INFO	N	The Boolean property indicating whether to include the thread information in the trace messages for every module and sub-module at start-up.
ADJ	LIGHT_TRACER	TIME_STAMP	N	The Boolean property indicating whether to include the time stamp in the trace messages for every module and sub-module at start-up.
ADJ	LIGHT_TRACER ASSERT	ENABLED	N	The Boolean property indicating whether to turn on the Assertion at the current call point.
ADJ	LIGHT_TRACER ASSERT	IS_ABORT	N	The Boolean property indicating whether to turn on the abortion if an assertion failed at the current call point.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ADJ	LIGHT_TRACER.FLOW_TRACE	BUFFER_SIZE	2,000 (slots)	The size of the flush buffer of the Flow Tracer. The Flow Tracer is a Light Tracer that traces a single flow or an event that is transferred between machines. When a Flow Tracer receives a request to flush the data to a file, it writes this request to the buffer. Every request occupies two buffer slots. This buffer is allocated once at initialization.
ADJ	LIGHT_TRACER.FLOW_TRACE	ENABLED	N	This Boolean property indicates whether to turn on the Flow Tracer functionality. The Flow Tracer traces a single flow or an event that is transferred between machines.
ADJ	LIGHT_TRACER.FLOW_TRACE	FILE_PATH	\${ABP_APP_LOG}/flowtrace/	The full path to the location of the Flow Tracer files.
ADJ	LIGHT_TRACER.FLOW_TRACE	FLOW_TRACE_BUFFER_SIZE	1024 (*64 bit)	The buffer size of the Flow Tracer (in 64-bit slots). The traces for each flow are written into its assigned buffers. This parameter determines the size of each such buffer.
ADJ	LIGHT_TRACER.FLOW_TRACE	INITIAL_BUFFERS_AMOUNT	10	The number of buffers initially allocated in the buffer pool.
ADJ	LIGHT_TRACER.FLOW_TRACE	INITIAL_FLOW_TRACERS_AMOUNT	10	The number of Flow Tracers initially allocated in the Flow Tracer pool.
ADJ	LIGHT_TRACER.FLOW_TRACE	MAX_READ_ATTEMPTS	20	An internal parameter of the Light Tracer. It specifies the maximum number of attempts that the reader makes to read the flushed Flow Tracer. After this number of attempts, it is assumed that the writer is stuck, and the reader synchronizes with the current writer.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ADJ	LIGHT_TRACER.FLOW_TRACE	MAXIMUM_BUFFERS_AMOUNT	-1	Limits the number of buffers that can be allocated for Flow Tracers. When a Flow Tracer requests a buffer, and there are no buffers in the buffer pool, a new buffer is allocated. However, if the number of allocated buffers has reached the limit specified by this parameter, no new buffers are allocated. This means that traces are lost. If the value of this parameter is set to -1, there is no limit on the number of buffers.
ADJ	LIGHT_TRACER.FLOW_TRACE	MAXIMUM_FLOW_TRACERS_AMOUNT	-1	Limits the number of Flow Tracers that can be allocated. When a Flow Tracer is requested, and there are no Flow Tracers in the Flow Tracer pool, a new Flow Tracer is allocated. However, if the number of allocated Flow Tracers has reached this limit, no new Flow Tracers are allocated. If the value of this parameter is set to -1, there is no limit on the number of Flow Tracers.
ADJ	LIGHT_TRACER.FLOW_TRACE	SHUT_DOWN_CHECK_FREQ	1000	A parameter that functions similarly to the SHUT_DOWN_CHECK_FREQ parameter of the LIGHT_TRACER. This parameter defines the frequency at which the Flow Tracer checks whether one of the limits for its shutdown has been reached. If one of these stop conditions is met, the Flow Tracer mechanism is turned off.
ADJ	LIGHT_TRACER.FLOW_TRACE	THRESHOLD_BUFFERS_AMOUNT	-1	When a buffer is returned to the buffer pool, and the number of allocated buffers is larger than the value of this threshold, the buffer is de-allocated. If the value of this threshold is set to -1, there is no threshold limitation. In this case, the buffer pool does not de-allocate buffers at run time.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ADJ	LIGHT_TRACER.FLOW_TRACE	THRESHOLD_FLOW_TRACERS_AMOUNT	-1	When a Flow Tracer is returned to the Flow Tracer pool, and the number of allocated Flow Tracers is larger than the value of this threshold, the Flow Tracer is deallocated. If the value of this threshold is set to -1, there is no threshold limitation. In this case, the Flow Tracer pool does not de-allocate Flow Tracers at run time.
ADJ	LIGHT_TRACER.LOG_IN_TRACE	ENABLED	<code> \${ADJ_LIGHT_TRACER_ENABLE}</code>	The Boolean property that indicates whether to turn on the log in trace functionality and merge the log and trace messages into a single binary file. This property is valid only when the <code>GNRL_LIGHT_TRACE</code> parameter is defined as one of the Logger outputs.
ADJ	LIGHT_TRACER.TRACE	ENABLED	N	The Boolean property indicating whether to turn on the Light Tracer at the current call point.
ADJ	Logger	Enable	<code> \${ADJ_LOGGER_ENABLE}</code>	Enables the creation of log messages via the Logger mechanism.
ADJ	Logger	ImmediateFlushSeverity	50000	The minimum level at which immediate flush occurs. If a log item with a higher level than the value of the <code>ImmediateFlushSeverity</code> parameter is encountered, all the buffered items along with the current item are immediately flushed into a log file.
ADJ	Logger	Severity	<code> \${ADJ_LOGGER_SEVERITY}</code>	The message severity threshold for the log. The severity range is 0–50000 according to the standard Logger definitions: <ul style="list-style-type: none">■ 0 – Trace■ 10000 – Debug■ 20000 – Info■ 30000 – Warning■ 40000 – Error■ 50000 – Fatal

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ADJ	Logger.ADJ.IWD_ALERTS	Enable	true	Enables the creation of alert messages.
ADJ	Logger.ADJ.IWD_ALERTS	Outputs	IWD_ALERTS_OUTPUT, GNRL_LOG,GN RL_LIGHT_TRA CE	The output object assigned to alert messages. Alerts are printed to the AVM1_ALERTS table and to the regular process logs.
ADJ	Logger.ADJ.IWD_ALERTS	Severity	0	The severity threshold for creating an alert message and sending it to the Availability Manager.
ADJ	Logger.ADJ.MEMDB.ARENA	BufferSize	\${BUFFER_SIZE}	The size of the temporary buffer of the log.
ADJ	Logger.ADJ.MEMDB.ARENA	Class	SimpleFileOutput	The template class.
ADJ	Logger.ADJ.MEMDB.ARENA	Enable	\${ADJ_LOGGER_MEMDB_AR ENA_ENABLE}	Enables the arena log output, which is usually printed to the application log.
ADJ	Logger.ADJ.MEMDB.ARENA	Filename	\${ABP_APPL_OG}/ \${GN1_TAS_K_NAME}_ \${A PPLICATION_I D}_ %D_ %T_ %n.log	The name of the log file.
ADJ	Logger.ADJ.MEMDB.ARENA	FlushPeriod	\${FLUSH_PERIOD}	The interval at which the data from the temporary buffer is flushed to a file.
ADJ	Logger.ADJ.MEMDB.ARENA	ImmediateFlush	\${IMMEDIATE_FLUSH}	Indicates whether the data is flushed to a file immediately and not periodically.
ADJ	Logger.ADJ.MEMDB.ARENA	Layout	Pattern	The layout type that the log uses to format messages.
ADJ	Logger.ADJ.MEMDB.ARENA	MaxFileSize	\${MAX_FILE_SIZE}	The maximum size of a log file.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ADJ	Logger.ADJ.MEM DB.ARENA	Mode	\${MODE}	The working mode of the log: synchronous or asynchronous.
ADJ	Logger.ADJ.MEM DB.ARENA	Outputs	coutlog	The output object assigned to the log.
ADJ	Logger.ADJ.MEM DB.ARENA	Severity	\${ADJ_LOGGER_MEMDB_AR ENA_SEVERITY}	The message severity threshold for the log. The severity range is 0–50000 according to the standard Logger definitions: <ul style="list-style-type: none"> ■ 0 – Trace ■ 10000 – Debug ■ 20000 – Info ■ 30000 – Warning ■ 40000 – Error ■ 50000 – Fatal
ADJ	Logger.LIGHT_T RACER	Enable	\${ADJ_LOGGER_LIGHT_TRACER_ENABLE}	Enables the creation of Light Tracer logs.
ADJ	Logger.LIGHT_T RACER	Severity	\${ADJ_LOGGER_LIGHT_TRACER_SEVERITY}	The message severity threshold for the log. The severity range is 0–50000 according to the standard Logger definitions: <ul style="list-style-type: none"> ■ 0 – Trace ■ 10000 – Debug ■ 20000 – Info ■ 30000 – Warning ■ 40000 – Error ■ 50000 – Fatal

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ADJ	Logger.output.GN RL_LOG	Class	SimpleFileOutput	<p>The template class. To filter log messages by time frame, set the value of the parameter to ‘FilteringSimpleFileOutput’ and define the following filtering parameters:</p> <ul style="list-style-type: none"> ▪ MessageFilterActive ▪ MessageFilterRepeatThreshold ▪ MessageFilterTimeThreshold <p>Although the filtering mechanism saves time on I/O, it may have an impact on performance because of the time it takes to check the messages for uniqueness and filter them.</p>
ADJ	Logger.output.GN RL_LOG	Filename	<code> \${ABP_APP_LOG}/\${GN1_TAS_K_NAME}_\${APPLICATION_ID}_%D_%T_%n.log</code>	The full path to the location of the process logs.
ADJ	Logger.output.GN RL_LOG	MessageFilterActive	true	Indicates whether the message filtering mechanism is active.
ADJ	Logger.output.GN RL_LOG	MessageFilterRepeatThreshold	1	The maximum number of repeated messages that can be printed in the period defined by the MessageFilterTimeThreshold parameter. For example, out of the box, only one unique instance of a message (according to its parameters and pattern) is printed within the period of 2 seconds. Any subsequent instance of the same message is not printed.
ADJ	Logger.output.GN RL_LOG	MessageFilterTimeThreshold	2	The period of time in which the repetition counter is checked. When the period is over, the counter is reset.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ADJ	Logger.output.GN RL_TRACE	Filename	<code> \${ABP_APPR_LOG}/\${GN1_TAS K_NAME}_TRA CE_\${APPLICA TION_ID}_%D_%T_%n.log</code>	The full path to the location of the process traces.
ADJ	Logger.output.IW D_ALERTS_OUTPUT	Class	IWDAlertsOutput	The template class.
ADJ	Logger.output.IW D_ALERTS_OUTPUT	Mode	Synch	The working mode: synchronous or asynchronous.
ADJ	Tracer	Enable	<code> \${ADJ_TRACER_ENABLE}</code>	Enables the creation of trace messages via the Logger mechanism.
ADJ	Tracer	Severity	<code> \${ADJ_TRACER_SEVERITY}</code>	The message severity threshold for the log. The severity range is 0–50000 according to the standard Logger definitions: <ul style="list-style-type: none"> ■ 0 – Trace ■ 10000 – Debug ■ 20000 – Info ■ 30000 – Warning ■ 40000 – Error ■ 50000 – Fatal
AGD	config	AGD1_ROUTE_TO_OLD_CYCLE	Y	If this parameter is set to ‘Y’, an event that in line with the logic must be guided according to the new cycle, is nevertheless guided according to the old cycle.
AGD	config	AGD1_USE_CACHE	Y	The Guiding mode, which indicates whether it uses cache (Y) or works only with the database (N).
AGD	config	AVERAGE_RESOURCES_PER_SUBSCRIBER	3	The average number of resources per subscriber, which is used to calculate hash sizes for the Guiding segment hash table.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
AGD	config	EXTENSION_PERIOD	0	The additional period in days that is to be subtracted from the effective date of a relevant entry in the AGD1_RESOURCES table if a subscriber was not found. (A period after the expiration date in which event detail records are valid.)
AGD	config	EXTENSION_PERIOD_PER_EVENT_TYPE	N/A	<p>Contains a list of event types and their specific extension periods for guiding to customer, in the following format:</p> <p><i><Event Type ID in Rating Logic Configurator>=<extension period in days>; <Event Type ID in Rating Logic Configurator>=<extension period in days>;...</i></p> <p><i>Example:</i> <i>'3841=3;76427=1;9898=4'</i></p> <p>If the Event Type ID as defined in the Rating Logic Configurator is not set in the mini-parse (CR) stage of the protocol, or if this parameter is not defined for a specific event type, the general extension period (as specified by the EXTENSION_PERIOD parameter) is used.</p> <p>This parameter applies both to offline and online events.</p>
AGD	config	GRACE_PERIOD	0	The additional period, in days, that is to be added to the effective date of a relevant entry in the AGD1_RESOURCES table if a subscriber was not found. (A period before the effective date in which event detail records are valid.)

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
AGD	config	GRACE_PERIOD_PER_EVENT_TYPE	N/A	<p>Contains a list of event types and their specific grace periods for guiding to customer, in the following format:</p> <p><i><Event Type ID in Rating Logic Configurator>=<grace period in days>; <Event Type ID in Rating Logic Configurator>=<grace period in days>;...</i></p> <p><i>Example:</i> '3841=3;76427=1;9898=4'</p> <p>If the Event Type ID as defined in the Rating Logic Configurator is not set in the mini-parse (CR) stage of the protocol, or if this parameter is not defined for a specific event type, the general grace period (as specified by the GRACE_PERIOD parameter) is used. This parameter applies both to offline and online events.</p>
AGD	config	HASH_TABLE_FACTOR	3	Another variable in the calculation of the size of resource hash tables. This is the desired ratio between the hash table entry count and resource count. For example, 2 means that there must be twice as many entries as resources.
AGD	config	HASH_TABLE_ROUNDING_PERCENT	30	The size of the resource segment hash table is always a power of 2 (bytes). This size (X) is between two powers of 2, X_SMALL and X_LARGE. If X reduced by the rounding percent of X is less than or equal to X_SMALL, X_SMALL is used; otherwise, X_LARGE is used.
AGD	config	TOTAL_NUMBER_OF_SUBSCRIBERS	\${AGD_TOTAL_NUMBER_OF_SUBSCRIBERS}	The total number of subscribers to be handled by all Guiding to Customer servers (used to calculate hash sizes for the Guiding segment hash table).
APE	config	AIMOS_FORMATTING_DD_LIB_NAME	mpr_aimosFormatting	The name of the library that contains the data dictionary of the Reconciliation tool.
APE	config	BUSINESS_LOAD_FROM_COVERAGE	356	The number of days before the system date for which the Offer Catalog is loaded.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APE	config	BUSINESS_LOAD_TO_COVERAGE	356	The number of days after the system date for which the Offer Catalog is loaded.
APE	config	CONSIDER_OFFER_TIME_FOR_PRORATE	TRUE	A parameter of the Proration extension function, which includes the offer time in the proration calculation.
APE	config	CYCLE_LOAD_FROM_COVERAGE	90	The number of days before the system date for which the cycle is loaded.
APE	config	CYCLE_LOAD_TO_COVERAGE	90	The number of days after the system date for which the cycle is loaded.
APE	config	DUP_CHECK_GROWTH_RATE	2.0	For the Duplicate Check mechanism. The rate at which the array of event start times grows. For example, if the value of this parameter is 2.0, every time the array of event start times needs to grow, it doubles its size.
APE	config	DUP_CHECK_INIT_SIZE	16	For the Duplicate Check mechanism. The initial size of the array of event start times in memory.
APE	config	DUP_CHECK_MAX_SIZE	256	For the Duplicate Check mechanism. The maximum number of event start times that the mechanism keeps in memory.
APE	config	DUP_CHECK_TIME_DELTA	60	For the Duplicate Check mechanism. To account for possible machine time differences between a particular Event Server, the network element, and other Event Servers it is replacing (in high availability scenarios), the Event Server does not begin saving event start times immediately upon initiation. Instead, the value of this time delta parameter (in seconds) is added to the current time. Only when the resultant time arrives does the Event Server begin to store the event start times in memory (possibly without checking them against the database if they do not exist in the array).
APE	config	EFFECTIVENESS_VERSION_HELPER	A	Treat all vertical elements as absolute (A) or relative (R). For more information, see <i>Rating Logic Configurator User Guide</i> .

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APE	config	ENABLE_TRACE_XML_VALIDATION	True	Indicates whether to perform validation checks on the data that is streamed to the Event Trace XML file.
APE	config	EVENT_TRACING_DIR	\${ABP_APP_LOG}	The location of all trace files. The Event Trace XML file and the Protocol Trace XML file are placed together with the logs of the Event Server.
APE	config	GRACE_AND_EXTENSION_HELPER	OfferDates	Determines how grace and extension periods are used in guiding to service. Values are: <ul style="list-style-type: none"> ▪ <i>OfferDates</i> – In the case of an extension period, the offer whose expiration date is the closest to the event start time is taken. In the case of a grace period, the offer that has the earliest effective date is taken. ▪ <i>EventTime</i> – If no subscriber offers are found, the grace and extension periods are used to move the event time and search again.
APE	config	GUIDING_TO_SERVICE_EXTENSION_PERIOD	0	The Guiding to Service extension period in days.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APE	config	GUIDING_TO_SERVICE_EXTENSION_PERIOD_PER_EVENT_TYPE	N/A	<p>Contains a list of event types and their specific extension periods for guiding to service, in the following format:</p> <p><i><Event Type ID in Rating Logic Configurator>=<extension period in days>; <Event Type ID in Rating Logic Configurator>=<extension period in days>;...</i></p> <p><i>Example:</i> '3841=3;76427=1;9898=4'</p> <p>If the Event Type ID as defined in the Rating Logic Configurator is not set in the mini-parse (CR) stage of the protocol, or if this parameter is not defined for a specific event type, the general extension period (as specified by the GUIDING_TO_SERVICE_EXTENSION_PERIOD parameter) is used.</p> <p>This parameter applies both to offline and online events.</p>
APE	config	GUIDING_TO_SERVICE_GRACE_PERIOD	0	The Guiding to Service grace period in days.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APE	config	GUIDING_TO_SERVICE_GRACE_PERIOD_PER_EVENT_TYPE	N/A	<p>Contains a list of event types and their specific grace periods for guiding to service, in the following format: <i><Event Type ID in Rating Logic Configurator>=<grace period in days>; <Event Type ID in Rating Logic Configurator>=<grace period in days>;...</i></p> <p><i>Example:</i> '3841=3;76427=1;9898=4'</p> <p>If the Event Type ID as defined in the Rating Logic Configurator is not set in the mini-parse (CR) stage of the protocol, or if this parameter is not defined for a specific event type, the general grace period (as specified by the GUIDING_TO_SERVICE_GRACE_PERIOD parameter) is used.</p> <p>This parameter applies both to offline and online events.</p>
APE	config	GUIDING_TO_SERVICE_ROLLING_ACC_EXTENSION_PERIOD	1	The extension period for a rolling accumulator, in days.
APE	config	GUIDING_TO_SERVICE_ROLLING_ACC_GRACE_PERIOD	1	The grace period for a rolling accumulator, in days.
APE	config	IGNORE_ONE_DAY_OFFER	Y	<p>Determines whether to grant an allowance if the allowance offer becomes effective and expires on the same day:</p> <ul style="list-style-type: none"> ■ <i>Y</i> – The allowance is zero (not granted). ■ <i>N</i> – One day of the allowance is granted.
APE	config	IGNORE_SUSPEND_RESUME_ON_GET_OFFER_DATE	Y	When the APE1_SUBSCR_OFFERS table contains multiple offer entries (because of suspend and resume activities), indicates whether the Get offer expiration date by instance extension function and the Get offer subscription date by instance extension function must always return the qualified entry ('Y'), or the last offer for the expiration date and the first offer for the subscription date ('N').

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APE	config	INCLUDE_OFFER_EXP_DATE	TRUE	A parameter of the Proration extension function, which includes the offer expiration date in the proration calculation.
APE	config	PRORATION_INCLUDE_OFFER_EXP_DATE	True	Indicates whether the offer expiration date is included in proration.
APE	config	PRORATION_INCLUDE_TIME_OF_DAY	True	Indicates whether the time of day is used in proration.
APE	config	REVENUE_TYPES_LIST	'UC'	The revenue types of items to be used by the Rating Engine.
APE	config	RUN_FORMAT_ON_TRACE_XML	False	Indicates whether to run the transformation on the Event Trace XML file.
APE	config	SHORT_CYCLE_PER_CYCLE_FREQUENCY_PERIOD	N/A	<p>The short cycle period per cycle type and cycle multiplier. If after a cycle change, the difference between the cycle close date of the new cycle and the cycle close date of the old cycle is less than or equal to the number of days defined for the new cycle type and its multiplier, the events are guided to the next cycle instance. In this case, the customer receives a combined bill for the short cycle instance and the next full cycle instance.</p> <p>The value of this configuration parameter consists of semicolon-separated groups of the following comma-separated parameters:</p> <ul style="list-style-type: none"> ■ Cycle type ('W' for weekly or 'M' for monthly) ■ Cycle multiplier ■ The maximum number of days until the cycle close day that qualify for a short cycle instance <p><i>Example:</i> W,2,0;M,1,14;M,2,30;M,3,45;</p>
APE	config	USE_OFFER_VERSIONING	N	Indicates whether offer versioning is enabled.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APE	config	XSLT_FORMATTER_NAME	\${ABP_BIN}/trace-xml-transform.xsl	The full path to the XSLT stylesheet that is to be used to format the Event Trace XML file.
APE/APR	config	EVENT_TRACING_MODE	\${APE_EVENT_TRACING_MODE}/\${APR_EVENT_TRACING_MODE}	A parameter that enables event tracing.
APR	config	ABORT_ON_REJECTED_ROLLING_EVENT	Y	Indicates whether rolling events must be rejected in rereate flows.
APR	config	ACC_SELECT_BUFFER_SIZE	100	The size of the buffer to fetch for each accumulator.
APR	config	ACCUM_INSERT_BUFFER_SIZE	\${APR_ACCUM_INSERT_BUFF_SIZE}	For the Persistence Writer Function. The size of the OCI Oracle buffer for keeping accumulators.
APR	config	AGGREGATE_EVENT_ENABLED	false	Indicates whether event aggregation is enabled. If this parameter is set to ‘true’, the Duplicate Check mechanism must work with the special index-organized table (IOT) (see DUPCHECK_MODE_IS_IOT); otherwise, the Event Server fails upon initialization.
APR	config	APRSR_DEOLDREPSESS_CHECKCUSTOMER_TIMEPERIOD	\${APR_SR_DELOLDREPSESS_CUSTOMERTIMEPERIOD}	The frequency with which a subscriber’s time stamp (updated every time a subscriber is accessed) under a customer is to be checked. A replication session can only be deleted when the customer is locked. To prevent unnecessary locks, the scan period is checked first. When the difference between the current time and the latest time stamp of the subscriber is greater than this parameter, the customer is locked and scanned for old replication sessions, which are subject to removal.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	config	APRSR_DEOLDREPSESS_HOLDSESSION_TIMEPERIOD	\${APR_SR_DEL_OLDREPSESS_HOLDSESSION_TIMEPERIOD}	The period of time for which the replication session is kept in memory (in milliseconds). When the difference between the current time and the creation time of the replication session is greater than or equal to the value of this parameter, the replication session is deleted.
APR	config	APRSR_DEOLDREPSESS_SCANCUSTOMERS_TIMEPERIOD	\${APR_SR_DEL_OLDREPSESS_SCANCUSTOMERS_TIMEPERIOD}	The frequency with which customers are to be scanned. Customers are scanned when the difference between the current time and the time stamp of the last scan is greater than this parameter, and not after each timeout of the maintenance thread.
APR	config	AVM_ENABLED	N	Defines whether the Event Server is working through the Availability Manager (AVM). If the Event Server is configured to be activated though the Availability Manager, it must be configured to work with shared memory and not virtual memory (see VIRTUAL_OR_SHARED_MEMORY_USED).
APR	config	CONN_EXTERNAL_RECV_BUF_SIZE_CONNECTION	\${APR_CONN_EXTERNAL_RECV_BUF_SIZE_CONNECTION}	The size of the TCP buffer that is used for receiving messages through an external connection (to an external network element).
APR	config	CONN_EXTERNAL_SEND_BUF_SIZE_CONNECTION	\${APR_CONN_EXTERNAL_SEND_BUF_SIZE_CONNECTION}	The size of the TCP buffer that is used for sending messages through an external connection (to an external network element).
APR	config	CONN_EXTERNAL_SLEEP_ON_BUSY	0	The sleep time if no messages arrive through an external connection (to an external network element).
APR	config	CONN_INTERNAL_CLIENT_RECV_BUFFER_SIZE	\${APR_CONN_INTERNAL_CLIENT_RECV_BUFFER_SIZE}	The size of the buffer that is used for receiving messages through a connection to an Event Server client, such as File2E.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	config	CONN_INTERNAL_CLIENT_RECV_BUFSIZE_CONNECTION	\${APR_CONN_INTERNAL_CLIENT_RECV_BUFSIZE_CONNECTION}	The size of the TCP buffer that is used for receiving messages through a connection to an Event Server client, such as File2E.
APR	config	CONN_INTERNAL_CLIENT_SEND_BUFSIZE	\${APR_CONN_INTERNAL_CLIENT_SEND_BUFSIZE}	The size of the buffer that is used for sending messages through a connection to an Event Server client, such as File2E.
APR	config	CONN_INTERNAL_CLIENT_SEND_BUFSIZE_CONNECTION	\${APR_CONN_INTERNAL_CLIENT_SEND_BUFSIZE_CONNECTION}	The size of the TCP buffer that is used for sending messages through a connection to an Event Server client, such as File2E.
APR	config	CONN_INTERNAL_CLIENT_SLEEP_ON_BUSY	0	The sleep time if no messages arrive through a connection to an Event Server client, such as File2E.
APR	config	CONN_INTERNAL_RECV_BUF_SIZE_CONNECTION	\${APR_CONN_INTERNAL_RECV_BUF_SIZE_CONNECTION}	The size of the TCP buffer that is used for receiving messages through an internal connection (between Event Servers).
APR	config	CONN_INTERNAL_SEND_BUF_SIZE_CONNECTION	\${APR_CONN_INTERNAL_SEND_BUF_SIZE_CONNECTION}	The size of the TCP buffer that is used for sending messages through an internal connection (between Event Servers).
APR	config	CONN_INTERNAL_SLEEP_ON_BUSY	0	The sleep time if no messages arrive through an internal connection (between Event Servers).
APR	config	CONN_NETWORK_RECV_MAX_WAIT_SEC	\${APR_CONN_NETWORK_RECV_MAX_WAIT_SEC}	The maximum number of seconds to try to read a buffer from a stream, as long as at least one byte is received per second.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	config	CONN_NETWORK_SEND_MAX_WAIT_SEC	\${APR_CONN_NETWORK_SE ND_MAX_WAI T_SEC}	The maximum number of seconds to try to send a buffer to a stream, as long as at least one byte is sent per second.
APR	config	CORE_DUMP_ENABLED	\${CORE_DUMP _ENABLED}	Indicates whether the core dump file must be created if the Event Server fails. By default, the creation of the core dump file is bypassed, and only a stack trace of the failure is written to the console log. To enable the regular flow with the core dump file, this parameter must be set to 'Y'.
APR	config	CUSTOMER_CACHE_USE	<ul style="list-style-type: none"> ■ Y – Hybrid (default) ■ N – Database Only 	<p>By default, the Event Server works in Hybrid mode. In this mode, the Event Server looks for customer data (offer parameters) in AIMOS. If the data is not found in AIMOS, the Event Server checks the database. For this mode, no special configuration parameter is needed. However, if for some reason, a communications service provider wishes to get customer data from the database only (for example, if there are problems with the DB2E process), this parameter must be added to the APR1_CONF_SECTION_PARAM table.</p> <p> Caution: <i>Working in the Database Only mode is not recommended. This mode negatively affects performance and is not supported; therefore, the results of using it are unpredictable.</i></p>
APR	config	CUSTOMER_GROUP_HASH_SIZE	100000	The size of the hash table for a customer group. This size must be close to <the number of customers in the cycle> / <the number of customer groups per the cycle>. This parameter affects performance.
APR	config	CUSTOMER_RERATE_GROUP_HASH_SIZE	100	The size of the hash table for a Rerate customer group.
APR	config	CUSTOMER_RO_RATER_GROUP_HASH_SIZE	100	The size of the hash table for a Read-Only Rater customer group.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	config	DB_FAIL_THRESHOLD	3	<p>The maximum number of consecutive failed attempts to reconnect to the home database, after which the Event Server switches to work with the alternate database. This number includes the attempts made by all the connections with the same connection string, and it is zeroed when any of the connections is successful. The DB_FAIL_THRESHOLD parameter is used together with the PW_HIGH_WATERMARK parameter.</p> <p>If the DB_FAIL_THRESHOLD parameter is missing or set to 0, the default value of 3 is used.</p>
APR	config	DEFAULT_SESSION_TTL	240	The default time to live (TTL) value for an Event Server session (in seconds).
APR	config	DIAMETER_PARSER_VALIDATION_ENABLED	N	<p>Indicates whether the Parser must perform validations on the input network buffer. The data content of the network request is validated during the parsing of the request. The Diameter Data Dictionary includes the information required for message content validation, such as the permitted range of an attribute, vendor-specific information, or message and attribute flags. There are only two validation modes: either all validations are performed or no validations are performed:</p> <ul style="list-style-type: none"> ■ <i>Y</i> – The Parser validates the events in terms of their compliance with the Diameter protocol. This mode can be used in the pre-production stage. ■ <i>N</i> – No validations are performed. This mode must be used in production to avoid impact on performance.
APR	config	DSP_COMMIT_ACK_TIMEOUT	1000	The timeout for the commit acknowledgement message from the Dispatcher process (in milliseconds).
APR	config	DSP_FORCE_DEGRADED_BUSY_TIM_EOUT	30	The time period of the Degraded mode if the Dispatcher is overloaded (in seconds).

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	config	DSP_FORCE_DEGRADED_DB_FAILURE_TIMEOUT	60	The time period of the Degraded mode in the case of a database connection failure on the Dispatcher side (in seconds).
APR	config	DUMMY_APPL_ID	9999	For the protocol error mechanism. This dummy parameter is used if the error was not reported.
APR	config	DUMMY_ERROR_CODE	9999	For the protocol error mechanism. This dummy parameter is used if the error was not reported, or the ADJ1_PROTOCOL_ERRS_CFG table is empty.
APR	config	DUMMY_MSG_ID	9999	For the protocol error mechanism. This dummy parameter is used if the error was not reported.
APR	config	DUP_CHECK_PRELOAD_FUTURE_EVENTS_TOLERANCE	0	<p>The time frame (in seconds) beyond the system time (or beyond the logical time, when the logical time is used) that the array of event start times must include when it is initially created in the shared memory. In other words, the initial array of event start times covers the following time frame:</p> <ul style="list-style-type: none"> ■ <i>Maximum time</i> – The system time when the array was created (or the logical time, when logical time is used) plus the value of the DUP_CHECK_PRELOAD_FUTURE_EVENTS_TOLERANCE parameter ■ <i>Minimum time</i> – The maximum time minus the value of the DUP_CHECK_WIN_SIZE parameter <p>These are the <i>initial</i> minimum and maximum times. The time frame that the array covers is shifted later during processing.</p>

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description																
APR	config	DUP_CHECK_WIN_SIZE	18	<p>For the Duplicate Check mechanism. Indicates for how long (in hours) event start times are kept in memory. Any event that started more than this number of hours ago is always checked against the database and not against the memory.</p> <p>This parameter, which defines the duplicate check time frame, affects the duplicate check time unit. The duplicate check time unit defines the smallest period of time in which two similar events cannot be identified as non-duplicates. This means that if two events of the same type arrive within this number of seconds, they are automatically considered as duplicate. An increased time frame provides better backward coverage for the duplicate check; however, it reduces the resolution of the check. The following table describes this trade-off.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Time Frame (Hours)</th><th>Time Unit (Seconds)</th></tr> </thead> <tbody> <tr><td>18</td><td>1</td></tr> <tr><td>36</td><td>2</td></tr> <tr><td>72</td><td>4</td></tr> <tr><td>144</td><td>8</td></tr> <tr><td>288</td><td>16</td></tr> <tr><td>576</td><td>32</td></tr> <tr><td>1152</td><td>64</td></tr> </tbody> </table>	Time Frame (Hours)	Time Unit (Seconds)	18	1	36	2	72	4	144	8	288	16	576	32	1152	64
Time Frame (Hours)	Time Unit (Seconds)																			
18	1																			
36	2																			
72	4																			
144	8																			
288	16																			
576	32																			
1152	64																			
APR	config	DUPCHECK_MODE_IS_IOT	N	Indicates whether the Duplicate Check mechanism must work with the special index-organized table (IOT). This type of duplicate check is not optimal in terms of performance because it requires the rating process to update an additional table. Therefore, it is to be used only when necessary.																

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	config	EAGER_LOAD_ACCUM_SELECT_BUFFER_SIZE	1000	The number of accumulators that the eager load process can load from the APE1_ACCUMULATORS table into the OCI Oracle buffer.
APR	config	ERROR_ENTITY_INSERT_BUFFER_SIZE	1000	The size of the error entity buffer in the data layer.
APR	config	EVENT_LOG_DELIMITER	,	The delimiter between the fields in the event log. By default, the delimiter is ','. To change the delimiter, this parameter must be added to the CFG1_CONF_SECTION_PARAM table.
APR	config	EVENT_SNIFFER_DEFAULT_DURATION_TIME_TO_RECORD	d001000	The period of time for which the trace is to be active, in 'HHMMSS' format.
APR	config	EVENT_SNIFFER_OUTPUT_PATH	\${ABP_AP4L0G}	The location of the output files of the Event Sniffer.
APR	config	EVENTS_INSERT_BUFFER_SIZE	1000	For the Persistence Writer Function. The size of the OCI Oracle buffer for keeping rated events.
APR	config	EXTERNAL_EVENT_SLA_MS	\${APR_EXTER_NAL_EVENT_SLA_MS}	Service Level Agreement (SLA) duration in milliseconds. If the time that an event spent in the Rater queue exceeds this SLA, the event is dropped in the Event Processing Module. A null (0) value disables this feature. Disabling it affects the host failover KPI.
APR	config	EXTERNAL_UDP_NW_ENABLED	Y	Indicates whether the External UDP Manager is enabled ('Y') or disabled ('N'). Sending Diameter events over a UDP connection is allowed only if this parameter is set to 'Y'. If there no special configuration that enables sending online events via a UDP port without a response, this parameter must be set to 'N'.
APR	config	FLUSH_THRESHOLD	\${APR_FLUSH_THRESHOLD}	After this number of records has been processed, the events are inserted into the Rated Events table. The value of this parameter cannot exceed 64 KB.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	config	FR_BLOCKING_THREAD_ENABLED	\${FR_BLOCKIN G_THREAD_EN ABLED}	Indicates whether guiding to customer in the Network thread (a blocking thread) is enabled.
APR	config	GENERAL_DUPCHECK_MODE_ENAB LED	\${GENERAL_D UPCHECK_MO DE_ENABLED}	Indicates whether the general Duplicate Check mechanism is enabled. Values are: <ul style="list-style-type: none"> ■ <i>Y</i> – A duplicate check is performed if it is required by the Message-Based Flow or if it is enforced by the client (for example, when File2E sends Rerun files). In IOT mode, the event is inserted into the index-organized table (IOT) for duplicate checks. ■ <i>N</i> – A duplicate check is not performed even if it is required by the Message-Based Flow. However, a duplicate check is still performed if it was forced by the client (for example, when File2E sends Rerun files). If a duplicate check is required by the Message-Based Flow or if it was forced by the client, the event is inserted into the IOT for duplicate checks.
APR	config	GR_ENABLED	\${GR_ENABLE D}	Indicates whether Geo Redundancy support is enabled.
APR	config	INTERNAL_COM_IN_SG	false	Indicates whether the Event Server sends messages in bulks or one by one.
APR	config	INTERNAL_COM_SG_MAX_MESSAG ES	16	If the Event Server sends messages in bulks, determines the maximum number of messages in a bulk.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	config	LF_FIFO_SLEEP_MILI_INTERVAL	0	<p>The locking strategy for the critical sections of the queue if there are no events between the fetch tries of the dequeue function. Possible values are:</p> <ul style="list-style-type: none"> ■ <i>0 (default)</i> – Use the tryacquire and yield functions in the case of a retry. ■ <i>999 milliseconds and up</i> – Use the acquire function. ■ <i>Any other value</i> – Use the tryacquire function and sleep for the specified period of time (in milliseconds) in the case of a retry.
APR	config	LOGICAL_DATE_DYNAMIC_MODE_IND	N	Indicates whether the logical time set must be synchronized with the process start time and increased with the clock. If the value is ‘N’ (non-dynamic mode), the value of the logical time always stays the same. This parameter is used only if USE_LOGICAL_TIME = ‘Y’.
APR	config	LOGICAL_TIME_DELTA_SECS	0	The number of seconds to be added to or reduced from the logical time set. This parameter can be used to simulate a time zone.
APR	config	LOGICAL_TIME_MODE	GMT	Indicates which logical time (local or GMT) to use if USE_LOGICAL_TIME = ‘N’.
APR	config	MAPPER_DATA_UPGRADER_ENABLED	Y	<p>Enables (‘Y’) or disables (‘N’) the Implementation Compiler Data Upgrader.</p> <p>When the Implementation Compiler is refreshed to work with a new version while the Event Server is running, the memory buffers are updated according to the new Implementation Compiler definitions only if the Implementation Compiler Data Upgrader is enabled. If the Refresh Libraries on the fly functionality is required, the Implementation Compiler Data Upgrader must be enabled.</p>

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	config	MARK_FOR_RERATE_INSERT_BUFFER_SIZE	1000	For the Persistence Writer Function. The size of the OCI Oracle buffer for keeping special records with data for the Subscriber Rerate table.
APR	config	MAX_SHARED_ACCUM_SESSIONS_LIST_SIZE	512	The maximum number of sessions per accumulator in the customer-level hash of sessions sharing the accumulator.
APR	config	MEMORY_ALERT_THRESHOLD	90	The percentage of AIMOS usage that triggers an alert.
APR	config	MEMORY_arena_CHUNK_SIZE_IN_BYTES_PART1	16	<p>The minimal memory arena allocation block in bytes. This is the resolution of the memory arena, which serves as the basis for calculating the total size of the shared memory arena. The default value is 16 bytes to support up to 64 Gb for a shared memory segment.</p> <p>To extend the shared memory range, the property must be set as follows:</p> <ul style="list-style-type: none"> ■ 32 bytes to support up to 128 Gb for a shared memory segment ■ 64 bytes to support up to 256 Gb for a shared memory segment ■ 128 bytes to support up to 512 Gb for a shared memory segment
APR	config	MEMORY_SIZE_IN_BYTES	\${APR_MEMORY_SIZE_IN_BYTES}	<p>The requested size of the shared memory. It is recommended that the size be a multiple of 256 MB.</p> <p>This parameter is used in conjunction with the VIRTUAL_OR_SHARED_MEMORY_USED parameter.</p>
APR	config	MODULE_NAME	ES	A name given to an Oracle session. Oracle restricts the resources for this particular session (module).
APR	config	NOTIFICATION_BUFFER_SIZE	1000	For the Persistence Writer Function. The size of the OCI Oracle buffer for keeping notifications.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	config	NUM_OF_CUSTOMERS_RECONNECTI ON_ATTEMPTS	N/A	The maximum number of attempts to reconnect to the database to query a specific customer that does not exist in shared memory. If the database connection cannot be renewed after this number of attempts, the transaction is rejected.
APR	config	NUM_OF_RECONNECTION_ATTEMPT S	3	The maximum number of reconnection attempts to the current database. When this number is exceeded, the connection attempts are switched to another database. If n < 0, reconnection attempts are made <i>only</i> to the current database (that has just failed). If n = 0, the default value is used (currently, 3). If n > 0, the value is taken from the \${RECONNECTATT} environment variable (currently, 3).
APR	config	NUM_OF_RESOURCE_RECONNECTIO N_ATTEMPTS	N/A	The maximum number of attempts to reconnect to the database to query a specific resource that does not exist in shared memory. If the database connection cannot be renewed after this number of attempts, the transaction is rejected. For online systems, in which performance is very important, the recommended value of this parameter is '1'.
APR	config	NUMBER_OF_CONNECTIONS_PER_C YCLE	\${NUMBER_OF _CONNECTION S_PER_CYCLE}	The number of connections to the Usage database for each cycle.
APR	config	PM_ADD REPL ACK TIMEOUT	5 sec	For planned maintenance (adding a replication). The timeout for receiving an acknowledgement (ACK) from the secondary Event Server.
APR	config	PM_ADD REPL SEND LOW MARK	10	For planned maintenance (adding a replication). A parameter of the Event Producing algorithm. According to this algorithm, events are produced and placed on a queue in iterations to prevent queue overload. The next portion is produced when the number of events in queue is less than the value of this parameter.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	config	PM_ADD_REPL_SEND_QUOTA	20	For planned maintenance (adding a replication). The number of events produced in one iteration by the Event Producing algorithm.
APR	config	PM_COPY_GROUP_MAX resend_TRIES	16	For planned maintenance (copying a group or a replication group). The maximum number of requests for copying a previously uncopied customer that the secondary Event Server can send to the master Event Server.
APR	config	PM_COPY_GROUP_SEND_LOW_MARK	Recommended value: 500	For planned maintenance (copying a group or a replication group). A parameter of the Event Producing algorithm. According to this algorithm, events are produced and placed on a queue in iterations to prevent queue overload. The next portion of Copy events is produced when the number of events in queue is less than the value of this parameter.
APR	config	PM_COPY_GROUP_SEND_QUOTA	Recommended value: 1000	For planned maintenance (copying a group or a replication group). The number of Copy events produced in one iteration by the Event Producing algorithm.
APR	config	PM_PERSIST_GROUP_SEND_LOW_MARK	100	For planned maintenance (copying a group and shadow recovery). A parameter of the Event Producing algorithm. According to this algorithm, events are produced and placed on a queue in iterations to prevent queue overload. The next portion of Persisting Trigger events is produced when the number of events in the queue is less than the value of this parameter.
APR	config	PM_PERSIST_GROUP_SEND_QUOTA	200	For planned maintenance (copying a group and shadow recovery). The number of Persisting Trigger events produced in one iteration by the Event Producing algorithm.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	config	PM_RECOVER_REPLICATION_SEND_LOW_MARK	50	For replication recovery. A parameter of the Event Producing algorithm. According to this algorithm, events are produced and placed on a queue in iterations to prevent queue overload. The next portion of unpersisted events is produced when the number of events in queue is less than the value of this parameter.
APR	config	PM_RECOVER_REPLICATION_SEND_QUOTA	200	For replication recovery. The number of events produced in one iteration by the Event Producing algorithm.
APR	config	PRELOAD_EVENTS_START_TIMES_LAZY_ENABLED	N	<p>For the Duplicate Check mechanism. Enables or disables preloading the existing event start times in lazy mode. Event start times are preloaded from the database into the initial array of event start times that is created in the shared memory for each subscriber.</p> <p>When this parameter is set to 'Y', the array of event start times of the subscriber is created in shared memory and loaded with start times from the database when the first event of the subscriber is rated.</p> <p>The Event Server uses this array of event start times for performing the event duplicate check efficiently, without accessing the database when possible.</p> <p>Preloading the existing event start times into the array enables performing a faster duplicate check for events whose start time is earlier than the time when the shared memory was created.</p> <p>This functionality is especially useful when a new (empty) shared memory is created.</p>
APR	config	PRIMARY_SESSION_EXPIRATION_PERIOD_IN_SEC	3600	The expiration time of the primary session in seconds (the time that must elapse since the last UPDATE event for the primary session until the primary session is expired).
APR	config	PROCESS_TYPE_UP_IN_PASSIVE_SITE	DB2E,ES,GAT2E,UQ_SERVER	The list of process types that can run on the standby site when Geo Redundancy support is enabled.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	config	PROTOCOL_FILES_LOCATION	<ul style="list-style-type: none"> ■ DB ■ A path in the file system 	The location of protocol files. For more information, see the “Environment Variables” section in this chapter.
APR	config	PROTOCOL_TRACING_MODE	false	Indicates whether the Protocol Trace XML file must be created.
APR	config	PW_DD_LIB_NAME	mpr_dd	The name of the library that contains the data layers generated by the Implementation Compiler for the Persistence Writer.
APR	config	PW_HIGH_WATERMARK	60	<p>The percentage of the Persistence Writer queue that must be full for the Event Server to start working with the alternate database. This parameter is used together with the DB_FAIL_THRESHOLD parameter.</p> <p>If the PW_HIGH_WATERMARK parameter is missing or if its value is set to a number greater than 60, the default value of 60 is used.</p>
APR	config	RADAR_ENABLED	\${RADAR_ENA BLED}	Indicates whether Pre-Balance reporting is enabled. The value of this parameter is an environment variable that must be defined in runtime environments. Its default value is ‘N’.
APR	config	RB_DD_LIB_NAME	mpr_dd	The name of the library that contains the data layers generated by the Implementation Compiler for the Event Processing Module.
APR	config	REJECTED_CUSTOMER_ID	999999999	The customer ID that is reserved for rejected events. This parameter is only used if REJECTED_CUSTOMER_ID_RANGE is not defined.
APR	config	REJECTED_CUSTOMER_ID_RANGE	1024	The range of rejected customer IDs within the Guiding to Customer Module. This parameter is used internally by the Event Server to control the distribution of rejected customers across multiple servers.
APR	config	REJECTED_CYCLE_CODE	80	The rejected cycle code.
APR	config	REJECTED_RESOURCE_TYPE	9	The resource type that is reserved for rejected events.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	config	REJECTED_RESOURCE_VALUE	9999	The resource value that is reserved for rejected events.
APR	config	REJECTED_SUBSCRIBER_ID	999999999	The subscriber ID that is reserved for rejected events.
APR	config	REJECTS_INSERT_BUFFER_SIZE	1000	For the Persistence Writer Function. The size of the OCI Oracle buffer for keeping rejected events.
APR	config	REMOVAL_GRACE_PERIOD_IN_SECS	15	<p>The time that must elapse from the expiration of the primary session until the session is deleted from memory.</p> <p>It is recommended that this parameter be set to a value that is 2-3 seconds longer than the duration of an average session.</p>
APR	config	ROLLBACK_FULL_TRANSACTION	\${ROLLBACK_FULL_TRANSACTION}	<p>Determines the behavior of the Persistent Writer if it fails to insert records into the database:</p> <ul style="list-style-type: none"> ■ <i>Y</i> – The entire transaction (event, accumulators, notification records, and dispatching records) is rolled back from the database, and all the records of the transaction (including the ones that did not fail) are inserted into the APE1_ERR_ENTITIES_DETAILS table. ■ <i>N</i> – Only the problematic record of a transaction is rolled back from the database and inserted into the APE1_ERR_ENTITIES_DETAILS table. The Persistence Writer attempts to insert the remaining records into the database. <p>If the remaining records cannot be inserted into the database, the event transaction is treated as if ROLLBACK_FULL_TRANSACTION had the <i>Y</i> value.</p>
APR	config	SEC_WAIT_AFTER_ES_RECOVERY_STARTED	180	In recovery from an active-shadow or host failover, the period (in seconds) after all the recovery messages have been sent until the Event Server becomes available for rerate.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	config	SHARED_ACCUM_SESSIONS_HASH_SIZE	32	The size of the hash table of sessions sharing the accumulator (the number of elements in the hash index array).
APR	config	SHM_POOL_INFILATION_SET	16:50268;32:1058 620:48:97924;64: 719520;80:1901;9 6:206699;112:248 859;128:50060;14 4:165826;160:257 249;176:50005;19 2:105367;208:115 7;240:50006;256: 71767;288:20339; 320:41465;352:97 229;384:120142;4 16:50175;448:611 70:480:2549;512: 11798;576:28464; 640:11511;704:14 5505;768:52830;8 32:16668;896:359 65;960:41763;128 0:83248;262160:2 56	The amount of memory that must be pre-allocated in every Event Server, even if it is started as a replication Event Server. This amount constitutes a fraction of the total shared memory. The value of the parameter is a string of <size>:<number> pairs, such as ‘10:2;16:34;32:13’. The first number in a pair denotes the size of an allocated block, and the second number indicates the number of such blocks. Thus, the ‘10:2’ pair means two blocks of 10 bytes each. The out-of-the-box value of this parameter must be changed during the UAT or performance tests on the full population of a project.
APR	config	SIC_ALLOW_PREFERRED_CR_ROUTING	Y	Indicates whether the Event Server functioning as the Network Interface Module that was on the session must be selected as the first path for server-initiated command (SIC) messages. This parameter is not included out of the box. If it is not added, the default value is used.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	config	SIC_AVP_DESCRIPTOR_XML	N/A	The location of the XML configuration file that defines the additional attribute-value pairs (AVPs) to be taken from the INIT session event to the generated server-initiated command (SIC) messages. This is an implementation item that is not required by the core. If this feature is not used, this line must be omitted.
APR	config	SIC_DELTA_TIME_BETWEEN_REPORT_DROP_TO_AVM_ALERT_IN_SEC	50	<p>The interval (in seconds) between sending alerts to the Availability Manager about dropping messages for a specific network element.</p> <p>This parameter is not included out of the box. If it is not added, the default value is used.</p>
APR	config	SIC_ENABLED	Y	Indicates whether server-initiated commands are enabled.
APR	config	SIC_ENABLING_PROTOCOL_ID	10	The ID of the protocol that enables server-initiated commands.
APR	config	SIC_NUM_OF_MAX_LOOPS	5	<p>The maximum number of loops per server-initiated command (SIC) message. A loop in this context is an unsuccessful attempt to send a SIC message to the target network element via all available Event Servers functioning as Network Interface Modules. After this number of loops is completed, the SIC message is dropped.</p> <p>This parameter is not included out of the box. If it is not added, the default value is used.</p>
APR	config	SIC_NUM_OF_MAX_SENDS	100	<p>The maximum number of resends for a SIC message.</p> <p>This parameter is not included out of the box. If it is not added, the default value is used.</p>
APR	config	SIC_RESEND_TIMEOUT_IN_SECS	2	The interval with which server-initiated commands are resent, in seconds.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	config	SIC_REPLICATION_EXPIRATION_RESETS_FACTOR	3	<p>The factor by which the maximum number of resends and the interval between them must be multiplied to calculate the expiration time of messages.</p> <p>This parameter is not included out of the box. If it is not added, the default value is used.</p>
APR	config	SIC_UPDATE_AVP_ON_UPDATE	N	Indicates whether every Update event must overwrite the attribute-value pairs (AVPs) that were saved on the primary session by the previous Init or Update event.
APR	config	SLEEP_BETWEEN_CUSTOMERS_CONNECTION_ATTEMPTS	N/A	The sleep time between two attempts to reconnect to the database to query a specific customer that does not exist in shared memory.
APR	config	SLEEP_BETWEEN_RESOURCE_CONNECTION_ATTEMPTS	N/A	<p>The sleep time between two attempts to reconnect to the database to query a specific resource that does not exist in shared memory.</p> <p>For online systems, in which performance is very important, the recommended value of this parameter is '0'.</p>
APR	config	SLEEP_IN_BETWEEN_CONNECTION	1 second	The time between connections to the ports (in nanoseconds). If the value of this parameter is less than 1 second, the default value of 1 second is used.
APR	config	SLEEP_TIME_BETWEEN_RECONNECTION_ATTEMPTS	5	<p>The sleep time (in seconds) between two attempts to reconnect to the database.</p> <p>If n <= 0, the default value is used (currently, 5 seconds). If n > 0, the value is taken from the \${SLEEPBETCONN} environment variable (currently, 30 seconds).</p>
APR	config	STALE_SESSIONS_IS_EVENT_TIME_GMT	\${APR_STALE_SESSIONS_IS_EVENT_TIME_GMT}	Indicates whether the time stamp of Sy-like events is in GMT. If this parameter is set to 'N', the logic of Sy-like session removal does not perform conversion to local time.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	config	STALE_SESSIONS_PROTOCOL_IDS	\${APR_STALE_SESSIONS_PROTOCOL_IDS}	The maximum number of the latest updated Sy-like sessions per subscriber to be retained in the memory of the Event Server when stale sessions are removed.
APR	config	STALE_SESSIONS_MAX_SESSIONS_PER_SUBSCRIBER	\${APR_STALE_SESSIONS_MAX_SESSIONS_PER_SUBSCRIBER}	A comma-separated list of Sy-like protocol IDs for which the removal of stale sessions from memory is enabled.
APR	config	SUBSCRIBER_DB_FETCHER_SIZE	100	The size of the buffer that is used when fetching the subscriber data from the database.
APR	config	SY_ENABLED	Y	Indicates whether the Sy reference point is enabled. If the Sy reference point is not used, this parameter must be set to ‘N’ to block Sy messages. If this parameter is not defined, the default value is ‘N’.
APR	config	SYNCH_GRACEFULL_SHUTDOWN	Y	Defines the behavior of the process upon receiving the Graceful Shutdown command from the Availability Manager: <ul style="list-style-type: none"> ■ <i>Y</i> – The process sends an ACK message to the Availability Manager when the graceful shutdown is complete. ■ <i>N</i> – The process sends an ACK message to the Availability Manager when it receives the command and starts to shut down.
APR	config	SYNCH_WITH_SYS_TIME_PERIOD	5 min	For performance reasons, a special Time Manager thread determines the time in the Event Server. This parameter defines the period (in seconds) after which the Time Manager synchronizes the time with the system time.
APR	config	TCCLIENTS_HANDSHAKE_RCV_TIMEOUT	\${ES_TCCLIENTS_HANDSHAKE_RCV_TIMEOUT}	The timeout for receiving the message in the handshake between the Event Server and Turbo Charging clients.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	config	TCCLIENTS_HANDSHAKE_SND_TIMEOUT	\${ES_TCCLien_TS_HANDSHAKE_SND_TIMEOUT}	The timeout for sending the message in the handshake between the Event Server and Turbo Charging clients.
APR	config	UDP_MULTICAST_TTL	1	The time-to-live (TTL) for UDP multicast networking, which must be equal to or greater than the number of subnets.
APR	config	UDP_RECV_BUF_SIZE	\${UDP_RECV_BUFFER_SIZE}	The size of the UDP buffer that is used for receiving messages.
APR	config	UDP_SEND_BUF_SIZE	\${UDP_SEND_BUFFER_SIZE}	The size of the UDP buffer that is used for sending messages.
APR	config	UPDATE_EVENT_FOR_DELETE_BUFFER_SIZE	1000	For the Persistence Writer Function. The size of the OCI Oracle buffer for keeping special records with data for marking rated events as deleted (setting the EVENT_STATE column in the APE1_RATED_EVENT table to 'X').
APR	config	USE_LOGICAL_DATE	N	Indicates whether the system uses the logical date from the LOGICAL_DATE table.
APR	config	USE_VISITED_DB	N	Indicates whether in the case of a database failure, the Event Server must switch to the alternate database or continue trying to reconnect to the home database until it succeeds.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	config	VIRTUAL_OR_SHARED_MEMORY_USED	SHARED	<p>Indicates whether the Event Server uses virtual or shared memory.</p> <ul style="list-style-type: none"> ▪ <i>SHARED</i> – For production ▪ <i>VIRTUAL</i> – For debugging and investigation only <p>This parameter is used in conjunction with the MEMORY_SIZE_IN_BYTES parameter. To configure the Event Server to use unlimited virtual memory (when memory size is not pre-allocated, and the Event Server consumes as much memory as it needs), set these parameters as follows:</p> <ul style="list-style-type: none"> ▪ On Windows: <ul style="list-style-type: none"> • VIRTUAL_OR_SHARED_MEMORY_USED to ‘VIRTUAL’ • MEMORY_SIZE_IN_BYTES to ‘4294967295’ ▪ On Unix: <ul style="list-style-type: none"> • VIRTUAL_OR_SHARED_MEMORY_USED to ‘VIRTUAL’ • MEMORY_SIZE_IN_BYTES to ‘18446744073709551615’ <p>This mode is recommended when working on Windows, or when multiple users access the same host (for example, in testing).</p>
APR	config	WORKING_PERCENTAGE_PARAM	100	For configurations with multiple Event Servers. Defines the percentage of Event Server instances that must be up so that a particular Event Server instance that tries to connect to them can be activated.
APR	ELA	COLLECT_LOOP_FACTOR	100	Determines the interval at which the Event Server sends the aggregated event-level auditing data to the ELA Collector. This interval equals COLLECT_LOOP_FACTOR * 50 (ms, as specified in the .ccf file).

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	ELA	ELA_MAX_RESEND_RETRIES_BEFORE_DROP	32	The maximum number of attempts to send a message to the ELA Collector. If an acknowledgement is not received from the ELA Collector after this number of attempts, the message is dropped, even when the queue is not full.
APR	ELA	ENABLE	\${APR_ELA_ENABLE}	Enables or disables the Event-Level Auditing mechanism.
APR	ELA	ES_ELA_RECOVERY_LEVEL	\${APR_ES_ELA_RECOVERY_LEVEL}	The recovery level for event-level auditing data in the Event Server. Values are: <ul style="list-style-type: none"> ■ <i>DISABLE</i> ■ <i>FULL_PERSISTENCE</i> – Every record is written to the database (a high recovery level). ■ <i>OVERFLOW_PERSISTENCE</i> – Records are written to the database only when the message queue for event-level auditing data is full.
APR	ELA	ES_ELA_QUEUE_LOW_WATERMARK	\${APR_ES_ELA_QUEUE_LOW_WATERMARK}	The percentage of the size of the ELA message queue that defines when the ELA Recovery Catch-up flow is activated. If the message queue was full previously and then its size decreased below this value, the Catch-up flow starts.
APR	ELA	ES_ELA_QUEUE_HIGH_WATERMARK	\${APR_ES_ELA_QUEUE_HIGH_WATERMARK}	The percentage of the size of the ELA message queue that defines when the ELA Recovery Catch-up flow is terminated. If the size of the message queue exceeds this value, the Catch-up flow stops.
APR	ELA	ES_ELA_RECOVERY_CLEANUP_TIME_MIN	\${APR_ES_ELA_RECOVERY_CLEANUP_TIME_MIN}	The intervals (in minutes) at which the ELA Recovery Clean-up flow must run. The Clean-up flow checks whether there are records in the APR1_ES_ELA_RECOVERY that were sent to the ELA Collector but did not receive a response. These records are marked as out of queue, and when the Catch-up flow is activated, they are resent.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	ELA	ES_ELA_RECOVERY_PARTITIONS_TO_KEEP	\${APR_ES_ELA_RECOVERY_PARTITIONS_TO_KEEP}	The number of partitions in the APR1_ES_ELA_RECOVERY table to keep when partitions are truncated.
APR	ELA	ES_ELA_RECOVERY_TRUNCATE_TIME_MIN	\${APR_ES_ELA_RECOVERY_TRUNCATE_TIME_MIN}	The intervals (in minutes) at which the ELA Recovery Truncate flow must run.
APR	ELA	F2E_LATENCY_BANDS_MSEC	0,5,25,100,200	The latency bands for reporting the latency of offline events measured at the Outgoing Events (CROut) reporting point. The system counts the number of events that falls within the time range of each band. The latency bands must be configured depending on the SLA per source type.
APR	ELA	LATENCY_BANDS_MSEC	0,5,25,100,200	The latency bands for reporting the latency of online events measured at the Outgoing Events (CROut) reporting point.  Note: In an integrated environment with Amdocs Service Platform, latency is measured at the OCFCOut reporting point in Amdocs Service Platform. The system counts the number of events that falls within the time range of each band. The latency bands must be configured depending on the SLA per source type.
APR	ELA	LOGGER_MAX_RECORDS_COUNT	540000	The maximum number of records in the queue that holds raw, unaggregated data (a record per event).
APR	ELA	MAX_RESEND_RETRIES	16	The maximum number of attempts to resend event-level auditing messages that did not get a response.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	ELA	MESSAGE_QUEUE_MAX_SIZE	10000	The size of the message queue for Event-Level Auditing messages (in bytes). This size determines how much event-level auditing data can be kept in the queue if the ELA Collector is down before the data is lost. Out of the box, the message queue supports 30 minutes of data.
APR	ELA	RESEND_TIMEOUT_EXTENSION_SEC	10	Extra safety time that is added to the specified timeout for communication between the Event Server and the ELA Collector.
APR	ELA	TIME_TO_COMMIT_SEC	5	The interval at which the ELA Collector commits data to the database.
APR	ELA	TOKEN_DURATION_MIN	5	The duration of a token (in minutes).
APR	ELA_SERVER	SEND_ACK_TIMEOUT_SEC	10	The timeout of the ELA Collector for sending an acknowledgement.
APR	EVENT_LOG_CR	ENABLE	\${APR_ENABLE_EVENT_LOG}	Indicates whether the event log at the Outgoing Network Interface reporting point is enabled at Event Server start-up. It can then be stopped and re-activated using an admin command.
APR	EVENT_LOG_CR	FILE_EXT	.log	The extension of the event log files produced at the Outgoing Network Interface reporting point.
APR	EVENT_LOG_CR	FILE_NAME_SUFFIX	EVENT_LOG_CR	A component of the event log file name that identifies the reporting point.
APR	EVENT_LOG_CR	FILE_PATH	\${ABP_APRLOG}/trace	The path to the event log files produced at the Outgoing Network Interface reporting point.
APR	EVENT_LOG_CR	FLUSH_SIZE	\${APR_EVENT_LOG_FLUSH_SIZE}	The size (in bytes) up to which an event log is accumulated before it is written to a file. Values are: <ul style="list-style-type: none">■ 0 – No buffering. Records are flushed to disk immediately.■ -1 – Default buffering of the operating system.■ >0 – The actual buffer size.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	EVENT_LOG_CR	MAX_FILE_SIZE	<code> \${APR_EVENT_LOG_MAX_FILE_SIZE}</code>	The maximum size (in megabytes) of an event log file produced at the Outgoing Network Interface reporting point.
APR	EVENT_LOG_CRIN	ENABLE	<code> \${APR_ENABLE_EVENT_CRIN_EVENT_LOG}</code>	Indicates whether the event log at the Incoming Network Interface reporting point is enabled at Event Server start-up. It can then be stopped and re-activated using an admin command.
APR	EVENT_LOG_CRIN	FILE_EXT	.log	The extension of the event log files produced at the Incoming Network Interface reporting point.
APR	EVENT_LOG_CRIN	FILE_NAME_SUFFIX	EVENT_LOG_CRIN	A component of the event log file name that identifies the reporting point.
APR	EVENT_LOG_CRIN	FILE_PATH	<code> \${ABP_APRLOG}/trace</code>	The path to the event log files produced at the Incoming Network Interface reporting point.
APR	EVENT_LOG_CRIN	FLUSH_SIZE	<code> \${APR_EVENT_LOG_FLUSH_SIZE}</code>	The size (in bytes) up to which an event log is accumulated before it is written to a file. Values are: <ul style="list-style-type: none">■ 0 – No buffering. Records are flushed to disk immediately.■ -I – Default buffering of the operating system.■ >0 – The actual buffer size.
APR	EVENT_LOG_CRIN	MAX_FILE_SIZE	<code> \${APR_EVENT_LOG_MAX_FILE_SIZE}</code>	The maximum size (in megabytes) of an event log file produced at the Incoming Network Interface reporting point.
APR	EVENT_LOG_FR	ENABLE	<code> \${APR_ENABLE_EVENT_FR_EVENT_LOG}</code>	Indicates whether the event log at the Guiding to Customer reporting point is enabled at Event Server start-up. It can then be stopped and re-activated using an admin command.
APR	EVENT_LOG_FR	FILE_EXT	.log	The extension of the event log files produced at the Guiding to Customer reporting point.
APR	EVENT_LOG_FR	FILE_NAME_SUFFIX	EVENT_LOG_FR	A component of the event log file name that identifies the reporting point.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	EVENT_LOG_FR	FILE_PATH	\${ABP_APRLOG}/trace	The path to the event log files produced at the Guiding to Customer reporting point.
APR	EVENT_LOG_FR	FLUSH_SIZE	\${APREVENTLOG_FLUSH_SIZE}	The size (in bytes) up to which an event log is accumulated before it is written to a file. Values are: <ul style="list-style-type: none">■ 0 – No buffering. Records are flushed to disk immediately.■ -1 – Default buffering of the operating system.■ >0 – The actual buffer size.
APR	EVENT_LOG_FR	MAX_FILE_SIZE	\${APREVENTLOG_MAXFILE_SIZE}	The maximum size (in megabytes) of an event log file produced at the Guiding to Customer reporting point.
APR	EVENT_LOG_RB	ENABLE	\${APRENABLE_RB_EVENTLOG}	Indicates whether the event log at the Event Processing reporting point is enabled at Event Server start-up. It can then be stopped and re-activated using an admin command.
APR	EVENT_LOG_RB	FILE_EXT	.log	The extension of the event log files produced at the Event Processing reporting point.
APR	EVENT_LOG_RB	FILE_NAME_SUFFIX	EVENT_LOG_RB	A component of the event log file name that identifies the reporting point.
APR	EVENT_LOG_RB	FILE_PATH	\${ABP_APRLOG}/trace	The path to the event log files produced at the Event Processing reporting point.
APR	EVENT_LOG_RB	FLUSH_SIZE	\${APREVENTLOG_FLUSH_SIZE}	The size (in bytes) up to which an event log is accumulated before it is written to a file. Values are: <ul style="list-style-type: none">■ 0 – No buffering. Records are flushed to disk immediately.■ -1 – Default buffering of the operating system.■ >0 – The actual buffer size.
APR	EVENT_LOG_RB	MAX_FILE_SIZE	\${APREVENTLOG_MAXFILE_SIZE}	The maximum size (in megabytes) of an event log file produced at the Event Processing reporting point.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	EVENT_LOG_STICKY_QUEUE	ELEMENTS	3	The maximum number of different target files that the event log sticky queue supports.
APR	EVENT_LOG_STICKY_QUEUE	PRIORITY	0	The priority of the threads when they perform an activity on the event log sticky queue.
APR	EVENT_LOG_STICKY_QUEUE	SLEEP_MILI_INTERVAL	0	<p>The locking strategy for the critical sections of the event log sticky queue if there are no events between the fetch tries of the dequeue function. Possible values are:</p> <ul style="list-style-type: none"> ■ <i>0 (default)</i> – Use the tryacquire and yield functions in the case of a retry. ■ <i>999 milliseconds and up</i> – Use the acquire function. ■ <i>Any other value</i> – Use the tryacquire function and sleep for the specified period of time (in milliseconds) in the case of a retry.
APR	ES_RR_DATA_PARAMS	MAX_CATCHUP_EVENTS	300	The maximum number of ongoing events per rerated customer that can be processed during rerate.
APR	LF_FIFO_QUEUE	LF_FIFO_ELEMENTS	30000	The maximum number of pending events that a queue based on the FIFO (first in, first out) principle can hold.
APR	LF_FIFO_QUEUE	LF_FIFO_PRIORITY	0	The priority of the threads when they perform an activity on a FIFO-based queue.
APR	LF_FIFO_QUEUE	LF_FIFO_SLEEP_MILI_INTERVAL	0	<p>The locking strategy for the critical sections of the FIFO-based queue if there are no events between the fetch tries of the dequeue function. Possible values are:</p> <ul style="list-style-type: none"> ■ <i>0 (default)</i> – Use the tryacquire and yield functions in the case of a retry. ■ <i>999 milliseconds and up</i> – Use the acquire function. ■ <i>Any other value</i> – Use the tryacquire function and sleep for the specified period of time (in milliseconds) in the case of a retry.
APR	LF_STICKY_QUEUE	LF_STICKY_ELEMENTS	30000	The maximum number of pending events that the Rater queue can hold.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	LF_STICKY_QUEUE	LF_STICKY_PRIORITY	0	The priority of the threads when they perform an activity on the sticky queue.
APR	LF_STICKY_QUEUE	LF_STICKY_SLEEP_MILI_INTERVAL	0	The locking strategy for the critical sections of the sticky queue if there are no events between the fetch tries of the dequeue function. Possible values are: <ul style="list-style-type: none"> ■ <i>0 (default)</i> – Use the tryacquire and yield functions in the case of a retry. ■ <i>999 milliseconds and up</i> – Use the acquire function. ■ <i>Any other value</i> – Use the tryacquire function and sleep for the specified period of time (in milliseconds) in the case of a retry.
APR	OMC_SDK	MAX_NUMBER_OF_RECORDS	65536	The maximum number of records in the memory-mapped file that is created at process initialization. The operational measurement data that the process generates is written to this file. The file has a fixed size of MAX_NUM_OF_RECORDS*record size (832, hard-coded) + MMF header size (856, hard-coded). By default, the maximum number of records in a file is 65536, which yields the file size of 54526808. To change the file size, the MAX_NUMBER_OF_RECORDS parameter must be added to the APR1_CONF_SECTION_PARAM table.
APR	OMC_SDK	OM_ENABLED	Y	Indicates whether the process generates operational measurement data. By default, the Operational Measurement mechanism is enabled. To disable the mechanism, the OM_ENABLED parameter must be added to the APR1_CONF_SECTION_PARAM table.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	OMC_SDK	OM_SHM_MON_TIMEPERIOD_SECONDS	1	<p>The frequency of scanning the shared memory for the purpose of recording shared memory operational measurements (counters).</p> <p>By default, the APR1_CONF_SECTION_PARAM table does not include this record. To change the default period (for example, to investigate a problem or determine the impact of a value on performance), the OM_SHM_MON_TIMEPERIOD_SECONDS parameter must be added to the table.</p>
APR	OMC_SDK	TIME_INTERVAL	10	The time interval for time-based monitoring objects (MOBs).
APR	OUTCOLLECT	IMSI_RESOURCE_TYPE	3	The IMSI resource type (outcollect check is performed for IMSI resources only).
APR	OUTCOLLECT	NATIVE_IMSI_LIST	12345	A comma-delimited list of provider IDs.
APR	OUTCOLLECT	OUTCOLLECT_RESOURCE_TYPE	111	The outcollect resource type (relevant only if the RESOURCE_TYPE_OVERWRITE field is 'Y').
APR	OUTCOLLECT	PREFIX_SIZE	5	The size of the resource value prefix that specifies the provider ID.
APR	OUTCOLLECT	RESOURCE_TYPE_OVERWRITE	N	Indicates whether the resource type is to be overwritten with the OUTCOLLECT_RESOURCE_TYPE value for outcollect events.
APR	PW_LF_STICKY_QUEUE	PW_LF_NO_COMMIT_TIMEOUT_DELTA	\${APR_PW_LF_NO_COMMIT_TIMEOUT_DELTA}	The timeout from the last commit, in milliseconds, after which the queue sends an expiration message to the Persistence Writer. This parameter is not mandatory; if it has not been defined, the default hard-coded value is used. If the commit timeout interval is greater than one day, the timeout mechanism is disabled.
APR	PW_LF_STICKY_QUEUE	PW_LF_STICKY_ELEMENTS	30000	The maximum number of customer groups processed by the Persistence Writer, multiplied by2.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	PW_LF_STICKY_QUEUE	PW_LF_STICKY_MAX_CHILD_MESSAGES	0	The maximum number of pending messages for a specific customer group, only if the group is suspended. When this limit is reached, the application may drop events.
APR	PW_LF_STICKY_QUEUE	PW_LF_STICKY_MAX_PARENT_MESSAGES	0	The maximum number of pending messages on a database connection. When this limit is reached, the application may drop events.
APR	PW_LF_STICKY_QUEUE	PW_LF_STICKY_MAX_TOTAL_MESSAGES	0	The maximum number of pending messages on the queue. When this limit is reached, the application may drop messages.
APR	PW_LF_STICKY_QUEUE	PW_LF_STICKY_PRIORITY	0	The priority of the threads when they perform an activity on the Persistence Writer queue.  Note: Currently, this parameter is not used.
APR	PW_LF_STICKY_QUEUE	PW_LF_STICKY_SLEEP_MILI_INTERVAL	0	The locking strategy for the critical sections of the Persistence Writer queue if there are no events between the fetch tries of the dequeue function. Possible values are: <ul style="list-style-type: none">■ 0 (default) – Use the tryacquire and yield functions in the case of a retry.■ 999 milliseconds and up – Use the acquire function.■ Any other value – Use the tryacquire function and sleep for the specified period of time (in milliseconds) in the case of a retry.
APR	RR_PARAMS	MAX_SESSION_COUNT	1	The maximum number of rerate sessions in the Event Server during the End-of-Cycle Rerate.
APR	SQL_TRACE	ACTIVATE	N	Specifies whether SQL tracing is activated at process start-up. This parameter can be redefined at the process type or process instance level.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	SQL_TRACE	ENABLE	Y	<p>Specifies whether the SQL Trace Manager must be initialized for the process. If the SQL Trace Manager is not initialized at start-up, the process cannot receive an admin command to activate SQL tracing at runtime.</p> <p>This parameter can be redefined at the process type or process instance level.</p>
APR	SQL_TRACE	IS_DURATION_REQUIRED	\${IS_DURATION_REQUIRED}	<p>Specifies whether SQL tracing must be activated for a limited amount of time only (for example, in production). If this is the case:</p> <ul style="list-style-type: none"> ▪ If SQL tracing is activated at process start-up, it is limited to 60 minutes. ▪ The command that activates the tracing on the fly must specify the required duration up to the allowed maximum (60 minutes). <p>This parameter can be redefined at the process type or process instance level.</p>
APR	TLS.Cleanup.SessionGuiding	DAYS_TO_KEEP_SESSION_GUIDING_DATA	100	The number of days to keep in the APR1_SESSION_GUIDING table.
APR	TLS.Cleanup.SharedAllowance	NUM_OF_MONTH_BACK	4	The number of months since expiration for which entries in the APE1_SA_STATUS table must be deleted.
APR/APE	config	DD_LIB_NAME	mpr_dd	The name of Dictionary library of the Implementation Compiler that contains all information relevant for the interface between an application and the generated code.
AVM	TLS.Cleanup.Avm1Alerts	DaysToSaveAlerts	30	The number of days to keep in the AVM1_ALERTS table.
AVM	TLS.Cleanup.OmCountersHistory	DaysToSaveCounters	30	The number of days to keep in the GN1_OM_COUNTERS_HISTORY table.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
EOC_SE RVER	config	EXTERNAL_UDP_NW_ENABLED	N	<p>Indicates whether the External UDP Manager is enabled ('Y') or disabled ('N'). Sending Diameter events over a UDP connection is allowed only if this parameter is set to 'Y'.</p> <p>If there no special configuration that enables sending online events via a UDP port without a response, this parameter must be set to 'N'.</p>
EOC_SE RVER	config	FORCE_GAT_SCOPE	A	<p>Enables a specific class to receive dummy responsibility in order to load GATs. Values are:</p> <ul style="list-style-type: none"> ■ C – Network Interface responsibility ■ G – Guiding to Customer responsibility ■ R – Event Processing responsibility ■ A – All responsibilities
EOC_SE RVER18 0	config	VIRTUAL_OR_SHARED_MEMORY_U SED	VIRTUAL	Indicates whether the Event Server uses virtual or shared memory in the End-of-Cycle map.
ES	config	CROUT_BUSYRESPONCE_ENABLE	FALSE	Indicates whether the Network Interface Function must send a 'busy' response to the network ('TRUE') or to drop the 'busy' response ('FALSE') if an Event Server cannot send the event to the Event Server performing the next processing stage (rating or replication). In both cases, the event is reported to Event-Level Auditing. However, dropping the response causes the network element to resend the event.
ES	config	DISPATCHER_ENABLED	Y	Indicates whether the Dispatcher process is enabled for this Event Server.
ES	config	MAX_OPEN_SESSIONS	200000	The maximum number of open sessions in the Event Server.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ES	config	MEMORY_SIZE_IN_BYTES	\${ES_MEMORY_SIZE_IN_BYTES}	<p>The requested size of the shared memory. This parameter is used in conjunction with the VIRTUAL_OR_SHARED_MEMORY_USED parameter. To configure the Event Server to use unlimited virtual memory (when memory size is not pre-allocated, and the Event Server consumes as much memory as it needs), set these parameters as follows:</p> <ul style="list-style-type: none"> ■ On Windows: <ul style="list-style-type: none"> • VIRTUAL_OR_SHARED_MEMORY_USED to ‘VIRTUAL’ • MEMORY_SIZE_IN_BYTES to ‘4294967295’ ■ On Unix: <ul style="list-style-type: none"> • VIRTUAL_OR_SHARED_MEMORY_USED to ‘VIRTUAL’ • MEMORY_SIZE_IN_BYTES to ‘18446744073709551615’ <p>This mode is recommended when working on Windows, or when multiple users access the same host (for example, in testing).</p>
ES	config	NUM_OF_CUG_RECONNECT_ATTEMPTS	\${NUM_OF_CUG_RECONNECT_ATTEMPTS}	The number of attempts to reconnect to the resource database.
ES	config	OCFE_PROTOCOL_ID	\${APR_OCFE_PROTOCOL_ID}	The ID of the Amdocs Service Platform (OCFE) protocol.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ES	config	PERSIST_ALL_SESSIONS_NOW_BLOCK_IF_BUSY_MILI	100	<p>This parameter is used by the Session Expiration thread when processing the PERSIST_ALL_SESSIONS_NOW command that is sent from the Availability Manager to the Event Server. When the Event Server receives the PERSIST_ALL_SESSIONS_NOW command, the Session Expiration thread expires all the sessions that are still active in shared memory and sends a message to Rater thread for each such session.</p> <p>This parameter is used to prevent the Session Expiration thread from overloading the Rater thread with too many sessions to expire.</p> <p>The value of the parameter is the time (in milliseconds) for which the Session Expiration thread must wait before adding a new message to the Rater queue in the following cases:</p> <ul style="list-style-type: none"> ■ If the Persistence Writer queue is busy ■ If the Rater queue already contains the number of messages specified by the PERSIST_ALL_SESSIONS_NOW_RB_MAX_MS_GS_IN_QUEUE parameter

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ES	config	PERSIST_ALL_SESSIONS_NOW_RB_MAX_MSGS_IN_QUEUE	2000	<p>This parameter is used by the Session Expiration thread when processing the PERSIST_ALL_SESSIONS_NOW command that is sent from the Availability Manager to the Event Server. When the Event Server receives the PERSIST_ALL_SESSIONS_NOW command, the Session Expiration thread expires all the sessions that are still active in shared memory and sends a message to Rater thread for each such session.</p> <p>This parameter is used to prevent the Session Expiration thread from overloading the Rater thread with too many sessions to expire.</p> <p>The value of the parameter is the maximum number of messages that the Rater queue can contain while the Session Expiration thread is performing the PERSIST_ALL_SESSIONS_NOW command.</p>
ES	config	PROTOCOLS_ALLOW_UPDATE_WITH_OUT_INIT	N/A	<p>A list of protocols that allow an UPDATE event to open a session without an INIT event arriving first.</p> <p>If an UPDATE event arrives for one of these protocols, and if there is no open session, the event category is changed to INIT, and a session is opened.</p> <p>This feature can be used in Sy-like protocols only.</p>
ES	config	PROTOCOLS_DROP_TERMINATE_WITHOUT_OPEN_SESSION	N/A	<p>A list of protocols for which a TERMINATE event is dropped with an error if there is no open session in memory.</p> <p>This feature can be used in Sy-like protocols only.</p>
ES	config	SESSION_BASED_GUIDING_AGING_IN_MINUTES	600	The period (in minutes) for which to keep the data for session-based guiding in AIMOS. For optimal performance, this number must be a multiple of 4.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ES	config	SESSION_BASED_GUIDING_HASH_SIZE	\${ADJ1_SESSIO N_BASED_GUI DING_HASH_SI ZE}	The number of entries in the hash table that contains data for session-based guiding in AIMOS. This number must be set according to the expected number of Sy sessions.
ES	config	SESSION_BASED_GUIDING_INSERT_BUFFER_SIZE	1000	The maximum number of records in the internal insert and delete buffers. The value must be greater than 1.
ES	config	SESSION_BASED_GUIDING_PERSIST_TIMES_TO_RECONNECT	3	The maximum number of attempts to reconnect when persistence of the data for session-based guiding fails due to connection problems.
ES	config	SESSION_BASED_GUIDING_PROTOCOL_IDS	17	A comma-delimited list of numeric protocol IDs for which session-based guiding is enabled.
ES	config	SESSION_BASED_GUIDING_SELECT_HINT	/* */	The hint for the SQL statement used for extracting session information from the APR1_SESSION_GUIDING table.
ES	config	SESSION_GUIDING_SELECT_AGE_IN_DAYS	0	A filter for records in the APR1_SESSION_GUIDING table. <ul style="list-style-type: none"> ■ <i>0</i> – No filter is applied. ■ <i>1 and above</i> – Records older than the specified number of days are not included in the search.
ES	config	SHM_POOL_INFILATION_LOW_WATER_MARK_THRESHOLD	\${SHM_POOL_INFILATION_LOW_WATER_MARK_THRESHOLD}	The smallest shared memory pool inflation size that can be taken from a file. If the calculated size from the file is lower than this threshold, the value of the SHM_POOL_INFILATION_SET parameter in the database is used instead.
ES	config	SHM_POOL_INFILATION_SET_FILE_PATH	\${APR_SHM_POOL_INFILATION_SET_FILE_PATH}	The path to the file with the shared memory inflation pool sizes, which is created by the ADJ1_AllocatedSharedMemoryBlocks_Sh script during host failover.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ES	config	SHM_POOL_INFILATION_SET_OVERRIDE_FROM_FILE	FALSE	Indicates whether the Event Server must use shared memory inflation pool sizes from the file created by the ADJ1_AllocatedSharedMemoryBlocks_Sh script. If this parameter is set to 'FALSE', the sizes are taken from the SHM_POOL_INFILATION_SET parameter.
ES	config	SLEEP_BTWN_CUG_RECONNECT_ATTEMPTS	\${SLEEP_BTWN_CUG_RECONNECT_ATTEMPTS}	The sleep time between attempts to reconnect to the resource database.
ES	config	TCCLIENTS_BUSY_TIMEOUT	1000000	The time-out (in milliseconds) of the Persistence Writer queue. A queue is full when the max_number_of_element - delta (currently, delta = 100) is less than the number_of_elements_in_que. If the Persistence Writer queue is full, the thread sleeps for 100 ms and then checks whether the queue is still full. If so, it sleeps for another 100 ms and checks again. The thread continues with this loop until the queue is no longer full or the TCCLIENTS_BUSY_TIMEOUT is reached.
ES	ROLL_ACC_RR_OG	EVENT_TYPE	DEFAULT	The type of event that must be generated and sent to the rating flow at the end of Ongoing Rerate. Values are: <ul style="list-style-type: none">■ DEFAULT – Internal Accumulator Maintenance event.■ EXTERNAL – Accumulator Trigger event, which enables the implementation to identify that rerating has been performed and apply additional steps, such as sending a notification to an external system. If this value is set, the value of the ABORT_SESSION_ON_DUPLICATE parameter of the Dispatcher must be set to 'N' to enable generating several records from the Accumulator Trigger event.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ES	ROLL_ACC_RR_EOC	EVENT_TYPE	DEFAULT	<p>The type of event that must be generated and sent to the rating flow at the end of End-of-Cycle Rerate. Values are:</p> <ul style="list-style-type: none"> ▪ <i>DEFAULT</i> – Internal Accumulator Maintenance event. ▪ <i>EXTERNAL</i> – Accumulator Trigger event, which enables the implementation to identify that rerating has been performed and apply additional steps, such as sending a notification to an external system. If this value is set, the value of the ABORT_SESSION_ON_DUPLICATE parameter of the Dispatcher must be set to ‘N’ to enable generating several records from the Accumulator Trigger event.
ROR_SE RVER	config	AVM_ENABLED	N	<p>Indicates whether Availability Manager is used when running the process.</p> <p>If the Event Server is configured to be activated though the Availability Manager, it must be configured to work with shared memory and not virtual memory (see VIRTUAL_OR_SHARED_MEMORY_USED).</p>
ROR_SE RVER	config	DISPATCHER_ENABLED	N	Indicates whether the Dispatcher process is enabled for this read-only Event Server.
ROR_SE RVER	config	EVENT_TRACING_MODE	Y	Indicates whether the Event Trace XML file must be created.
ROR_SE RVER	config	FORCE_GAT_SCOPE	A	<p>Enables a specific class to receive dummy responsibility in order to load GATs. Values are:</p> <ul style="list-style-type: none"> ▪ <i>C</i> – Network Interface responsibility ▪ <i>G</i> – Guiding to Customer responsibility ▪ <i>R</i> – Event Processing responsibility ▪ <i>A</i> – All responsibilities

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ROR_SE_RVER	config	IMMEDIATE_EXIT	Y	Indicates whether upon receiving the shutdown command, the read-only Event Server must shut down without waiting for the rest of the shutdown flow.
ROR_SE_RVER	config	IS_ROR_SERVER	Y	Indicates whether the Event Server is a read-only Event Server.
ROR_SE_RVER	config	VIRTUAL_OR_SHARED_MEMORY_USED	VIRTUAL	Indicates whether the read-only Event Server uses virtual or shared memory.

For each process instance, an entry with the name of the process instance must be created in the corresponding *nnn_CONF_HIERARCHY* table.

Example:

PROCESS_NAME	PROCESS_GROUP
ES100	APE

Any parameter to be overridden at the process instance level must be added to the corresponding *nnn_CONF_SECTION_PARAM* table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
ES100	config	GRACE_PERIOD	2

For more information on parameters and the properties mechanism, see the “Configuration Parameter Mechanism” appendix.

Configuration Files

The Event Server requires a configuration file (.ccf) that is defined via an internal configuration tool.

In general, the .ccf file can contain only a few particular features. For example, one can configure an Event Server that only reads from a switch and performs initial parsing, or an Event Server that only reads from a switch, and performs initial parsing and guiding to customer. Each site can create an architecture that includes several Event Servers, where each Event Server can perform different functions to suit the needs of the site.

Configuring the Number of Threads

The out-of-the-box .ccf file contains the full Event Server configuration, including:

- Reading from a switch and initial parsing (one thread)
- Guiding to customer (one thread)
- Guiding to service and rating (one thread)
- Persistence Writer functionality (one thread)
- Rerating (one thread)

The number of threads for these functions can be configured in the op_adj_env_sh script (for more information, see the “Environment Variables” section in this chapter). With every change, the Event Server must be restarted.

Configuring Thread Priority

The priority of threads can be configured in the op_adj_env_sh script (for more information, see the “Environment Variables” section in this chapter).

In addition, it is possible to define thread priority in the .ccf file as follows.



Note: The following instructions apply to the HP platform only.

To set the Real Time priority for threads:

1. In the initializers section, find the CPF_Default_ThreadsPropertiesInitializer initializer.
2. Set the Priority attribute to the appropriate value between **-128** and **-1**.
3. Set the Policy attribute to **RealTime**.

Example:

```
<initializer>
  <property name="ConcreteType" value="CPF_Default_ThreadsPropertiesInitializer" />
  <property name="Library Name" value="gcpf1fwc" />
  <property name="Entry Function" value="CPF_Default_ThreadsPropertiesInitializer_func" />
  <property name="Stack Size" value="Default" />
  <property name="Priority" value="-101" />
  <property name="Policy" value="RealTime" />
  <property name="Scope" value="Default" />
</initializer>
```

In this example, the Real Time policy (SCHED_RTPRIO) is set with priority -101. Priority -101 is reflected in the viewing utility as RTPRIO 100.

To reduce the priority of a certain task or thread:

1. In the tasks section, find the task whose priority you want to override.
2. Set the Override Priority attribute to **true**.
3. Set the Priority attribute to the appropriate value.

Example:

```
<property name="Override Priority" value="true" />
<property name="Priority" value="-108" />
```

In this example, the priority of the thread is -108. Priority -108 is reflected in the viewing utility as RTPRIO 107.

Database Connections

During initialization, the process connects to many reference tables. For a complete description of each table, see *Turbo Charging Data Model*.

After initialization, the Event Server connects primarily to the following tables.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Usage	APE1_ACCUMULATORS	Select, Update, Insert, Delete
Usage	APE1_EVENT_DUP_KEYS	Select, Insert, Update
Usage	APE1_EVENT_DUP_KEYS_EXT	Select, Insert, Update (if the USE_NEW_IOT_TABLE environment variable is set to 'Y')
Usage	APE1_NOTIFICATION_CONTOL	Insert
Usage	APE1_NOTIFICATIONS	Insert
Usage	APE1_RATED_EVENT	Select, Update, Insert

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Usage	APE1_REJECTED_EVENT	Update, Insert
Usage	APE1_RR_BLACK_LIST	Insert
Usage	APE1_SUBSCR_RERATE	Insert

If the required data is not found in shared memory (AIMOS), the Event Server process accesses the following tables.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Customer	APE1_SUBSCR_DATA	Select
Customer	APE1_SUBSCR_OFFERS	Select
Customer	APE1_SUBSCR_PARAMS	Select
Resource	AGD1_RESOURCES	Select

Admin Commands

The Event Server supports the following admin commands. For information on how to execute the commands, see the “[Handling Admin Commands](#)” section in the “[High Availability](#)” chapter.

Command	Description	Parameters
ACCUM_EAGERLOAD_RESUME_CYCLE	Resumes a specific cycle for the eager load of accumulators. This command can be executed by the shadow process.	--cycleCode “<Cycle code>”
ACCUM_EAGERLOAD_SUSPEND_CYCLE	Suspends a specific cycle for the eager load of accumulators. This command can be executed by the shadow process.	--cycleCode “<Cycle code>”
DELETE_CUSTOMER_COMMAND	<p>Deletes a single requested customer from the shared memory.</p> <p> Caution: <i>Do not use this command for mass removal of customers from the shared memory. The command does not physically remove the customer; it merely makes the customer invisible to the system. Therefore, extensive usage of this command causes a memory leak.</i></p>	--customerId <customer ID> --groupId <customer group ID>

Command	Description	Parameters
DELETE_RESOURCE_COMMAND	<p>Deletes a single requested resource from the shared memory.</p> <p> Caution: <i>Do not use this command for mass removal of resources from the shared memory. The command does not physically remove the resource; it merely makes the resource invisible to the system. Therefore, extensive usage of this command causes a memory leak.</i></p>	--resourceType <resource type> --resourceValue <resource value>
DELETE_SUBSCRIBER_COMMAND	<p>Deletes a single requested subscriber from the shared memory.</p> <p> Caution: <i>Do not use this command for mass removal of subscribers from the shared memory. The command does not physically remove the subscriber; it merely makes the subscriber invisible to the system. Therefore, extensive usage of this command causes a memory leak.</i></p>	--customerId <customer ID> --groupId <customer group ID> --subscriberId <subscriber ID>

Command	Description	Parameters
ENABLE_NETWORK_ELEMENT_COMMAND	<p>Activates an external port at run time without restarting the Event Server.</p> <p>This functionality is optional:</p> <ul style="list-style-type: none"> ■ To enable this functionality for a specific port, set the value of the corresponding ENABLE field of the ADJ1_NETWORK_ELEMENT_CFG table to 'N'. ■ To disable this functionality, leave the ENABLE field blank. ■ To ensure that a port is active at all times, set the value of the ENABLE field for that port to 'Y'. 	--id "<N1>;<N2>;...", where the values of the parameter are the network element IDs
DISABLE_NETWORK_ELEMENT_COMMAND	Disables an external port at run time without restarting the Event Server.	--id "<N1>;<N2>;...", where the values of the parameter are the network element IDs

Command	Description	Parameters
EVENT_SNIFFER_COMMAND	Traces incoming or outgoing events for a specified period of time or event count.	<p>START/STOP <Network source ID> <Start time> <Stop criteria> <Header/No header> <In/Out/Both>...</p> <p><i>Examples:</i></p> <ul style="list-style-type: none"> • START 1 HHMMSS Dhhmmss (duration in the 24-hour format) – Start recording at HHMMSS and continue for the given period of time (hhmmss). • START 2 NOW Dhhmmss (duration in the 24-hour format) – Start recording immediately and continue for the given period of time (hhmmss). • START 2 N560 – Start recording both incoming and outgoing events immediately and continue for the next 560 events. • STOP 2 IN – Stop recording incoming events from network source 2 immediately. • STOP 1 – Stop recording both incoming and outgoing events for network source 1 immediately. • START 2 160005 d102000 header both – Start recording at 16:00:05 for both incoming and outgoing events on network source 2 and continue for 10 hours and 20 minutes. Write into a dump file with headers.
REFRESH_CYCLE_DATA_COMMAND	Refreshes cycle data from the ADJ1_CYCLE_STATE table. This command can be executed by the shadow process.	--cycleCode "<Cycle code>" --cycleInstance "<Cycle instance>"
REFRESH_IMPL_TABLES_COMMAND	Refreshes all implementation reference tables. This command can be executed by the shadow process.	N/A
REFRESH_ALL_GENCODE_AND_IMPL_TABLES_COMMAND	Refreshes all generated implementation libraries and implementation reference tables. This command can be executed by the shadow process.	N/A

Command	Description	Parameters
SWITCH_ALL_GENCODE_AND_IMPL_TABLES_COMMAND	Enables the Event Server to work with the new implementation after reference data and libraries have been refreshed. Unless this command is issued, the process continues to work with the previously loaded implementation.	N/A
REFRESH_PROTOCOL_REF_DATA_COMMAND	Refreshes the protocol reference data container.	Reference table IDs delimited by a comma. If the reference table IDs are not specified, all tables are refreshed.
REMOVE_MEMORY_COMMAND	Removes accumulators related to a specific cycle instance and cycle code from shared memory. This command cannot be executed by the shadow process.	--cycleCode "<Cycle code>" --cycleInstance "<Cycle instance>" --cycleYear "<Cycle year>"
SET_EVENT_LOG_COMMAND	<p>Activates or stops the event log at runtime. The command name with its parameters must be enclosed in quotation marks.</p> <p> <i>Note: The command is handled asynchronously, so its effect is not immediate.</i></p>	<p>--target "\${TARGET}" --opt "\$OP", where:</p> <ul style="list-style-type: none"> ■ TARGET is the reporting point at which the event log must be activated or stopped. Values are: <ul style="list-style-type: none"> • ALL – All supported reporting points • CR – Outgoing Network Interface reporting point • FR – Guiding to Customer reporting point • RB – Event Processing reporting point ■ OP specifies whether the event log must be activated or stopped for the target reporting point. Values are: <ul style="list-style-type: none"> • ON • OFF <p>The wrapper script for this command is ADJ1_RunEventLog_Sh. It can be used as follows.</p> <p><i>Examples:</i></p> <ul style="list-style-type: none"> • ADJ1_RunEventLog_Sh <Process instance> ALL ON • ADJ1_RunEventLog_Sh <Process instance> CR ON • ADJ1_RunEventLog_Sh <Process instance> FR OFF • ADJ1_RunEventLog_Sh <Process instance> RB OFF

Command	Description	Parameters
SET_LIGHT_TRACER_COMMAND	Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters.	<p>To activate or stop the Light Tracer in various modes, specify the following parameters:</p> <ul style="list-style-type: none"> ▪ Activation parameters: <ul style="list-style-type: none"> • <code>--turnTracer "on/off"</code> – Activates or stops the Light Tracer • <code>--turnAssert "on/off"</code> – Activates or stops Assertions • <code>--turnLogOnTrace "on/off"</code> – Activates or stops writing log messages in Light Tracer files • <code>--turnFlowTracer/turnAll "on/off"</code> – Activates or stops tracing a single flow or event that is transferred between machines • <code>--turnAll "on/off"</code> – Activates or stops all of the above. <p>The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter).</p> <ul style="list-style-type: none"> ▪ One of the following sets of parameters specifying the module and the submodule: <ul style="list-style-type: none"> • Module ID only: <ul style="list-style-type: none"> ▫ <code>--moduleID "<n>/ALL"</code> – Applies the activation command to a specific module or to all modules. If only the module ID is specified, the command applies to all of its submodules as well. • Both module ID and submodule ID: <ul style="list-style-type: none"> ▫ <code>--moduleID "<n>"</code> – Applies the activation command to a specific module or to all modules. ▫ <code>--subModuleIDs "<n>"</code> – Applies the activation command to a specific submodule or to all submodules. <p> Note: A module corresponds to a project, and a submodule corresponds to a building block.</p> <p>For a list of the possible module and submodule IDs, see the “Light Tracer Command Parameters” section in this chapter.</p> <ul style="list-style-type: none"> ▪ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • <code>--threadInfo "Y/N"</code> – Specifies whether trace or assertion messages include the thread information.

Command	Description	Parameters
		<ul style="list-style-type: none"> • <code>--timestampInd "Y/N"</code> – Specifies whether trace or assertion messages include the time stamp. • <code>--contextInd "Y/N"</code> – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • <code>--noOfMessages "<number>"</code> – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down. • <code>--period "5/10/20/60/120/480/1440"</code> – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated. <p> Note: If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached.</p> <ul style="list-style-type: none"> • <code>--assertAbortInd "Y/N"</code> – Specifies whether the application is to be aborted if the assertion condition fails. ▪ <code>--commandType "changeBufferSize"</code> – Resizes the Light Tracer buffer at run time. • <code>--bufferSize "<number>"</code> – The new size for the Light Tracer buffer.
SET_TRACE_SQL_COMMAND	<p>Activates or stops SQL tracing at runtime.</p> <p>The command name with its parameters must be enclosed in quotation marks.</p> <p><i>Example:</i></p> <pre>ADJ1_Send_Admin_Command_Sh illin1027 <process instance> "SET_TRACE_SQL_COMMAND --traceSqlOn \"yes\" --duration \"1\""</pre>	<p><code>--traceSqlOn "<yes or no>" --duration "<minutes>"</code></p> <p>If the IS_DURATION_REQUIRED environment variable is set to 'Y', the --duration parameter is mandatory. If the duration specified in the command exceeds the allowed maximum value, SQL tracing is activated for the maximum allowed period of time only (60 minutes).</p> <p> Note: The SHUT_DOWN_CHECK_FREQ parameter of the Light Tracer in the ADJ1_CONF_SECTION_PARAM table determines the frequency (defined in the number of messages) at which the SQL Trace Manager checks whether the period specified in the --duration parameter has passed. Therefore, SQL tracing is stopped after the specified duration or after this number of messages has been reached, whichever comes later.</p>

Command	Description	Parameters
SWITCH_LIGHT_TRACER_ON	Activates the Light Tracer for a specific combination of module and submodules or for all modules, without the ability to specify the mode or send parameters.	<Module Name> For a list of the possible module names, see the “Light Tracer Command Parameters” section in this chapter.
SWITCH_LIGHT_TRACER_OFF	Stops the Light Tracer for a specific combination of module and submodules or for all modules.	<Module Name> For a list of the possible module names, see the “Light Tracer Command Parameters” section in this chapter.
EVENT_LIGHT_TRACER_CONFIG_COMM AND	Defines the events whose traces must be grouped and written to a trace per event (.toe) log. Eligible events are defined in terms of the network element that sent them and the flow that is applicable. The messages displayed for each event depend on the configuration of the Flow Tracer. For more information, see the “Configuration Parameters” section in this chapter.	--action “add/remove” --source_id <source ID> --flow <flow ID>, where: <ul style="list-style-type: none">■ Action defines whether to add or remove traces of events from a particular network element, in a particular flow, or both■ Source ID is the ID of the network element that sent the event■ Flow ID is one of the following:<ul style="list-style-type: none">• 1 – Main Event Server flow (Network Interface Function > Guiding to Customer Module > Event Processing Module > Replication > Network Interface Function)• 2 – Event Processing Module > Persistence Writer• ALL – Both flows
SET_PRE_SHUTDOWN_COMMAND	Sets the pre-shutdown flag. This flag is sent to the Event Server some time prior to a planned shutdown (for example, 5 minutes before).	N/A
RESET_PRE_SHUTDOWN_COMMAND	Resets the pre-shutdown flag. This command is sent to the Event Server to turn off the pre-shutdown flag if the Graceful Shutdown command is not sent.	N/A

Command	Description	Parameters
SET_PROTOCOL_TRACE_MODE ALL_ON	Enables tracing of every incoming or outgoing message for all protocols.	N/A
SET_PROTOCOL_TRACE_MODE ALL_OFF	Disables tracing for all protocols.	N/A
SET_PROTOCOL_TRACE_MODE PROTOCOL <protocol number> ON	Enables tracing every incoming or outgoing message for the specified protocol.	N/A
SET_PROTOCOL_TRACE_MODE PROTOCOL <protocol number> OFF	Disables tracing for the specified protocol.	N/A
CPF_GracefulShutdownCommand	Shuts down the Event Server gracefully. This command can be executed by the shadow process.	N/A

Examples:

- ADJ1_Send_Admin_Command_Sh hpbl820 ES 100 ACCUM_EAGERLOAD_SUSPEND_CYCLE --cycleCode "1"
- ADJ1_Send_Admin_Command_Sh hpbl820 ES 100 REMOVE_MEMORY_COMMAND --cycleCode "1" --cycleInstance "8" --cycleYear "2008"

Light Tracer Command Parameters

The following table shows the values of the moduleID and subModuleIDs parameters for the Light Tracer commands.

moduleID	subModuleIDs	Module Name	Comments
1	1	TC Rater	-
1	2	TC SR	Shift Responsibility
1	3	TC PW	Persistence Writer
1	4	TC ES Networking	Event Server Networking
1	5	TC FR	Guiding to Customer Module
1	6	TC Eager Load	-
1	7	TC ES GAT	Event Server – Generic Applicative Tables
1	8	TC Refresh Libraries	-
1	9	TC ES Rerate	Event Server – Rerate
2	1,2,3,4,5,6,7,8	TC Framework	-
2	12	TC FW	Framework
3	1	TC Connectability	-
5	1,2,3,4,5,6	TC Engine	-
6	1,2,3,4	TC Infrastructure	-
7	1,2,3	TC Guiding	-
8	12	TC PR	Processes
64	64	TC Log In Trace	-

Recovery Instructions

In an environment without the Availability Manager, rerun the process.

In an environment with the Availability Manager, the External Watchdog daemon reruns the process automatically.

Important Remarks

This section explains how to control the KPI, Event-Level Auditing, and Service Level Management mechanisms.

KPI Mechanism

The Key Performance Indicator (KPI) mechanism is used for performance check at the event level. It writes to the log a set of statistics, which indicates the period of time, in nanoseconds, that the input event spent in each processing station during its life cycle in the Event Server and the total event process time. The KPIs are gathered per input event.

Example:

```
TOTAL: Start-1319981904767696000 Diff-11221000 ; CRIN: Start-  
1319981904767759000 Diff-22000 ; CROUT: Start-1319981904778885000 Diff-6000  
; FR: Start-1319981904767817000 Diff-273000 ; RB: Start-1319981904810692000  
Diff-10318000 ; RB_GET_CUST: Start-1319981904819496000 Diff-6000 ;  
RB_PROCESS_EVENT: Start-1319981904819924000 Diff-696000 ;  
RB_DUP_CHECK: Start-1319981904819619000 Diff-289000 ;  
FR_GET_RESOURCES: Start-1319981904767820000 Diff-266000 ; SEND_NET:  
Start-1319981904778910000 Diff-6000 ; CONN_CRE: Start-1319981904810700000  
Diff-3000 ; GET_ACCUM: Start-1319981904820376000 Diff-5000 ;
```

To display KPI data in the logs, set the `KPI_TRACE_FREQUENCY` environment variable.

Event-Level Auditing and Service Level Management

Event-Level Auditing supplies Revenue Assurance teams with information about the completeness and comprehensiveness of the system, and its ability to deal successfully with requests that are sent to it. Service Level Management focuses on the performance of the system, its well-being, and compliance with the service level agreement (SLA). Despite their differences in purpose and users, the two mechanisms are very similar in their basic structure.

To enable Event-Level Auditing:

1. Activate the ELA Collector daemon. For more information, see the “Activation” section in the “ADJ1ELASRV – ELA Collector” chapter.
2. Set the following configuration parameter in the `APR1_CONF_SECTION_PARAM` table:

`SECTION_NAME = ELA, PARAM_NAME = ENABLE, PARAM_VALUE = Y`

For more information on these parameters, see the “Configuration Parameters” section in this chapter.

4 ADJ1FILE2ESRV – File2E

This chapter describes the File to Event Server (File2E) process.

Description

The File to Event Server (File2E) process extracts event data from the files that were prepared by Amdocs Acquisition & Formatting, an external mediation device, or the Rejected Event Recycler. After extracting the event data, File2E sends the events to the relevant Event Servers.

The File2E process extracts event data from files and sends the events to the relevant Event Servers.

One File2E process may distribute data to several different Event Server processes, according to the event resource type and value. The Event Servers may be located on the same machine as the File2E process, or on different machines.

Process Type

Daemon

Run Frequency

Runs continuously

Activation

Command Line:	ADJ1_File2E_Daemon_Shell_Sh -n <APPLICATION_ID> -c "-f <CCF_FILE_NAME>" -e <EWD_EXE>  Note: The -e parameter is optional. It is used in environments with Availability Manager.
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_File2E_Daemon_Shell_Sh
Executable Name:	gcpf1fwcApp

Example:

ADJ1_File2E_Daemon_Shell_Sh -n F2E500 -c "-f APR_File2E.ccf" -e gn1avm_ewd

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> F2E \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

ADJ1_Send_Admin_Command_Sh hpbl820 F2E 500
CPF_GracefulShutdownCommand

Dependent Processes

This process retrieves files that contain events and sends the events to the Event Server. The Event Server, in turn, updates the Accumulators and Events tables. Therefore, every process that retrieves updated data from the Events or Accumulators table depends on this process.

Affected Applications

File2E affects the following applications:

- *Event Server* – File2E sends events to the Event Server for processing
- *ELA Collector* – File2E participates in Event-Level Auditing for offline events at two reporting points: CRIn (when the event enters the system) and CROut (after the last processing stage). After File2E has aggregated the data gathered at Event-Level Auditing reporting points into messages, it sends these messages to an ELA Collector daemon. If several ELA Collector daemons are configured in the system, the File2E determines the correct ELA Collector daemon using an internal algorithm and sends all the events of a token to the same ELA Collector. The ELA Collector performs additional aggregations and comparisons of the data and then stores it in the relevant Turbo Charging and Audit & Control tables for reporting purposes.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*

ADJ1FILE2ESRV_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1FILE2ESRV_F2E500_20081109_130927_1.log

- *Console log files:*

ADJ1_File2E_Daemon_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1_File2E_Daemon_inner_Sh_F2E500_20081109_130927.log

- *Operational log files:*

ADJ1FILE2ESRV_<APPLICATION_ID>_\${ABP_MARKET}_<Date>_<Time>.log

Example:

ADJ1FILE2ESRV_F2E500_M3G_20081109_130927.log

- *Environment variable log files:*

ADJ1_File2E_Daemon_Shell_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1_File2E_Daemon_Shell_Sh_F2E500_20081109_130927.log

- *Light Tracer files, including event log* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG
- *Environment variable log files* – \$ABP_LOG
- *Light Tracer files, including event log* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter)

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information.
- *Operational log files* – The output of operational scripts.

- *Console log files* – Information that the process prints upon initialization before it opens the regular log file and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).

File2E Statistics

File2E can report the current state of event processing in the console log. Statistics are collected at different stages of File2E processing. These statistics help to improve the performance of File2E. For example, in a case of degraded performance of File2E, they can show which of the connected Event Servers is responsible.

Environment variables define whether statistics are collected and how often they are printed. For more information, see the “[Environment Variables](#)” section in this chapter.

Example:

```
(07:49:03.797859|67) FILE 357551331941200|started processing.  
(07:49:03.798131|67)  
F:357551331941200;TOT:93;FR(s:0|rs:0|r:0|nr:0|f:0|fr:0);RB(s:0|rs:  
0|r:0|nr:0|f:0|d:0|fr:0);DUP(0);GRDUP(0);MF(0);SU(0)  
(07:49:03.830770|67)  
F:357551331941200;TOT:232;FR(s:231|rs:0|r:0|nr:0|f:0|fr:0);RB(s:0|  
rs:0|r:0|nr:0|f:0|d:0|fr:0);DUP(0);GRDUP(0);MF(0);SU(0)  
(07:49:03.830984|67);F:357551331941200|FR wait enter...  
(07:49:03.844892|67);F:357551331941200|FR wait exit...  
(07:49:03.844938|67)  
F:357551331941200;TOT:232;FR(s:232|rs:0|r:232|nr:0|f:0|fr:0);RB(s:  
232|rs:0|r:0|nr:0|f:0|d:0|fr:0);DUP(0);GRDUP(0);MF(0);SU(0)  
(07:49:03.845140|67)  
F:357551331941200;TOT:232;FR(s:232|rs:0|r:232|nr:0|f:0|fr:0);RB(s:  
232|rs:0|r:0|nr:0|f:0|d:0|fr:0);DUP(0);GRDUP(0);MF(0);SU(0)  
(07:49:03.845181|67);F:357551331941200|RB wait enter...  
(07:49:04.212967|66);F:357551331937200|RB wait exit...  
(07:49:04.213360|66)  
F:357551331937200;TOT:34;FR(s:34|rs:0|r:34|nr:0|f:0|fr:0);RB(s:34|  
rs:0|r:34|nr:0|f:0|d:27|fr:0);DUP(0);GRDUP(0);MF(0);SU(0)  
(07:49:04.267477|66)  
F:357551331937200;TOT:34;FR(s:34|rs:0|r:34|nr:0|f:0|fr:0);RB(s:34|  
rs:0|r:34|nr:0|f:0|d:27|fr:0);DUP(0);GRDUP(0);MF(0);SU(0)  
(07:49:04.267554|66) FILE 357551331937200|ended processing|10000  
records|2 sec|AVG TPS 100.
```

The following statistics are included:

- (07:49:03.844938/67) – Time and thread ID
- F:357551331941200 – File identifier
- TOT:232 – Total number of records in the file
- FR(s:232/rs:0/r:232/nr:0/f:0/fr:0) – Statistics on the records sent to the Guiding to Customer Module:
 - Sent to the Guiding to Customer Module
 - Resent to the Guiding to Customer Module
 - Received by the Guiding to Customer Module
 - Not received by the Guiding to Customer Module

- Failed in the Guiding to Customer Module
- Forced rerun to the Guiding to Customer Module
- $RB(s:232/rs:0/r:0/nr:0/f:0/d:0/fr:0)$ – Statistics on the records sent to the Event Processing Module:
 - Sent to the Event Processing Module
 - Resent to the Event Processing Module
 - Received by the Event Processing Module
 - Not received by the Event Processing Module
 - Failed in the Event Processing Module
 - Dropped by the Event Processing Module
 - Forced rerun to the Event Processing Module
- $DUP(0)$ – Number of duplicate records
- $GRDUP(0)$ – Number of Geo Redundancy duplicate records
- $MF(0)$ – Number of records requiring a manual fix
- $SU(0)$ – Number of suspended cycle code records
- $FR\ wait\ enter$ – Indicates that File2E entered the waiting state after sending the records to the Guiding to Customer Module
- $FR\ wait\ exit$ – Indicates that File2E exited the waiting state after receiving the records from the Guiding to Customer Module
- $RB\ wait\ enter$ – Indicates that File2E entered the waiting state after sending the records to the Event Processing Module
- $RB\ wait\ exit$ – Indicates that File2E exited the waiting state after receiving the records from the Event Processing Module

When the entire file has been processed, the following statistics are included:

- Number of records in the file
- Time it took to process the entire file
- Average throughput for the file in transactions per second

In addition, statistics can be collected per Event Server. In this case, the statistics are presented in the following format:

```
(%T|%t)  
PER_ES_STAT;F:%lld;ES_ID:"%ES_ID%";ACC_NAME:"%ACC_NAME%";FR(%s:%u|%s:  
%u|%s:%u|%s:%u);RB(%s:%u|%s:%u|%s:%u|%s:%u);%s:%u;%s:%u;%s:%u;%  
s:%u;%s:%u;%s:%u;%s:%u;
```

where:

- $%T$ is the time stamp.
- $%ot$ is the thread ID.
- $%lld$ is the file ID.

- “%ES_ID%” is the ID of the Event Server.
- “%ACC_NAME%” is the name of the acceptor (regular acceptor, rerun acceptor, or resume acceptor).
- %s is the counter name, as follows:
 - ePending – “pen”
 - eFRSent – “s”
 - eFRArrived – “ar”
 - eRBSent – “s”
 - eRBArrived – “ar”
 - eReject – “rej”
 - eError – “err”
 - ePcError – “prcerr”
 - eFRExpired – “ex”
 - eRBExpired – “ex”
 - eFRCommError – “cerr”
 - eRBCommError – “cerr”
 - eDropped – “dr”
 - eSusCycle – “suscy”
 - eDuplicate – “dup”
 - eDupNotPersisted – “dupnp”



Note: In general, if an event was sent to an Event Server and no response is received, the event is reported in the “s” counter. However, when File2E is aware beforehand that the Event Server is not available, the event is reported in the “cerr” or “ex” counter (depending on the Event Server functionality and the stage at which it was disconnected).

- %u is the counter value (the number of events in the file with the corresponding status).



Note: File2E statistics per Event Server are provided for debugging purposes. They must not be used for generating formal data.

- *Environment variable log files* – Environment variables used by the process.
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

- *Event log files* – Fields with useful statistical information per event. There is a common unique identifier for lines *produced by the same event* in any module on any machine. This enables the user to construct a sequential history of an event as it progressed from module to module. File2E event log files are produced by the Light Tracer mechanism and include the following attributes:
 - Unique ID
 - Module = F2E
 - Event start handling time
 - Total handling time
 - Last handling result
 - Last handling result code
 - Resource type
 - Resource value
 - Resource segment
 - Customer segment
 - Customer ID
 - Suscriber ID
 - Cycle code
 - Event data
 - Network source
 - File identifier
 - Record number
 - Status
 - Last handling erro code
 - Number of retries to Guiding to Customer Module
 - Number of retries to Event Processing Module
 - Total handling time until Guiding to Customer Module
 - Total handling time until Event Processing Module

Input Files

The process takes input files according to the AC1_CONTROL table of Audit & Control. The table specifies the names and location of these files.

File2E retrieves files with the following data:

- *Next Program* – File2E
- *File Union* – TCUSAGE

The priority of input file types is set in the PRIORITY field of the AC_PGM_RULES_CONTROL table. If the priority mechanism is enabled (by setting the APR_ABP_F2E_ORDER_BY_PRIO_IND_<Acceptor> variable to ‘TRUE’), file types with a higher priority (whose priority *value* is lower) are processed first, and file types with the same priority are handled on a ‘first come – first served’ basis.

Output Files

This section describes the output files of the process.

Name

The names of the files are as follows:

- *For events requiring a manual fix* – ManFix_<counter>! <Input file name>
- *For events requiring an environment fix* – EnvFix_<counter>! <Input file name>
- *For expired events* – ReRun_<counter>! <Input file name>
- *For successful events* – DumSuc_<counter>! <Input file name>
- *For rejected events* – DumRej_<counter>! <Input file name>
- *For generated events* – DumGen_<counter>! <Input file name>
- *For events whose cycle is suspended* – Suspend_<counter>! <Cycle_code>_<Input file name>
- *For duplicated events* – Dup_<counter>! <Input file name>
- *For Geo Redundancy duplicated events* – GRDup_<counter>! <Input file name>
- If Pre-Balance reporting is enabled:
 - *For Pre-Balance reporting on an input file* – DumPreBalGen_<counter>! IN_<Input file name>
 - *For Pre-Balance reporting on an output file* – DumPreBalGen_<counter>! OUT_<Output file name>

Location

The output directory is located under \$ABP_APP_ROOT/interfaces/output/f2e/<file type>, where <file type> is one of the following:

- *duplicate* – Dup_ files
- *error* – ManFix_ files
- *grduplicate* – GRDup_ files
- *pcerr* – EnvFix_ files
- *rerun and rerun_2* – ReRun_ files
- *suspend* – Suspend_ files

The complete directory is determined at runtime according to the Audit & Control file allocation mechanism.

Contents

The contents of the files are as follows:

- *Manual Fix* – Contain the events that had an error in the Event Server and require manual intervention. The events in this file and their number are specified in the AC1_CONTROL table. These files are not recycled automatically. When the problem is fixed, the file format and the next program for such files must be changed so that File2E can reprocess them.
- *Environment Fix* – Contain the events that had an error in the Event Server as a result of environment issues (for example, because of a Rating Logic Configurator problem). The events in this file and their number are specified in the AC1_CONTROL table. These files are not recycled automatically. When the problem with the Rating Logic Configurator is fixed, the file format and the next program for such files must be changed so that File2E can reprocess them.
- *Rerun* – Contain all events that could not be passed to the Event Server as a result of a communication error. The events in this file and their number are specified in the AC1_CONTROL table.



Note: If a communication error occurs a second time, the event is written to the Rerun-2 file.

- *Dummy Success* – A dummy entry in the AC1_CONTROL table of Audit & Control. It contains the number of events that were inserted into the Rated Events table (APE1_RATED_EVENT).
- *Dummy Reject* – A dummy entry in the AC1_CONTROL table of Audit & Control. It contains the number of events that were inserted into the Rejected Events table (APE1_REJECTED_EVENT).
- *Dummy Generate* – A dummy entry in the AC1_CONTROL table of Audit & Control. It contains the number of events that were generated.
- *Suspend* – Contain all events whose cycle is in Suspend status. If there are two events that are marked as suspended, one related to cycle 5 and one to cycle 8, there are two Suspend output files: one for cycle 5 and one for cycle 8.
- *Duplicate* – An entry in the AC1_CONTROL table of Audit & Control. It contains the number of duplicated events.
- *Geo Redundancy (GR) Duplicate* – An entry in the AC1_CONTROL table of Audit & Control. It contains the number of Geo Redundancy duplicated events. Geo Redundancy duplicate events can arrive at File2E if the standby site that became active receives a file that was in the middle of processing at the originally active site when the switchover occurred. In the AC1_CONTROL table, such files are identified by the GR_IND field, which is set to ‘Y’. Because it is not possible to differentiate between technical duplicates and true duplicates in such files, the events are reported as Geo Redundancy (GR) duplicates.
- *Pre-balance* – Files created when Pre-Balance reporting is enabled. These files contain the number of events coming from the same physical file for the Pre-Balance report.

Flow

This section describes the process flow of File2E.

Process Start-up

At start-up, the File2E process performs the following:

1. Checks whether there are files that were not completed in the previous run because of process failure. If such files exist, File2E completes them before starting working on new files.
2. Starts handling all new files registered in Audit & Control according to criteria specified via an internal configuration tool.

Main Loop

Each thread in the process has its own main loop, and all threads work in parallel. Each thread performs the following as part of the main loop:

1. The File Handling component retrieves a new file from Audit & Control according to the specified criteria.
2. The File Handling component extracts an event from the retrieved file.
3. The File Handling component passes the event to the Event Processing Framework for processing. The Event Processing Framework performs the following:
 - a. Uses the Event Interface Module to decide on the Event Server that can perform guiding to customer. File2E determines the format of the input file records and parses them at this stage.
 - b. Sends the event to the corresponding Event Server for guiding to customer. The protocol that is used for sending the event is determined by the input file alias. The APR1_EVENT_FILE_ALIAS_CFG table maps the file alias to its corresponding protocol ID.
4. The File Handling component passes the response received from the Event Server that performed Guiding to Customer to the Response Handler. The response includes the processing status of the event (Processed or Dropped). This status indicates one of the following:
 - The message is to be sent to the Event Server for rating.
 - The message is to be sent to the Event Processing Module of the Event Server for updating the Rejected Events table.
 - An error occurred at the Guiding to Customer stage.

5. The Response Handler receives the messages and checks the cycle code status for each.

- If the status is Processed, the Response Handler checks whether the cycle was suspended.

A cycle is considered suspended in the following cases:

- If the cycle has been suspended.
- If the cycle has already been resumed, files were created for this cycle as a result of the suspension. Therefore, there is a check on the `CHECK_ACFILES_FOR_SUS` and `MAX_PENDING_AC_FILES` parameters to determine whether the cycle is still suspended.

Depending on the results of the check, the Response Handler performs the following:

- If the cycle has been suspended, marks the event for writing to a Suspend file
- If the cycle has not been suspended, sends the messages to the Event Server that is capable of rating the customer's events, in their original order
- If the status is Dropped, the event is marked accordingly. In this case, manual intervention is required.

6. The response received from the Event Processing Module of the Event Server is passed to the Response Handler. It includes the following:

- The status of the event, for example, Processed or Dropped. This status indicates whether the Event Server wrote the event into a table at the Event Processing stage.
- The error code, which supplements the status of the event.
- The number of notifications created for the event.

File2E gathers this information per event.

- If the status is Processed, the error code indicates into which table the event was inserted, as follows:
 - If the error code is `success` or `implementation_dropped`, the event is inserted into the `APE1_RATED_EVENT` table. In this case, the Response Handler sets the status of the event to Success.
 - Events with all other error codes are inserted into the `APE1_REJECTED_EVENT` table. In this case, the Response Handler sets the status of the event to Reject.

- If the status is Dropped, the event was not inserted into the database for a reason described by the error code. The event is handled as follows:
 - If there was a connection error with the Event Server, the Response Handler marks the event for rerun.
 - If the Event Server could not create the business entity, the Response Handler marks the event as requiring an environment error because it is likely that the error was caused by a problem in the Rating Logic Configurator.
 - If the error code is `implementation_dropped`, the event is marked with the relevant status to update the corresponding input file counter (DR_REC_QUANTITY in the AC1_CONTROL table).
 - Events with all other error codes require manual intervention. In this case, the Response Handler sets the status of the event to Manual Error.
 - If the status is GRDuplicated, the number of Geo Redundancy duplicate events in the GRDuplicate entry in the AC1_CONTROL table is incremented.
 - If the status is Duplicated, the event is handled as follows:
 - If the event is truly a duplicate event, the number of duplicate events in the Duplicate entry in the AC1_CONTROL table is incremented.
 - Otherwise, the event is duplicated due to Recovery flow and is treated as a successfully processed one.
 - If the status is Technical Duplicate, File2E checks two bits on the event counters returned from Event Server and handles the event as follows:
 - If File2E is in Rerun or Recovery:
 - ◆ If the drop bit is returned, the event is marked as dropped.
 - ◆ If the drop bit is not returned, the event is marked as a successfully processed one.
 - If File2E is not in Rerun or Recovery, the event is marked as a duplicate.
7. File2E repeats steps 2 to 6 until all events contained in the retrieved file have been passed to the Event Processing Framework.
 8. Sending a message may fail with the EWOULDBLOCK error, which indicates that the socket is not available for the operation. When this happens a configurable number of times (the default is five times), File2E sleeps for a configurable time interval (the default is 20000 milliseconds) before retrying to send the message. Continuous sleep time is accumulated, and once it passes a configurable maximum sleep time (the default is -1), File2E disconnects the thread from the channel. Next time File2E needs to send a message, the thread attempts to reconnect to the channel.



Note: These parameters are configurable under the Amdocs Real-Time (ART) framework. File2E invokes the 'set' method of each parameter (`int m_numOfRetriesBeforeSleep`, `int m_sleepTime`, and `int m_timeBeforeDisconnect`) to set the required value.

9. The File Handling component waits for answers for all read events from the Event Processing Framework.
10. When done processing the file, File2E writes the information of each event to the relevant output file and updates its auditing counters. If File2E works with Pre-Balance reporting, it creates a pre-balance file for both input and output files, prefixed with IN and OUT, respectively.
11. File2E summarizes the number of written events as follows:

The number of events processed by the Event Server (with the Processed status) + the number of events written by File2E
12. File2E summarizes the number of dropped events.
13. File2E summarizes the number of generated events (notifications).
14. File2E summarized business audit counters.



Note: Multi-Service Credit Control (MSCC) files that arrive in degraded mode are audited using the Event-Level Auditing (ELA) mechanism, not Audit & Control.

For such files, audit counters in Audit & Control are populated as follows:

- *Technical counters are set to 1 per frame event.*
- *Business counters are not populated at all.*

Process Shutdown

Because File2E is a daemon, it only shuts down if it receives a specific manual request to do so. Before shutting down, File2E completes the files in processing.

Environment Variables

File2E uses the following variables that are initialized by the op_adj_env_sh script.

Variable Name	Description	Valid Values/ Default Value
APR_AB_P2E_ORDER_BY_PRIO_IND_RECOVERY_ACC	Determines whether the Recovery acceptor must handle files according to the priority defined in the AC_PGM_RULES_CONTROL table. If this variable is set to ‘FALSE’ or not set, all files are handled on a ‘first come – first served’ basis.	FALSE
APR_AB_P2E_ORDER_BY_PRIO_IND_REG_ACC	Determines whether the Regular acceptor must handle files according to the priority defined in the AC_PGM_RULES_CONTROL table. If this variable is set to ‘FALSE’ or not set, all files are handled on a ‘first come – first served’ basis.	FALSE

Variable Name	Description	Valid Values/ Default Value
APR_AB_P2E_ORDER_BY_PRIO_IND_RERUN_ACC	Determines whether the Rerun acceptor must handle files according to the priority defined in the AC_PGM_RULES_CONTROL table. If this variable is set to ‘FALSE’ or not set, all files are handled on a ‘first come – first served’ basis.	FALSE
APR_AB_P2E_ORDER_BY_PRIO_IND_RESUME_ACC	Determines whether the Resume acceptor must handle files according to the priority defined in the AC_PGM_RULES_CONTROL table. If this variable is set to ‘FALSE’ or not set, all files are handled on a ‘first come – first served’ basis.	FALSE
APR_F2E_AC_CACHE_SIZE	The size of the cache in Audit & Control.	1000
APR_F2E_BACKLOG_SECONDS_RECOVERY_ACC	The backlog time (in seconds) for the Recovery acceptor.	10
APR_F2E_BACKLOG_SECONDS_REG_ACC	The backlog time (in seconds) for the Regular acceptor.	10
APR_F2E_BACKLOG_SECONDS_RERUN_ACC	The backlog time (in seconds) for the Rerun acceptor.	10
APR_F2E_BACKLOG_SECONDS_RESUME_ACC	The backlog time (in seconds) for the Resume acceptor.	10
APR_F2E_CR_IN_NUM_OF_THREADS	The number of incoming Event Interface (also known as CR Incoming) threads in File2E.	3
APR_F2E_CR_OUT_NUM_OF_THREADS	The number of outgoing Event Interface (also known as CR Outgoing) threads in File2E.	2
APR_F2E_ERROR_FILE_UNION	The file union for the Manual Fix files.	As defined in the AC_PGM_RULES_CONTROL table
APR_F2E_FR_RESP_NUM_OF_THREADS	The number of Guiding to Customer response (also known as ResponseFR) threads in File2E.	2
APR_F2E_NUM_REG_ACC_INSTANCES	The number of instances of the regular acceptor.	1
APR_F2E_NUM_RERUN_ACC_INSTANCES	The number of instances of the rerun acceptor.	1
APR_F2E_NUM_RSP_ACC_INSTANCES	The number of instances of the resume acceptor.	1
APR_F2E_PCERR_FILE_UNION	The file union for the Environment Fix files.	As defined in the AC_PGM_RULES_CONTROL table
APR_F2E_RB_RESP_NUM_OF_THREADS	The number of Event Processing response (also known as Response RB) threads in File2E.	2

Variable Name	Description	Valid Values/ Default Value
APR_F2E_RCV_KILOBYTE_NUM	The size of the receiving socket.	<ul style="list-style-type: none"> ■ <i>HP</i> – 24576 ■ <i>AIX</i> – 6
APR_F2E_RECOVERY_CHECK_BALANCE	<p>Indicates whether the balance check is enabled for the recovery files. This check performs the following:</p> <ul style="list-style-type: none"> ■ Checks the internal balance for generated records ■ Checks the balance between the previous application and the current one <p>In regular scenarios, the balance check must be turned on. For revenue recovery files, it must be turned off.</p>	True
APR_F2E_REGULAR_CHECK_BALANCE	<p>Indicates whether the balance check is enabled for the regular files. This check performs the following:</p> <ul style="list-style-type: none"> ■ Checks the internal balance for generated records ■ Checks the balance between the previous application and the current one <p>In regular scenarios, the balance check must be turned on. For revenue recovery files, it must be turned off.</p>	True
APR_F2E_REGULAR_FILE_UNION	The file union for regular files.	As defined in the AC_PGM_RULES_CONTROL table
APR_F2E_RERUN_2_FILE_UNION	The file union for the Rerun-2 files.	As defined in the AC_PGM_RULES_CONTROL table
APR_F2E_RERUN_CHECK_BALANCE	<p>Indicates whether the balance check is enabled for the Rerun files. This check performs the following:</p> <ul style="list-style-type: none"> ■ Checks the internal balance for generated records ■ Checks the balance between the previous application and the current one <p>In regular scenarios, the balance check must be turned on. For revenue recovery files, it must be turned off.</p>	True
APR_F2E_RERUN_FILE_UNION	The file union for the Rerun files.	As defined in the AC_PGM_RULES_CONTROL table
APR_F2E SND_KILOBYTE_NUM	The size of the sending socket.	<ul style="list-style-type: none"> ■ <i>HP</i> – 24576 ■ <i>AIX</i> – 6

Variable Name	Description	Valid Values/ Default Value
APR_F2E_SUSPEND_CHECK_BALANCE	<p>Indicates whether the balance check is enabled for the Suspend files. This check performs the following:</p> <ul style="list-style-type: none"> ▪ Checks the internal balance for generated records ▪ Checks the balance between the previous application and the current one <p>In regular scenarios, the balance check must be turned on. For revenue recovery files, it must be turned off.</p>	True
APR_F2E_SUSPEND_FILE_UNION	The file union for the Suspend files.	As defined in the AC_PGM_RULES_CONTROL table
APR_F2E_TCP_SLEEP_EVENT_FR	The number of events that File2E must send to the Event Server performing the Guiding to Customer function before sleeping for 100 milliseconds. This parameter controls the rate with which events are sent to the Event Server.	200
APR_F2E_TCP_SLEEP_EVENT_RB	The number of events that File2E must send to the Event Server functioning as the Event Processing module before sleeping for 100 milliseconds. This parameter controls the rate with which events are sent to the Event Server.	200
APR_F2E_WAITTIME_MILLISECOND_FR	The maximum amount of time (in milliseconds) to wait for an event to be processed by the Guiding to Customer Module.	100000
APR_F2E_WAITTIME_MILLISECOND_RB	The maximum amount of time (in milliseconds) to wait for an event to be processed by the Event Processing Module.	200000
EVENTS_NUM_FOR_ALL_ACC_THRESHOLD	<p>The threshold for the activation of an additional acceptor. If there is a free acceptor, the master thread checks whether the number of events handled by File2E is smaller than this parameter.</p> <ul style="list-style-type: none"> ▪ If so, the master thread activates an additional acceptor to retrieve and handle a file from Audit & Control. ▪ If not, the free acceptors continue to be idle. 	100000
F2E_STATISTIC_ENABLE	Determines whether File2E statistics are printed to the console log.	Y/N

Variable Name	Description	Valid Values/ Default Value
F2E_STATISTIC_PERIOD	The interval at which File2E statistics are printed to the log (in milliseconds).	N/A
F2E_STATISTIC_PER_ES_ENABLED	Determines whether File2E statistics per Event Server are printed to the console log.	N

For more information on the script, see the “Amdocs Batch Job Manager Application Script” section in the “Introduction” chapter.

Parameters

The following parameters must be set for the File2E process to run properly.

Parameter	Description	Default Value
APPLICATION_ID	The process instance	F2E500
CCF_FILE_NAME	An XML file that contains the relevant configuration	APR_File2E.ccf

Configuration Parameters

The configuration parameters of File2E are defined in the APR1_CONF_SECTION_PARAM table and can be modified using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*).

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	config	RADAR_ENABLED	<code> \${RADAR_ENABLE_D}</code>	Indicates whether Pre-Balance reporting is enabled. The value of this parameter is an environment variable that must be defined in runtime environments. Its default value is ‘N’.
F2E	CHANNEL_STREAM	RETRIES_BEFORE_SLEEP	5	If a socket is full, the number of times that File2E retries to resend an event before going to sleep.
F2E	CHANNEL_STREAM	SLEEP_TIME	20	If a socket is full, the number of milliseconds for which File2E stops sending events.
F2E	CHANNEL_STREAM	TIME_BEFORE_DISCONNECT	120	If a socket is full, the number of milliseconds for which File2E tries to send events via the socket since it became full. If this time has passed, but the socket is still full, File2E closes the socket.
F2E	config	CHECK_ACFILES_FOR_SUS	true	If a cycle is resumed, indicates whether to check for files in RD (Ready) status for this cycle.
F2E	config	EVENTS_NUM_FOR_ALL_ACCEPTOR_THRESHOLD	<code> \${EVENTS_NUM_FOR_ALL_ACCEPTOR_THRESHOLD}</code>	The threshold for the activation of an additional acceptor. If there is a free acceptor, the master thread checks whether the number of events handled by File2E is smaller than this parameter. <ul style="list-style-type: none"> ▪ If so, the master thread activates an additional acceptor to retrieve and handle a file from Audit & Control. ▪ If not, the free acceptors continue to be idle.
F2E	config	F2E_RCV_KILOBYTE_NUM	<code> \${APR_F2E_RCV_KILOBYTE_NUM}</code>	The size of the receiving socket.
F2E	config	F2E_RCV_MSG_BULK	10	The number of events that File2E attempts to read from one socket in one go.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
F2E	config	F2E_SND_KILOBYTE_NUM	\${APR_F2E_SND_KILOBYTE_NUM}	The size of the sending socket.
F2E	config	F2E_TCP_SLEEP_EVENT_FR	\${APR_F2E_TCP_SLEEP_EVENT_FR}	The number of events that File2E must send to the Event Server performing the Guiding to Customer function before sleeping for 100 milliseconds. This parameter controls the rate with which events are sent to the Event Server.
F2E	config	F2E_TCP_SLEEP_EVENT_RB	\${APR_F2E_TCP_SLEEP_EVENT_RB}	The number of events that File2E must send to the Event Server functioning as the Event Processing module before sleeping for 100 milliseconds. This parameter controls the rate with which events are sent to the Event Server.
F2E	config	FILES_NUMBER_IN_DIR	50	The maximum number of files in a directory.
F2E	config	FR_MAX_ADDITION_TRIES	2	The maximum number of times that File2E must attempt to resend an event to the Guiding to Customer Module when File2E does not receive a response on time. After this number of attempts, File2E declares the event as expired and writes it to the Rerun file. This is not an out-of-the-box parameter. It must be added to override the default value.
F2E	config	MAX_ADDITION_TRIES	2	The number of additional attempts to send the event to the Event Processing Module.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
F2E	config	MAX_PENDING_AC_FILES	3	If the CHECK_ACFILES_FOR_SUS parameter is set to true, indicates the maximum number of files that can be in RD (Ready) status for the cycle to be considered resumed.
F2E	config	MEM_ALLOC	1000	The maximum number of events that File2E can process in one bulk. If a file is large, it is processed in bulks.
F2E	config	NUM_OF_RB_STICKY_QUEUE	1	The number of sticky queues that the Router RB module (the module that is connected to the Event Server functioning as the Event Processing module) has per file.
F2E	config	OUTPUT_DUPLICATE_FILE_PA TH	\${ABP_APP_ROOT}/interfaces/output/f2e/duplicate	The location of Duplicate files.
F2E	config	OUTPUT_ERROR_FILE_PATH	\${ABP_APP_ROOT}/interfaces/output/f2e/error	The location of Manual Fix files.
F2E	config	OUTPUT_FILE_PATH	\${ABP_APP_ROOT}/interfaces/output	The location of the output files of File2E.
F2E	config	OUTPUT_GR_DUPLICATE_FIL E_PATH	\${ABP_APP_ROOT}/interfaces/output/f2e/gr duplicate	The location of Geo Redundancy duplicate files.
F2E	config	OUTPUT_PCERR_FILE_PATH	\${ABP_APP_ROOT}/interfaces/output/f2e/p cerr	The location of Environment Fix files.
F2E	config	OUTPUT_RERUN_2_FILE_PA TH	\${ABP_APP_ROOT}/interfaces/output/f2e/re run_2	The location of Rerun-2 files.
F2E	config	OUTPUT_RERUN_FILE_PATH	\${ABP_APP_ROOT}/interfaces/output/f2e/re run	The location of Rerun files.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
F2E	config	OUTPUT_SUSPEND_FILE_PATH	<code> \${ABP_APP_ROOT}/interfaces/output/f2e/suspend</code>	The location of Suspend files.
F2E	config	PROGRAM_NAME	File2E	The name of File2E.
F2E	config	SYNCH_GRACEFULL_SHUTDOWN	N	<p>Defines the behavior of the process upon receiving the Graceful Shutdown command from the Availability Manager:</p> <ul style="list-style-type: none"> ■ <i>Y</i> – The process sends an ACK message to the Availability Manager when the graceful shutdown is complete. ■ <i>N</i> – The process sends an ACK message to the Availability Manager when it receives the command and starts to shut down.
F2E	config	VIRTUAL_OR_SHARED_MEMORY_USED	VIRTUAL	Indicates whether the process uses virtual or shared memory.
F2E	config	WAITTIME_MILLISECOND_FR	<code> \${APR_FR_WAITTIME_MILLISECOND_FR}</code>	The maximum amount of time (in milliseconds) to wait for an event to be processed by the Guiding to Customer Module.
F2E	config	WAITTIME_MILLISECOND_RB	<code> \${APR_FR_WAITTIME_MILLISECOND_RB}</code>	The maximum amount of time (in milliseconds) to wait for an event to be processed by the Event Processing Module.
F2E	F2E_LF_STICKY_QUEUE	F2E_LF_STICKY_SLEEP_MILLI_INTERVAL	0	<p>The locking strategy for the critical sections of the sticky queue if there are no events between the fetch tries of the dequeue function. Possible values are:</p> <ul style="list-style-type: none"> ■ <i>0 (default)</i> – Use the tryacquire and yield functions in the case of a retry. ■ <i>999 milliseconds and up</i> – Use the acquire function. ■ <i>Any other value</i> – Use the tryacquire function and sleep for the specified period of time (in milliseconds) in the case of a retry.

In addition, the process uses Light Tracer configuration parameters, which are defined in the ADJ1_CONF_SECTION_PARAM table. For more information, see **Light Tracer Parameters** in the “**Configuration Parameters**” section in the “**ADJ1EVENTSRV – Event Server**” chapter.

For each process instance, an entry with the name of the process instance must be created in the APR1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
F2E500	F2E

Any parameter to be overridden at the process instance level must be added to the APR1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
F2E500	DUMMY_DUPLICATE_FILE	ENABLED	Y

For more information on parameters and the properties mechanism, see the “**Configuration Parameter Mechanism**” appendix.

Configuration Files

The File2E Server process requires a configuration file (.ccf) that is defined via an internal configuration tool.

Only a few variables can be customized. The environment variables that appear in the .ccf file and can be customized are defined in the op_adj_env_sh script (for more information, see the “**Environment Variables**” section in this chapter).

There are variables that can be customized in the .ccf file itself:

- In <task_properties> CR Incoming <property name="Task Name" value="CR Incoming" />, the decision whether the guiding to customer is to be local or remote (<property name="FR is local" value="false" />) can be made. The default value is remote (false).
- The number of threads of the following File2E modules can be changed by modifying the corresponding <property name="Threads" value="1" /> line:
 - RouterRB
 - RouterFR

The following settings are recommended:

- The number of threads of the RouterRB should be approximately NUM_OF_RB_STICKY_QUEUE * APR_F2E_NUM_REG_ACC_INSTANCES.
- The number of threads of the ResponseFR task should be equal to or less than APR_F2E_NUM_REG_ACC_INSTANCES + APR_F2E_NUM_RERUN_ACC_INSTANCES + APR_F2E_NUM_RSP_ACC_INSTANCES.

Database Connections

During initialization, the process connects to many reference tables. For a complete description of each table, see *Turbo Charging Data Model*.

After initialization, File2E connects primarily to the following tables.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	AC1_CONTROL	Select
Configuration	APR1_EVENT_FILE_ALIAS_CFG	Select
Resource	AGD1_RESOURCES	Select

Admin Commands

File2E supports the following admin commands. For information on how to execute the commands, see the “[Handling Admin Commands](#)” section in the “[High Availability](#)” chapter.

Command	Description	Parameters
ADMIN_F2E_RSP_CYC_CMD	Resumes the cycle	--cycleCode “<Cycle code>”
ADMIN_F2E_SUS_CYC_CMD	Suspends the cycle	--cycleCode “<Cycle code>”
SET_LIGHT_TRACER_COMMAND	Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters	<p>To activate or stop the Light Tracer in various modes, specify the following parameters:</p> <ul style="list-style-type: none"> ■ Activation parameters: <ul style="list-style-type: none"> • <i>turnTracer “on/off”</i> – Activates or stops the Light Tracer • <i>--turnAssert “on/off”</i> – Activates or stops Assertions • <i>--turnLogOnTrace “on/off”</i> – Activates or stops writing log messages in Light Tracer files • <i>--turnFlowTracer/turnAll “on/off”</i> – Activates or stops tracing a single flow or event that is transferred between machines • <i>--turnAll “on/off”</i> – Activates or stops all of the above. The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter). ■ <i>--moduleID “ALL”</i> ■ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • <i>--threadInfo “Y/N”</i> – Specifies whether trace or assertion messages include the thread information. • <i>--timestampInd “Y/N”</i> – Specifies whether trace or assertion messages include the time stamp. • <i>--contextInd “Y/N”</i> – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • <i>--noOfMessages “<number>”</i> – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down.

Command	Description	Parameters
		<ul style="list-style-type: none"> • <code>--period "5/10/20/60/120/480/1440"</code> – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated. <p> Note: <i>If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached.</i></p> <ul style="list-style-type: none"> • <code>--assertAbortInd "Y/N"</code> – Specifies whether the application is to be aborted if the assertion condition fails. ■ <code>--commandType "changeBufferSize"</code> – Resizes the Light Tracer buffer at run time. <ul style="list-style-type: none"> • <code>--bufferSize "<number>"</code> – The new size for the Light Tracer buffer. <p>To activate or stop only the event log , which provides useful data at the File2E reporting point, specify all of the following parameters:</p> <ul style="list-style-type: none"> ■ <code>--turnTracer "on/off"</code> ■ <code>--moduleID "1"</code> ■ <code>--subModuleIDs "11"</code> <p> Note: <i>The event log is automatically activated or stopped when the Light Tracer is activated or stopped. These parameters are used to activate or stop the event log separately from the Light Tracer, regardless of how it was activated.</i></p>

Command	Description	Parameters
SET_TRACE_SQL_COMMAND	<p>Activates or stops SQL tracing at runtime.</p> <p>The command name with its parameters must be enclosed in quotation marks.</p> <p><i>Example:</i> ADJ1_Send_Admin_Command_Sh illin1027 <i><process instance></i> "SET_TRACE_SQL_COMMAND --traceSqlOn \"yes\" --duration \"1\""</p>	<pre>--traceSqlOn "<yes or no>" --duration "<minutes>"</pre> <p>If the IS_DURATION_REQUIRED environment variable is set to 'Y', the --duration parameter is mandatory. If the duration specified in the command exceeds the allowed maximum value, SQL tracing is activated for the maximum allowed period of time only (60 minutes).</p>  <p>Note: The SHUT_DOWN_CHECK_FREQ parameter of the Light Tracer in the ADJ1_CONF_SECTION_PARAM table determines the frequency (defined in the number of messages) at which the SQL Trace Manager checks whether the period specified in the --duration parameter has passed. Therefore, SQL tracing is stopped after the specified duration or after this number of messages has been reached, whichever comes later.</p>
SWITCH_LIGHT_TRACER_ON ALL	Activates the Light Tracer without the ability to specify the mode or send parameters	N/A
SWITCH_LIGHT_TRACER_OFF ALL	Stops the Light Tracer	N/A
CPF_GracefulShutdownCommand	Shuts down File2E gracefully	N/A

Examples:

- ADJ1_Send_Admin_Command_Sh hpbl820 F2E 500 ADMIN_F2E_SUS_CYC_CMD --cycleCode "1"
- ADJ1_Send_Admin_Command_Sh hpbl820 F2E 500 ADMIN_F2E_RSP_CYC_CMD --cycleCode "1"

Recovery Instructions

After a process or system failure, rerun the process.

If File2E crashes, the subsequent File2E run detects that some files were not completed. In this case, File2E starts its work by reprocessing the files that were in use when failure occurred. All the events contained in these recovered files are passed to the Event Processing Framework with the Check Duplicate indicator set to prevent double charging.

5 ADJ1UH – Update Handler in Full and Incremental Mode

This chapter describes the Update Handler in Full and Incremental mode.

Description

The Update Handler is a multi-threaded component in Turbo Charging that extracts data from a customer management system (such as Amdocs Billing Customer Manager) and loads it to a target database (such as the Turbo Charging Customer database).

In Full mode, the Update Handler updates a mass of data from the customer management system to the target database. In Incremental mode, the Update Handler updates data changes only rather than the whole mass of data. The update is asynchronous and is based on transactions published to the Transaction Broker by the customer management system.

Depending on the implementation, the Update Handler can work on various types of data and at various levels, providing the following functionality:

- *Guiding and Rating modes* – In these modes, the Update Handler extracts the data required for guiding to customer or for rating from the Amdocs Billing Customer Manager database and loads it into the Turbo Charging Customer database. These modes are described in this chapter.
- *Synchronous mode* – In this mode, the Update Handler is deployed on the application server and thus is invoked directly. This enables the customer management system to publish the changes in customer data to the Update Handler APIs synchronously, without using any request adapters. For more information, see the “[ADJ1UH – Update Handler in Synchronous Mode](#)” chapter.
- *Mark for Rerate mode* – In this mode, the Update Handler takes Audit & Control files and inserts requests based on the file contents into the APE1_SUBSCRIBER_RERATE database table. The requests pertain to all subscribers who match the predefined criteria for rerating. For more information, see the “[ADJ1UHMARK4RE – Update Handler in Mark for Rerate](#)” chapter.
- *Reconciliation mode* – In this mode, the Update Handler is used to perform reconciliation between the shared memory of the Event Server and the data extracted from the database. The Update Handler compares the data and may also synchronize data between the memory and the database. For more information, see the “[ADJ1RCN – Update Handler in Reconciliation Mode](#)” chapter.
- *Mini-Full mode* – In this mode, the Update Handler updates the data of a specific customer or subscriber rather than the data of the entire population. For more information, see the “[ADJ1UHMINFULL – Update Handler in Mini-Full Mode](#)” chapter.
- *Closed User Group (CUG) mode* – In this mode, the Update Handler extracts from the customer management system database the lists of resources per group entitled for reduced rates or discounts on tariffs. Then the Update Handler updates the Turbo Charging database with this information. For more information, see the “[ADJ1UHCUG – Update Handler in CUG Mode](#)” chapter.

Process Type

The Update Handler has two process types:

- *Incremental mode* – Daemon (ADJ1UHD)
- *Full mode* – Batch job (ADJ1UHB)

Run Frequency

- *Full mode* – Runs by request, when refresh of all data is necessary. Run frequency is therefore determined by customer need.
- *Incremental mode* – Runs continuously.

Activation

This section describes how to activate the Update Handler in various modes.

Full Mode

Command Line:	ADJ1_UH_Job_Shell_Sh -n <APPLICATION_ID> -c "-profileFile <The name of the profile file> -runningMode <Running mode> -envVar<Environment variables> -debugPort <Debug port number>" -e <EWD_EXE> -l
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_UH_Job_Shell_Sh
Executable Name:	N/A

Example:

```
ADJ1_UH_Job_Shell_Sh -n UHF_GD1110 -c "-profileFile
UHImplFullModeGuidingDelete -runningMode NORMAL -debugPort 1234" -e
gn1avm_ewd
```

Incremental Mode

Command Line:	ADJ1_UH_Daemon_Shell_Sh -n <APPLICATION_ID> -c "-profileFile <The name of the profile file> -runningMode <Running mode> -envVar<Environment variables> -debugPort <Debug port number>" -e <EWD_EXE> -l
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_UH_Daemon_Shell_Sh
Executable Name:	N/A

Example:

```
ADJ1_UH_Daemon_Shell_Sh -n UHI_GD1210 -c "-profileFile  
UHIImplIncModeGuiding -runningMode FORCE_START -debugPort 1234" -e  
gn1avm_ewd
```

For more information about the activation scripts, see the “[Scripts](#)” section in the “[Introduction](#)” chapter.

Shutdown

The Update Handler in Incremental mode can be shut down gracefully using the following command.

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> <procType> <instanceNumber> 100
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

```
ADJ1_Send_Admin_Command_Sh hpbl820 UHI_GD 1210 100
```

When the Update Handler in Full mode is done, it shuts down on its own.

Preceding Processes

The following process must run successfully before this process is activated:

- *Implementation Compiler* – The Implementation Compiler creates an SQL file, which contains the data of the ADJ1_SUBSCR_ATTRIBUTES table. The table contains the mapping between the generated attribute IDs, and the attribute names in the Rating Logic Configurator and in the Amdocs Billing Customer Manager. The Update Handler takes the attribute name in Amdocs Billing Customer Manager and populates the database with the corresponding ID.

Affected Applications

The process affects the Event Server, the main processing daemon of Turbo Charging.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*

```
 ${TASK_NAME}_${uhProcInstanceName}`date  
 +%Y_%m_%d_%H_%M_%S.log
```

Example:

ADJ1UH_UHI_GD1210_2010_06_08_09_09_44.log

- *Console log files:*

```
 ${TASK_NAME}_${uhProcInstanceName}_console`date  
 +%Y%m%d%H%M%S.log
```

Example:

ADJ1UH_UHI_GD1210_console_20100608_090944.log

TASK_NAME can be one of the following:

- *ADJ1UH* – Full or Incremental mode
- *ADJ1UHMARK4RE* – Mark for Rerate mode
- *ADJ1RCN* – Reconciliation mode
- *ADJ1UHCUG* – CUG mode
- *ADJ1MINFUL* – Mini-Full mode

Location

Log files are located in the following directories:

- *Log files* – \${ABP_AJTUH_ROOT}/logs
- *Console log files* – \${ABP_AJTUH_ROOT}/console

Contents

The log files contain the following:

- *Log files* – Information regarding any errors that occurred during the Update Handler run. If the Update Handler ran in debug mode, the logs also contain all debugging messages.
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).

Input Files

This section describes the input files.

Update Handler Implementation File

The input to the process is the implementation profile XML file of the Update Handler.

Location

The process extracts the file from the AUH1_PROFILES table. The name of the profile XML file is supplied to the script of the Update Handler.

Contents

This file describes the actions that the Update Handler processes must perform. For a full description, see *Turbo Charging Update Handler XML Configuration Guide*.

Amdocs Billing Customer Manager Transaction Files

These files contain transactions that Amdocs Billing Customer Manager sends to the Update Handler via the Transaction Broker.

Location

The Subscriber Transactions Log (TRB1_SUB_LOG) table of the Transaction Broker

Contents

Various data from Amdocs Billing Customer Manager, for example, New Customer, Change Offer, or Change Price Plan transactions

Audit & Control Files

These are files that Audit & Control sends to the Update Handler.

Location

The file system, according to input from Audit & Control, which provides the location of the files (and not the files themselves)

Contents

Different implementations may contain different data. Currently, in the out-of-the-box implementation, the Update Handler gets files from Amdocs Invoicing. These files contain data on the customers that need to be marked for rerate.

Output Files

The output of the process is the data in tables with the APE prefix and the Update Handler control tables. For more information, see *Turbo Charging Data Model*.

Flow

The following subsections describe the high-level workflows of the Update Handler, in both Full and Incremental modes.

Initialization

During the initialization stage, the process creates the required temporary tables as defined in the implementation XML file. These tables are dropped when the process ends, or when a process instance is restarted after failure. For more information on how to define temporary tables, see *Turbo Charging Update Handler XML Configuration Guide*.

In addition, if the Availability Manager map has changed, the Update Handler in Incremental mode receives the list of segment groups whose processing was switched to the alternate database when the Update Handler was down. In this case, the data of the relevant customers is updated in the alternate database.

Full Mode Workflow

Full mode updates a mass of data from the customer management system to the target database.

The Update Handler requires an XML file as input. This file, defined by the implementation, includes all queries to be run for the specific execution (see *Turbo Charging Update Handler XML Configuration Guide*).

The structure and primary keys of all target tables that are populated by the Update Handler must appear in the ADJ1_GAT_DESC and ADJ1_GAT_FIELDS tables. This enables populating any required target table, core or customization, by modifying the profile XML file and adding the target table description in the generic applicative tables (GAT), without changing the code.

The Update Handler's sequence of actions for Full mode is as follows:

1. The Update Handler takes a leading query defined for Full mode from the implementation XML file.
2. The Leading Data Reader executes the query, transforms the output to Java objects, and transfers these objects to the Feeder.
3. The Feeder divides the leading data into groups, on the basis of the leading group calculation function (also described in the XML file). This function ensures that each group contains only information destined for a particular database. (Several groups may contain information for a specific database.)
4. When a group reaches a certain defined size (threshold), it is passed to the Ready for Work queue.
5. The Worker thread takes a group from the queue and begins to process it (extract, map, and load). It then commits the processed data to the target database.
6. When the Worker finishes working on this group, it takes the next group from the queue and begins to process it.
7. The Worker is not defined per database or group – it is completely generic and can work with any group and any database. This enables parallelism and load balancing.
8. The flow is finished when the Leading Data Reader finishes sending leading data to feeder and the worker threads finish processing.

Incremental Mode Workflow

In Incremental mode, the Update Handler updates data according to external requests, for example, transactions from the Transaction Broker, or files from Audit & Control or any other source. External requests are handled by implementing the relevant request adapter.

In this mode, the Update Handler handles the input that comes from sources other than the Amdocs Billing Customer Manager database. Usually, this preparation includes parsing the data for the Full flow.

The Update Handler's sequence of actions for Incremental mode is as follows:

1. The adapter draws a batch of records. If the TRB Adapter is used, it draws a batch of transactions published by the customer management system (Amdocs Billing Customer Manager). If multiple Transaction Broker subscribers are defined, a number of batches are drawn in parallel. The size of each batch is configurable.
2. The adapter converts the records into request objects. If the TRB Adapter is used, it parses the transactions in the batch, creates requests, and sends them to the Request Manager.
3. The Request Manager stores all the transactions, to monitor their status later. It also transfers them to LDR Preprocessing.
4. LDR Preprocessing takes the batch of requests and supplies the leading queries to the Leading Data Reader. The output of the leading queries is specific per request.
5. From this point, the Full flow is performed.
6. After the data is inserted into the target tables, the Activity Loader is invoked to create activities according to the implementation XML file.
7. The status of all requests is returned to the Request Manager.
8. The Request Manager returns the requests to the adapters.
9. Each adapter can implement its own error handling behavior. If the TRB Adapter is used, it marks the original transaction with an error.

Environment Variables

The process parameters of the Update Handler are stored in the ADJ1_CONF_SECTION_PARAM table, which is included in the reference database. There are two types of parameters:

- Parameters represented by environment variables in the database (for example, \${ADJUH_AC_DATA_GROUP}). Such parameters can be configured differently for each instance. For example, a parameter may be an environment variable out of the box, but it may be defined as a constant in customization.
- Hard-coded parameters in the database. These parameters must be modified in the database.

The following table shows additional environment variables for the Update Handler, including the Mark for Rerate and Reconciliation modes.

Name	Description	Location	Default Value
ABP_AJTUH_ROOT	The root directory that contains the operability directories for the Update Handler, such as logs, console, and so on	`\${ABP_BIN}/w.ini	\$HOME/var/m3g/projs/ajtuh
log4jFND	The minimum severity of Foundation (FND) messages that are printed to the log. Valid values: <ul style="list-style-type: none">■ ERROR■ WARNING■ DEBUG■ INFO	`\${HOME}/.w.ini.pre.env	ERROR
log4jJF	The minimum severity of Java Foundation (JF) messages that are printed to the log. Valid values: <ul style="list-style-type: none">■ ERROR■ WARNING■ DEBUG■ INFO	`\${HOME}/.w.ini.pre.env	ERROR
log4jRTA	The minimum severity of RTA messages that are printed to the log. Valid values: <ul style="list-style-type: none">■ ERROR■ WARNING■ DEBUG■ INFO	`\${HOME}/.w.ini.pre.env	ERROR
log4jUH	The minimum severity of Update Handler (UH) messages that are printed to the log. Valid values: <ul style="list-style-type: none">■ ERROR■ WARNING■ DEBUG■ INFO	`\${HOME}/.w.ini.pre.env	ERROR
RUN_BACKGROUND	Indicates whether the Update Handler must run in the background.		N

Parameters

The following input parameters must be set for the process to run properly.

Parameter	Description	Possible Values
profileFile	The name of the profile XML file, without a path to the local disk. All profile files are taken from the AUH1_PROFILES table. For more information, see the “Out-of-the-Box Profiles” section in this chapter.	The name of the selected implementation XML file.
runningMode	The mode in which the Update Handler is run.	<ul style="list-style-type: none"> ■ <i>FORCE_START</i> – If there is an open entry in the Update Handler Control (AUH1_CTRL) table (RUNNING_STATUS = R for Running), and the value of the UH_MODE field is opposite to the current mode (for example, incremental if the current mode is full), the Update Handler changes the status to E and starts from the beginning. ■ <i>Normal</i> – If there is an open entry in the Update Handler Control (AUH1_CTRL) table (RUNNING_STATUS = R for Running), and the value of the UH_MODE field is opposite to the current mode (for example, incremental if the current mode is full), the process does not start. ■ <i>Recovery</i> – The user indicates the need to perform recovery. The system cannot identify such need by itself. The Update Handler checks whether the profile matches the one currently listed in the Update Handler Control (AUH1_CTRL) table (RUNNING_STATUS = R for Running) with the same parameters as in the Update Handler Profile Context Parameters (AUH1_PROF_CTXT_PARAMS) table, changes the status to E (error) in the Update Handler Control (AUH1_CTRL) table, Checks in the Checkpoint (AUH1_CHECKPOINT) table where the previous process stopped and continues from there.
APPLICATION_ID	The name of the process instance. When a process instance is started from the Availability Manager, this name must match the one in the ADJ1_CONF_HIERARCHY table.	N/A
debugPort	An optional debug port that enables remote debugging of the process.	N/A

Out-of-the-Box Profiles

The following full mode implementations of the Update Handler are provided out of the box.

Profile Name	Purpose	Description	Recovery Mode	Delete Mode
UHImplFullModeGuiding	Update guiding data in Turbo Charging tables	These implementations of the Update Handler extract the data required for guiding to customer from the Amdocs Billing Customer Manager database and load it into the Turbo Charging Customer database.	N	N
UHImplFullModeGuidingDelete			N	Y
UHImplFullModeGuidingRecovery			Y	N
UHImplFullModeGuidingRecoveryDelete			Y	Y
UHImplFullModeRating	Update rating data in Turbo Charging tables	These implementations of the Update Handler extract the data required for rating from the Amdocs Billing Customer Manager database and load it into the Turbo Charging Customer database.	N	N
UHImplFullModeRatingDelete			N	Y
UHImplFullModeRatingRecovery			Y	N
UHImplFullModeRatingRecoveryDelete			Y	Y
UHImplFullModeRatingDeleteMultiEgi		This flavor of the Full Rating Delete implementation of the Update Handler supports multiple Event Group Items (EGIs) in addition to the general functionality.	N	Y
UHImplFullModeRatingDelete_AGR_HIST			N	Y
UHImplManMark	Manually mark for rerating (for more information, see the “ADJ1UHMARK4RE – Update Handler in Mark for Rerate” chapter)	This implementation of the Update Handler retrieves a list of parameters (such as a resource, a subscriber, or a customer) from the AUH1_REQUESTS table and marks each subscriber or customer for rerate in the APE1_SUBSCRIBER_RERATE table.	N	N
UHImplManMarkByOffer			N	N
UHImplFullModeCUG	Update CUG data in Turbo Charging tables	These implementations of the Update Handler extract the Closed User Groups (CUG)* data required for rating from the	N	N
UHImplFullModeCUGDelete			N	Y

Profile Name	Purpose	Description	Recovery Mode	Delete Mode
UHImplFullModeCUGRecovery	(for more information, see the “ADJ1UHCUG – Update Handler in CUG Mode” chapter)	Amdocs Billing Customer Manager database and load it into the Turbo Charging Customer database.	Y	N
UHImplFullModeCUGRecoveryDelete			Y	Y
UHImplMiniFullGuiding	Update guiding data in Turbo Charging tables (for more information, see the “ADJ1UHMINFULL – Update Handler in Mini-Full Mode” chapter)	This implementation of the Update Handler retrieves a list of parameters (such as a resource, a subscriber, or a customer) from the AUH1_REQUESTS table. Then it extracts the data required for guiding to customer from the Amdocs Billing Customer Manager database and loads it into the Turbo Charging Customer database.	N	Y
UHImplMiniFullRating	Update rating data in Turbo Charging tables (for more information, see the “ADJ1UHMINFULL – Update Handler in Mini-Full Mode” chapter)	This implementation of the Update Handler retrieves a list of parameters (such as a resource, a subscriber, or a customer) from the AUH1_REQUESTS table. Then it extracts data required for rating from the Amdocs Billing Customer Manager database and loads it into the Turbo Charging Customer database.	N	Y
UHImplFullModeOutcollectGuidingDelete	Update guiding data in Turbo Charging tables	This implementation of the Update Handler retrieves a list of outcollect customers with the details relevant to Guiding to Customer (such as provider ID and bill cycle) from the AGD1_RESOURCES_REF table, which is populated by the Turbo Charging Configurator for all cycles. Then the implementation loads this information to the corresponding AGD1_RESOURCES table for the cycle.	N	Y

Profile Name	Purpose	Description	Recovery Mode	Delete Mode
UHImplFullModeOutcollectRatingDelete	Update rating data in Turbo Charging tables	This implementation of the Update Handler retrieves a list of outcollect customers with the details relevant to Guiding to Service (such as provider ID and provider agreement billing offers) from the APE1_SUBSCR_DATA_REF, APE1_SUBSCR_OFFERS_REF, and APE1_SUBSCR_PARAMS_REF tables, which are populated by the Turbo Charging Configurator for all cycles. Then it loads this information to the corresponding APE1_SUBSCR_DATA, APE1_SUBSCR_OFFERS, and APE1_SUBSCR_PARAMS tables for the cycle.	N	Y

- * The CM_USER_GROUPS table of Amdocs Billing Customer Manager includes lists of resources per group ID. All resources in the list are usually entitled for reduced rates or discounts on tariffs. In the CUG mode, the Update Handler updates the APE1_USER_GROUP_MEM table of Turbo Charging with this information from Amdocs Billing Customer Manager according to the implementation.

The following incremental mode implementations of the Update Handler are provided out of the box.

Profile Name	Purpose	Description	Recovery Mode	Delete Mode
UHImplBillUndoMark	Mark for rerating by Invoicing request (for more information, see the “ADJ1UHMARK4RE – Update Handler in Mark for Rerate” chapter)	This implementation of the Update Handler retrieves input files from Audit & Control. Then, for each combination of customer, cycle code, and cycle instance in the file, it inserts a request into the APE1_SUBSCRIBER_RERATE table.	N	N
UHImplIncModeGuiding	Update guiding data in Turbo Charging tables	This implementation of the Update Handler listens to external requests and extracts the data required for guiding to customer or CUG data required for rating from the Amdocs Billing Customer Manager database and loads it into the Turbo Charging Customer database.	N	Y

Profile Name	Purpose	Description	Recovery Mode	Delete Mode
UHImplIncModeRating	Update rating data in Turbo Charging tables	This implementation of the Update Handler listens to external requests and extracts the data required for rating from the Amdocs Billing Customer Manager database and loads it into the Turbo Charging Customer database.	N	Y
UHImplIncModeRating_AGR_HIST		This flavor of the Incremental Rating implementation of the Update Handler supports offer history in addition to the general functionality.	N	Y
RCN_Implementation	Compare data in two sources and synchronize the data between the sources (for more information, see the “ADJ1RCN – Update Handler in Reconciliation Mode” chapter)	This implementation of the Update Handler extracts data from the database tables and AIMOS, compares it, and displays the comparison results in the results table. The compare and update requests residing in the ADJ1_REQUESTS table trigger the process.	N	N

For more information on how to modify these profile files or create new ones, see *Turbo Charging Update Handler XML Configuration Guide*.

Configuration Parameters

The configuration parameters (properties) of the Update Handler are defined in the database.

The following table shows the performance tuning (configuration) parameters of the Update Handler. These parameters are defined in the ADJ1_CONF_SECTION_PARAM table using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*).

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
AC	ac.connection	maxConnectAttempts	3	The maximum number of attempts to connect to Audit & Control, if no group was retrieved.
AC	ac.init	checkBalanceInd	N	Ensures that all the records have been processed, and none of them were lost between the components.
AC	ac.init	DataGroup	AC_DATA_GROUP	The group name for the data contained in the file.
AC	ac.init	maxGroupSize	100	The maximum group size to be retrieved by the Get Next Group API.
AC	ac.init	minGroupSize	0	The minimum group size to be retrieved by the Get Next Group API.
AC	ac.init	ProgramName	<ProgramName>	The program that is to process the file.
AC	ac.init	recycleType	AR	Indicates whether only new records, only in-use records, or both types of records must be processed.
AC	ac.init	retrieveType	L	Used to select Audit & Control records from the AC1_CONTROL table, this parameter indicates whether to filter the records by the DATA_GROUP attribute using the “like” keyword or the “=” sign (the exact group name).
AC	ac.sleepTime	init	1000	The initial sleep time for Audit & Control, if no group was retrieved.
AC	ac.sleepTime	max	30000	The maximum sleep time for Audit & Control, if no group was retrieved.
ADJ	config	AVM_ENABLED	Y	Indicates whether Availability Manager is used when running the process.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
AJTFND	fnd.connection.test	query	SELECT 1 FROM DUAL	The query that checks the connection.
AJTFND	fnd.db	fetchSize	50	The size of the bulk that can be fetched.
CM	cm.date.time	format	yyyy-MM-dd'T'HH:mm:ss	The date format as it appears in Amdocs Billing Customer Manager.
TRB	trb.group	number	1	The group number in the Transaction Broker. This parameter must also be defined for each Update Handler process instance in Incremental mode.
TRB	trb.member.code	guiding	UHG	The member code of the Update Handler Guiding process (which populates the resource information) in the Transaction Broker.
TRB	trb.member.code	rater	UHR	The member code of the Update Handler Rating process (which populates the customer information) in the Transaction Broker.
TRB	trb.member.id	guiding	3016	The member ID of the Update Handler Guiding process in the Transaction Broker.
TRB	trb.member.id	rater	3015	The member ID of the Update Handler Rating process in the Transaction Broker.
UH	fnd.rac	mode	true	The way in which Java creates a connection to the database.
UH	uh.checkpoint	sleepTime	100000	The sleep time of the Checkpoint Manager.
UH	uh.db.temp.tables	prefix	AUH1	The prefix of the Update Handler temporary tables in the database.
UH	uh.db.Threshold	error	1000	The maximum number of errors for a query. When the process reaches this number of errors, it sends an alert to the Availability Manager and stops.
UH	uh.dummy.fatal.sql.exception	errorCodes	600,7445,3113	The list of dummy fatal exception codes. In the case of a dummy fatal exception, the transaction fails, but the Update Handler does not shut down.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
UH	uh.error.alert	threshold	100	The maximum number of errors. When the process reaches this number of errors, it sends an alert to the Availability Manager and continues running.
UH	uh.eventBuffer	byteOrder	2	<p>The order of bytes in the event buffer. This property depends on the architecture of the Event Server machine. Can be:</p> <ul style="list-style-type: none"> ■ 1 – Big Endian (the low-order byte of the number is stored in memory at the lowest address, and the high-order byte at the highest address) ■ 2 – Little Endian (the high-order byte of the number is stored in memory at the lowest address, and the low-order byte at the highest address)
UH	uh.eventBuffer	charset	UTF-8	The character set to be used for conversion of strings to byte arrays.
UH	uh.fatal.sql.exception	errorCodes	942, 913, 904, 980, 1017	<p>The error code. Can be:</p> <ul style="list-style-type: none"> ■ 942 – Table or view does not exist. ■ 913 – Too many values. ■ 904 – Invalid column name. ■ 980 – Synonym translation is no longer valid. ■ 1017 – Invalid username/password; logon denied.
UH	uh.feeder	groupSizeThreshold	3000	The size of the group that is eventually handled by the Worker. This value is very important and must be carefully tuned.
UH	uh.org.xml.sax	driver	org.apache.xerces.parsers.SAXparser	<p>The class name of the parser. This property is used only if the parser has failed to instantiate. Normally, this property is not populated.</p> <p><i>Example:</i> <code>org.xml.sax.driver=org.apache.xerces.parsers.SAXParser</code> <code>Java parser – javax.xml.parsers.SAXParser</code></p>
UH	uh.pending.requests	threshold	100	The maximum number of requests that the system supports.
UH	uh.socket	receiveBufferSize	1048576	The size of the receive buffer.
UH	uh.socket	sendBufferSize	524288	The size of the send buffer.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
UH	uh.socket	soTimeOut	10000	The socket time-out.
UH	uh.thread.pool	ldrCoreSize	4	<p>The number of Leading Data Reader (LDR) threads that can run in parallel.</p> <p>This parameter is relevant for Incremental mode, where each LDR thread handles a different request batch.</p>
UH	uh.thread.pool	reqProviderCoreSize	2	Defines the number of provider threads that can run in parallel.
UH	uh.thread.pool	workerCoreSize	4	The number of Worker threads that can run in parallel.
UH	uh.time.measurements	numberOfRequests	-1	<p>Limits the number of requests for the process. For Non-Functional Test (NFT) use.</p> <p>In production, the value of this parameter is -1.</p>
UH	uh.validate.subscriber	param	N	Indicates whether the Update Handler must validate that the subscriber parameter values in the APE1_SUBSCR_PARAMS table match their type. If this parameter is set to 'Y', and a value does not match its type (for example, the value is a number, but the PARAM_VALUE column contains a String representation of this number), the Update Handler throws an exception and ignores the row.
UHI	uh.character.encoding	class	ISO-8859-1	The encoding that the Update Handler uses.
UHI	uh.complete.messages	ratio	3	The number of completed messages to handle before handling new messages.
UHI	uh.fileAdapter	FileLevelErrorPolicy	SkipFile	<p>The error handling policy in the event of a file-level error. Can be:</p> <ul style="list-style-type: none"> ▪ AbortProcess ▪ SkipFile
UHI	uh.fileAdapter	fullInputFileName	N/A	The input file name used only when uh.fileAdapter.mode=SpecificFile.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
UHI	uh.fileAdapter	maxBufferSize	1000	The maximum buffer size that is read each time data is extracted from a file.
UHI	uh.fileAdapter	mode	SpecificFile	The File Adapter mode. Can be: <ul style="list-style-type: none"> ■ AC – In this mode, the File Adapter takes files from Audit & Control. ■ <i>SpecificFile</i> – In this mode, the File Adapter takes files directly from disk according to the fullInputFileName property.
UHI	uh.fileAdapter	RecordLevelErrorPolicy	SkipRecord	The error handling policy in the event of a record-level error. Can be: <ul style="list-style-type: none"> ■ SkipRecord ■ AbortProcess
UHI	uh.general	batchSize	500	The maximum number of transactions taken from the Transaction Broker in a single bulk.
UHI	uh.mode	deamon	Y	Indicates whether the Incremental Update Handler is run in daemon mode.
UHI	uh.num.continuous.access.with.no.trx	returned	1000	When there are no transactions from the Transaction Broker, the number of retries before going to sleep.
UHI	uh.request.manager	objectsThreshold	10000	The maximum number of requests that the Request Manager can handle.
UHI	uh.request.messages	ratio	10	The number of request messages to handle before handling completed messages.
UHI	uh.requests.bulk.extract	size	5000	The bulk size for a number of requests taken from the database.
UHI	uh.thread.pool	reqParserCoreSize	4	The number of request Parser threads that can run in parallel.
UHI	uh.thread.pool	reqProviderCoreSize	2	Defines the number of provider threads that can run in parallel.
UHI	uh.thread.sleep.time	init	1000	The minimum sleep time when the process receives no input.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
UHI	uh.thread.sleep.time	max	30000	The maximum sleep time when the process receives no input.
UHI	uh.trb	errorTimeThreshold	30000	The time frame used to determine the threshold of errors after which the Update Handler in Incremental mode sends an alert to the Availability Manager. The number of errors within this time frame that triggers an alert is defined by the maxErrors parameter.
UHI	uh.trb	maxErrors	1000	The maximum number of errors in the period of time defined by the errorTimeThreshold parameter. When this threshold of errors is reached, and the AVM_ENABLED parameter is set to 'Y', the Update Handler sends an alert to the Availability Manager and continues.
UHI	uh.trb.bulk.waiting.max.cache	size	1000	The cache size for waiting messages from the Transaction Broker.
UHI	uh.trb.internal.cache	size	1000	The internal cache size of the Transaction Broker client.

For each process instance, an entry with the name of the process instance must be created in the ADJ1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
UHF123	UHF
UHI456	UHI

Any parameter to be overridden at the process instance level must be added to the ADJ1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
UHF123	uh.feeder	groupSizeThreshold	1000

For more information on parameters and the properties mechanism, see the “Configuration Parameter Mechanism” appendix.

Database Connections

The process connects to the following tables during processing:

- Various tables in the Turbo Charging Usage, Reference, and Application areas:
 - ADJ1_GAT_DESC
 - ADJ1_GAT_FIELDS
 - AGD1_RESOURCES
 - AGD1_RESOURCES_REF (Outcollect profiles only)
 - AGD1_UH_TEMP_CUG (Synchronous mode only)
 - AGD1_UH_TEMP_INC (Synchronous mode only)
 - AGD1_UH_TEMP_RESOURCES (Synchronous mode only)
 - AGD1_UH_TEMP_SA_CUG (Synchronous mode only)
 - APE1_CUST_CYCLE_HISTORY
 - APE1_SUBSCR_DATA
 - APE1_SUBSCR_DATA_REF (Outcollect profiles only)
 - APE1_SUBSCR_OFFERS
 - APE1_SUBSCR_OFFERS_REF (Outcollect profiles only)
 - APE1_SUBSCR_PARAMS
 - APE1_SUBSCR_PARAMS_REF (Outcollect profiles only)
 - APE1_TECH_EVENTS
 - APE1_UH_TEMP_INC (Synchronous mode only)
 - APE1_UH_TEMP_SUBSCRIBER (Synchronous mode only)

- APE1_USER_GROUP_MEM
- AUH1_CHECKPOINT
- AUH1_CTRL
- AUH1_CTRL_TEMP_TABLES
- AUH1_PROF_CTXT_PARAMS
- AUH1_PROFILES
- AUH1_REQUESTS
- AUH1_UPDATES
- Tables in the Turbo Charging Configuration area
- Amdocs Billing Customer Manager tables (Oracle):
 - Customer database (depending on implementation)
- Transaction Broker tables:
 - Turbo Charging client area (TRB1_SUB_LOG)

Admin Commands

The Update Handler supports the following admin commands. For information on how to execute the commands, see the “Handling Admin Commands” section in the “High Availability” chapter.

Command Name	Command Value	Description
REFRESH_ALL	1	The process performs all supported refresh actions.
REFRESH_REFERENCE	2	The process refreshes all reference data.
SUSPEND	7	The process waits until the running threads complete their work and then enters suspended mode.
RESUME	8	The process exits suspended mode and resumes its work.
GRACEFULL_SHUTDOWN	100	The process shuts down gracefully.

Examples:

ADJ1_Send_Admin_Command_Sh hpbl820 UHF_GD 1110 1

ADJ1_Send_Admin_Command_Sh hpbl820 UHF_GD 1110 7

Distribution

The Update Handler inserts the output as required to each of the relevant databases.

Recovery Instructions

This section describes the steps to be taken if the Update Handler process fails in Full or Incremental mode.

Full Mode

To enable the Update Handler to recover after failure, the leading query must be sorted. Sorting according to the database index is recommended.

In addition, the profile XML file must specify that the implementation supports recovery. For more information, see *Turbo Charging Update Handler XML Configuration Guide*.

Checkpoints

At each defined interval of time (configurable), a checkpoint is performed. During a checkpoint:

1. The Leading Data Reader is suspended, and the Feeder is emptied of all groups.
2. The groups are moved to the Ready for Work queue (regardless of size) and are processed as usual. The Update Handler Updates (AUH1_UPDATES) control table contains the number of leading data objects that have been processed, per leading query.
3. After this is finished, the Leading Data Reader resumes and continue as usual until the next checkpoint.



Note: Naturally, use of checkpoints has a negative impact on performance because checkpoints require system suspension.

Incremental Mode

In the event of failure, each request adapter is responsible for implementing its own recovery mechanism.

TRB Adapter

Some of the transactions received from the Transaction Broker are completed, and others remain in process. In this case, when the system restarts, the TRB Adapter extracts all transactions that have not been completed from the Transaction Broker, and receives those that were being processed when the failure occurred.

File Adapter

When working with Audit & Control, the File Adapter first extract files in IU (In Use) status. This ensures that if the process crashes, the files that were open at the time of failure are handled again.



Note: For these reasons, there is no need to perform checkpoints in Incremental mode.

Recovery Flow

When the Update Handler process is started with the Recovery flag (only for the Full mode), it first checks whether the profile matches the one currently listed in the Update Handler Control (AUH1_CTRL) control table (status=R for Running) with the same parameters as in the Update Handler Profile Context Parameters (AUH1_PROF_CTXT_PARAMS) table. If it does, the Update Handler:

1. Changes the status to E (error) in the Update Handler Control (AUH1_CONTROL) table
2. Checks the Checkpoint (AUH1_CHECKPOINT) table to find out where the previous process stopped
3. Continues from the point at which it stopped

This implies the Update Handler skips the leading queries and leading data that have already been processed.

6 ADJ1UH – Update Handler in Synchronous Mode

This chapter describes the Update Handler in Synchronous mode.

Description

Some changes in the customer data entities mastered Amdocs Billing Customer Manager must be synchronously replicated in Turbo Charging, in the internal Turbo Charging Customer database and the shared memory of the Event Server.

When an Amdocs Billing Customer Manager activity is marked for a synchronous update, the TRB Publish API called from Amdocs Billing Customer Manager invokes the Update Handler in Synchronous mode.

The Update Handler in Synchronous mode is deployed on an application server (J2EE) and provides two APIs:

- Rating API, for updating the relevant subscriber data
- Guiding API, for updating the relevant resource data

During installation, one of the following modes of work can be selected:

- *Regular* – The extract and load functionality of the Rating API is provided on the same application server. In this mode:
 - Only the AJUH module must be deployed on the application server.
 - The AJUH module and Amdocs Billing Customer Manager must be deployed on the same application server.
- *Split* – The logic of the Rating API is split into two parts: Extract and Load. In this mode:
 - The AJTUHEXTRACT module must be deployed on the same application server as Amdocs Billing Customer Manager.
 - The AJTUHLOAD module must be deployed on another application server that is located on a different site, close to the Usage database.
 - The AJUH mode must not be deployed.

The invoked Update Handler API receives the transaction XML file as input directly from Amdocs Billing Customer Manager (bypassing the Transaction Broker Engine, via the TRB Publish API) and creates or updates the relevant subscriber or resource data in the Turbo Charging Customer database. The data is committed to the database only after sending an indication to the Event Server to delete the existing subscriber or resource data from the shared memory. The Event Server loads the updated data from the internal Turbo Charging Customer database into its shared memory in the following cases:

- When an event arrives for the subscriber
- When DB2E updates the Event Server with newer data from the database

After Update Handler in Synchronous mode has processed the transaction, Amdocs Billing Customer Manager commits the transaction, and the data of all the components involved in the flow is inserted to the database at the same time. If the transaction fails in Turbo Charging, it is rolled back in all the components, including Amdocs Billing Customer Manager. For more information about the end-to-end flow of synchronized customer activities, see *Amdocs Billing Introduction*.

The Update Handler in Synchronous mode uses the same implementation logic as the Update Handler in Incremental mode. For more information, see the “[ADJ1UH – Update Handler in Full and Incremental Mode](#)” chapter.

For guidelines on the implementation in the Rating Logic Configurator, see *Rating Logic Configurator Implementation Best Practices*.

Process Type

Daemon

Run Frequency

The Update Handler in Synchronous mode is activated when the application server on which it is deployed is started. It handles incoming requests from Amdocs Billing Customer Manager synchronously.

Activation

To activate the Update Handler in Synchronous mode, start the application server on which the Update Handler module that exposes the Rating and Guiding APIs is deployed.

Shutdown

To stop the Update Handler in Synchronous mode, stop the application server on which the Update Handler module is deployed.

Preceding Processes

The following process must run successfully before this process is activated:

- *Implementation Compiler* – The Implementation Compiler creates an SQL file, which contains the data of the ADJ1_SUBSCR_ATTRIBUTES table. The table contains the mapping between the generated attribute IDs, and the attribute names in the Rating Logic Configurator and in the Amdocs Billing Customer Manager. The Update Handler takes the attribute name in Amdocs Billing Customer Manager and populates the database with the corresponding ID.

Affected Applications

The Update Handler in Synchronous mode affects the following processes:

- *Update Handler in Full mode* – The Update Handler in Full mode and the Update Handler in Synchronous mode cannot run in parallel.
- *Update Handler in Mini-Full mode* – The Update Handler in Mini-Full mode and the Update Handler in Synchronous mode can run in parallel. In this case, if customized profiles for the Mini-Full mode are used, the Delete queries must have an additional WHERE clause so that only the records that are older than the number of seconds specified as a context parameter are deleted. For more information, see *Turbo Charging Update Handler Implementation Best Practices* and *Turbo Charging Update Handler XML Configuration Guide*.
- *Event Server* – After receiving a request from the Rating or Guiding API of the Update Handler, the Event Server marks that the requested subscriber or resource data that already exists in shared memory must be updated with newer data from the Turbo Charging Customer database. This mark can be viewed using the AIMOS MetaInfo tool. For more information, see the “[metainfo – AIMOS MetaInfo](#)” chapter.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Update Handler log files*:
ADJ1UH.<Date>_<Time>.log

Example:

ADJ1UH.20111018_1357.log

Location

The log files are located in the following directory:

`$(APP_DOMAIN_HOME)/$(APP_SERVER_NAME)/logs/`

Contents

The log files contain the following:

- *Update Handler log files* – Information regarding any errors that occurred during the Update Handler run. If the Update Handler ran in debug mode, the logs also contain all debugging messages.
- *Application server log files* – Information regarding any errors that occurred during the application server start-up.

Input Files

This section describes the input files.

Update Handler Implementation File

The input to the process is the implementation profile XML file of the Update Handler. In the out-of-the-box implementation, the profiles of the Incremental mode are used.

Location

The process extracts the profile XML file from the AUH1_PROFILES table. The name of the profile XML file to extract must be specified in the profilesName parameter in the ADJ1_CONF_SECTION_PARAM table. If a customized profile file is used, it must include the Delete Memory query as described in *Turbo Charging Update Handler Implementation Best Practices* and be specified in the profilesName parameter in the ADJ1_CONF_SECTION_PARAM table.

Contents

The profile XML file describes the actions that the Update Handler in Synchronous mode must perform. For a full description, see *Turbo Charging Update Handler XML Configuration Guide*.

The following profiles are used out of the box.

Profile Name	Purpose	Description	Recovery Mode	Delete Mode
UHImplIncModeGuiding	Update guiding data in Turbo Charging tables	This implementation of the Update Handler listens to external requests and extracts the data required for guiding to customer or CUG data required for rating from the Amdocs Billing Customer Manager database and loads it into the Turbo Charging Customer database.	N	Y
UHImplIncModeRating	Update rating data in Turbo Charging tables	This implementation of the Update Handler listens to external requests and extracts the data required for rating from the Amdocs Billing Customer Manager database and loads it into the Turbo Charging Customer database.	N	Y
UHImplFullModeRating Delete_AGR_HIST	Update rating data in Turbo Charging tables	This flavor of the Full Rating Delete implementation of the Update Handler supports offer history in addition to the general functionality.	N	Y

Amdocs Billing Customer Manager Transaction Files

Amdocs Billing Customer Manager transaction files contain the transaction that Amdocs Billing Customer Manager sends to the Update Handler in Synchronous mode.

Location

The Update Handler in Synchronous mode receives the transactions from Amdocs Billing Customer Manager via the TRB Publish API.

Contents

Amdocs Billing Customer Manager transaction files contain various data from Amdocs Billing Customer Manager, for example, New Customer, Change Offer, or Change Price Plan activities.

Output Files

The Update Handler in Synchronous mode does not produce any output files. The output of the module is the ‘delete’ indication in the Event Server and the data in the resource, subscriber, and Update Handler control tables. For more information, see *Turbo Charging Data Model*.

Flow

The following subsections describe the high-level workflows of the Update Handler in Synchronous mode.

Initialization

The Update Handler in Synchronous mode is initialized upon the start-up of the application server.

In contrast to the Update Handler in Incremental mode, the Update Handler in Synchronized mode does not create temporary tables during the initialization stage. Instead, it uses permanent physical tables that have the same structure as the temporary tables defined in the profile file. After the process run, these tables are cleared.

In addition, if the Availability Manager map has changed, the Update Handler in Synchronous mode receives the list of segment groups whose processing was switched to the alternate database when the Update Handler module was down. In this case, the data of the relevant customers is updated in the alternate database.

API Workflow

In Synchronous mode, the Update Handler updates data according to the transaction received from Amdocs Billing Customer Manager.

Regular Flow

If the split functionality is not used, the Rating and Guiding APIs of the Update Handler in Synchronous mode perform the following actions:

1. Receive an XML file that represents an Amdocs Billing Customer Manager transaction directly from Amdocs Billing Customer Manager via the TRB Publish API (bypassing the Transaction Broker Engine).
2. According to the configuration of the Update Handler in Synchronous mode (see the “ADJ1UH – Update Handler in Full and Incremental Mode” chapter):
 - a. Extract the data from the Amdocs Billing Customer Manager database:
 - The Guiding API extracts resource data.
 - The Rating API extracts subscriber data.
 - b. Map the extracted data to the output structure according to the configuration.
 - c. Create subscriber activities data.
3. According to the customer and resource information, route the data as follows:
 - *Guiding API* – To the relevant database that contains the AGD1_RESOURCES table. This database can be a home or an alternate database.
 - *Rating API* – To the relevant database that contains the APE_ subscriber tables. This database can be a home or an alternate database.
4. Load the data to the relevant database.
5. Route the data to the relevant Event Server according to the customer or resource information.
6. Send the delete request to the Event Server. The Event Server marks the subscriber or resource data in its shared memory for deletion. The newest data is updated in the shared memory of the Event Server by DB2E or by a related event coming from the network.
7. Return the status of the transaction in Turbo Charging to Amdocs Billing Customer Manager. Amdocs Billing Customer Manager commits the synchronous transaction, and only then is all the data (in Amdocs Billing Customer Manager and in Turbo Charging) actually inserted into the database.

Split Flow

If the split functionality is used, the Rating API is split into two parts: Extract and Load. This functionality enables deploying the Load logic on a different application server than the Extract logic, which must reside on the same application server as Amdocs Billing Customer Manager.

Extract Logic

The Extract part performs the following flow:

1. Receives an XML file that represents an Amdocs Billing Customer Manager transaction directly from Amdocs Billing Customer Manager via the TRB Publish API (bypassing the Transaction Broker Engine).
2. According to the configuration of the Update Handler (see the “ADJ1UH – Update Handler in Full and Incremental Mode” chapter):
 - a. Extracts the subscriber data from the Amdocs Billing Customer Manager database
 - b. Maps the extracted data to the output structure according to the configuration
 - c. Creates subscriber activities data
3. Routes the data to the application server on which the Load API is deployed. The application server is located according to the URL specified in the lookupName parameter that is sent in the API. The lookupName parameter must be defined as a session argument of the OP1URLABP task in the GN1_TASK_CONNECT table (including URL_ as a prefix). The connection code must be defined by the Infra team and include the correct URL of the application server. For more information, see the “Important Remarks” section in this chapter.
4. Invokes the Load API on the other application server.

Load Logic

The Load part performs the following flow:

1. According to the customer information, routes the data to the relevant database that contains the APE_ subscriber tables.
2. Loads the data to the relevant database.
3. Routes the data to the relevant Event Server according to the customer information.
4. Sends the delete request to the Event Server. The Event Server marks the subscriber or resource data in its shared memory for deletion. The newest data is updated in the shared memory of the Event Server by DB2E or by a related event coming from the network.
5. Returns the status of the transaction in Turbo Charging to Amdocs Billing Customer Manager. Amdocs Billing Customer Manager commits the synchronous transaction, and only then is all the data (in Amdocs Billing Customer Manager and in Turbo Charging) actually inserted into the database.

Environment Variables

The Update Handler in Synchronous mode uses the same environment variables as the Update Handler in Incremental mode. For more information, see the “Environment Variables” section in the “ADJ1UH – Update Handler in Full and Incremental Mode” chapter.

Configuration Parameters

The Update Handler in Synchronous mode requires the same configuration parameters as the Update Handler in Incremental mode (for more information, see the “Configuration Parameters” section in the “ADJ1UH – Update Handler in Full and Incremental Mode” chapter). However, the default values of some parameters in Synchronous mode are different, as shown in the following table.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
UHEJB	uh.db.temp.tables.guiding	prefix	AGD1	<p>The prefix of the physical tables that match the structure of the temporary tables defined in the profile file. In Synchronous mode, these matching physical tables are used instead of temporary tables.</p> <p>For example, if a guiding profile defines a temp table called RESOURCE_INFO, and if the value of this parameter is set to ‘AGD1’, the database must include a physical table called AGD1_RESOURCE_INFO.</p>
UHEJB	uh.db.temp.tables.rating	prefix	APE1	<p>The prefix of the physical tables that match the structure of the temporary tables defined in the profile file. In Synchronous mode, these matching physical tables are used instead of temporary tables.</p> <p>For example, if a rating profile defines a temp table called SUBSCR_INFO, and if the value of this parameter is set to ‘APE1’, the database must include a physical table called APE1_SUBSCR_INFO.</p>
UHEJB	uh.connection	maxConnectAttempts	-1	The number of attempts to reconnect to the Event Server if the first attempt has failed. The value of ‘-1’ indicates no reconnection attempts.
UHEJB	uh.db.Threshold	error	1	The maximum number of errors that causes a transaction to fail in the Update Handler.
UHEJB	uh.fatal.sql.exception	errorCodes	ALL	The list of SQL error codes that cause a transaction to fail. In Synchronous mode, all error codes are fatal.
UHEJB	uh.general	profilesName	UHImplIncMod eRating, UHImplIncMod eGuiding	The names of the profile XML files that are used in Synchronous mode.
UHEJB	uh.renovator.sleepTime	init	1000	The initial interval (in milliseconds) between reconnection attempts. This interval is increased up to the value of the uh.renovator.sleepTime.max parameter when failures occur.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
UHEJB	uh.renovator.sleepTime	max	30000	The maximum interval (in milliseconds) between reconnection attempts.
UHEJB	uh.sync.event	degradationMode	N	Indicates whether the Update Handler in Synchronous mode must work in degraded mode. In degraded mode, if an error or timeout occurs while the Update Handler is sending an event to the Event Server, the error or the timeout is ignored, and the Update Handler continues with the API. In regular, non-degraded mode, if the Update Handler cannot send events to the Event Server, the API fails, and the database changes are rolled back.
UHEJB	uh.timeout.response	threshold	10000	The period of time (in milliseconds) for which the Update Handler in Synchronous mode must wait for a response from the Event Server.

Database Connections

The Update Handler in Synchronous mode connects to the database using the data sources defined on the application server.

To create these data sources, the OP2J2EESERVER task in the GN1_TASK_CONNECT table is used. Each connection code for the task name in this table represents a database. The details of each database connections (including the relevant connection parameters) are specified in the GN1_CONNECT_PARAMS table.

Each entry in the GN1_CONNECT_PARAMS table includes the relevant connection parameters and their properties in the CONN_PARAMS field, in XML format.

Example:

```
<Parameters>
  <Parameter Name="ADJAPPL_DB_USER" Value="ELODB16" IsSensitive="false"/>
  <Parameter Name="ADJAPPL_DB_PASSWORD" Value="ELODB16" IsSensitive="true"/>
  <Parameter Name="ADJAPPL_DB_INSTANCE" Value="RM810IA" IsSensitive="false"/>
  <Parameter Name="ADJAPPL_DB_HOST" Value="illin418" IsSensitive="false"/>
  <Parameter Name="ADJAPPL_DB_TYPE" Value="ORA" IsSensitive="false"/>
  <Parameter Name="ADJAPPL_DB_TRX_DS_JNDI" Value="j2ee.jdbc trxDataSource" IsSensitive="false"/>
  <Parameter Name="ADJAPPL_DB_TRX_DS_NAME" Value="ABPTrxDataSource" IsSensitive="false"/>
  <Parameter Name="ADJAPPL_DB_JDBC_XA_DRIVER" Value="oracle.jdbc.xa.client.OracleXADataSource" IsSensitive="false"/>
  <Parameter Name="ADJAPPL_DB_JDBC_XA_POOLNAME" Value="ABP_CORE_XA_POOL" IsSensitive="false"/>
  <Parameter Name="ADJAPPL_DB_NON_TRX_DS_JNDI" Value="j2ee.jdbc.regularDataSource" IsSensitive="false"/>
  <Parameter Name="ADJAPPL_DB_NON_TRX_DS_NAME" Value="ABPRegularDataSource" IsSensitive="false"/>
  <Parameter Name="ADJAPPL_DB_JDBC_POOLNAME" Value="ABP_CORE_XA_POOL" IsSensitive="false"/>
  <Parameter Name="ADJAPPL_DB_JDBC_DRIVER" Value="oracle.jdbc.driver.OracleDrivers" IsSensitive="false"/>
</Parameters>
```

One of the properties specified in the database connection details is the JNDI name. If the same JNDI name is defined for two different connection codes only one data source is created on the application server to serve them both.

If the Update Handler in Synchronous mode must work with an additional Customer Usage database, a new data source must be added to the application server.

To add a new data source on the application server:

1. Add a new record to the IDAT of the GN1_TASK_CONNECT with the OP2J2EESERVER task name and the relevant connection code (for example, ADJCUSTUSG3).
2. Ask the Infra team to add a new record with the relevant connection code to the GN1_CONNECT_PARAMS table. The parameters of the new connection code must specify the Customer Usage database.



Note: The Update Handler in split Synchronous mode supports a maximum of 999 Customer Usage databases.

For specific tables to which the Update Handler in Synchronous mode connects, see the “Database Connections” section in the “ADJ1UH – Update Handler in Full and Incremental Mode” chapter.

Admin Commands

The Update Handler in Synchronous mode supports the following admin commands. Because the Update Handler module resides on the application server, the operator can send these commands only via Amdocs Monitoring & Control and not via the command line.

Command Name	Description
REFRESH_ALL	The Update Handler module performs all supported refresh actions.
REFRESH_REFERENCE	The Update Handler module refreshes all reference data.

Distribution

The Update Handler in Synchronous mode inserts the updated data as required to each of the relevant databases.

Recovery Instructions

If failure occurs, the transaction is returned to the calling application with an error. This transaction is rolled back in Amdocs Billing Customer Manager. The transaction is not resent automatically. The client that invoked the Amdocs Billing Customer Manager API must invoke it again.

Important Remarks

If the logic of the Rating API is split into the Extract and Load logic, the Extract logic must call the Load logic on a different application server. The application server is located according to the URL specified in the lookupName parameter that is sent in the API. The lookupName parameter must be defined as a session argument of the OP1URLABP task in the GN1_TASK_CONNECT table (including URL_ as a prefix). The connection code must be defined by the Infra team and include the correct URL of the application server.

Example:

GN1_TASK_CONNECT

TASK_NAME	DB_CODE	RUN_MODE	SESSION_ARG	CONNECT_CODE
OP1URLABP	M3G	F	URL_UHL1	ELNUHL1URL
OP1URLABP	M3G	F	URL_UHL2	ELNUHL2URL

GN1_CONNECT_PARAM

CONNECT_CODE	SERVER_TYPE	HOST_NAME	CONNECTION_PARAMS
ELNUHL1URL	ORA	illin440	(HUGECLLOB)
ELNUHL2URL	ORA	illin440	(HUGECLLOB)

```
<Parameters>
  <Parameter Name="URL" Value="illin440:13487" IsSensitive="false"/>
  <Parameter Name="USER" Value="Jeeadmin1" IsSensitive="false"/>
  <Parameter Name="PASSWD" Value="123456" IsSensitive="true"/>
  <Parameter Name="CONN_FACTORY" Value="JMSSConnectionFactory1" IsSensitive="false"/>
</Parameters>
```

7

ADJ1UHMARK4RE – Update Handler in Mark for Rerate Mode

This chapter describes the Update Handler in Mark for Rerate mode.

Description

Mark for Rerate is a mode of the Update Handler that extends its functionality to mark subscribers for rerate. This implementation uses the same code as the Update Handler in Full or Incremental mode. Therefore, for information not covered in this chapter, see the “[ADJ1UH – Update Handler in Full and Incremental Mode](#)” chapter.

Process Type

The Update Handler in Mark for Rerate mode supports two process types:

- *ADJ1UHMARK4RD (daemon, incremental processing)* – In this mode, the Mark for Rerate process takes files from Audit & Control. Then, for each combination of customer, cycle code, and cycle instance in the file, it inserts a request into the APE1_SUBSCRIBER_RERATE table.
Although ADJ1UHMARK4RD is a daemon, when activated as part of an operational map, ADJ1UHMARK4RD behaves as a batch job. In this case, the uh.mode.deamon property is set to N, and the process exits when there are no more files in Audit & Control.
- *ADJ1UHMARK4RB (batch job, full processing)* – In this mode, the implementation defines the leading query to extract leading data from the AUH1_REQUESTS table. This implementation marks all subscribers that match predefined criteria for rerate, using parameters provided by the user via Amdocs Monitoring & Control. This job is executed by request.

Run Frequency

- *Daemon (incremental processing)* – Runs continuously.
- *Batch job (full processing)* – Runs by request, when refresh of all data is necessary. Run frequency is therefore determined by customer need.

Participation in Generic Maps

This section contains information on the generic maps in which the process participates.

Map Name

The ADJ1UHMARK4RD process participates in the Undo map.

For more information about this map, see *Amdocs Billing Operator Guide*.

Severity Level

Severity Level 2 – The map must hold until the process is fixed.

Activation

This section describes how to activate the Mark for Rerate process in various modes.

Mark for Rerate in Daemon Mode

Command Line:	<pre>ADJ1_MARK4RERATE_Daemon_Shell_Sh -n <APPLICATION_ID> -c "-profileFile <The name of the profile file> -runningMode <Running mode> -envVar<Environment variables> -debugPort <Debug port number>" -e <EWD_EXE> -l</pre>  <p><i>Note: The -e parameter is optional. It is used in environments with Availability Manager. The -l parameter is also optional. It indicates that the process must run with the last working properties directory without creating a new one.</i></p>
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_MARK4RERATE_Daemon_Shell_Sh
Executable Name:	N/A

Example:

```
ADJ1_MARK4RERATE_Daemon_Shell_Sh -n MAN_MARK459 -c "-profileFile UHImplBllUndoMark -runningMode FORCE_START -envVar_abc 123 -debugPort 1234" -e gn1avm_ewd
```

Mark for Rerate in Job Mode

Command Line:	<pre>ADJ1_MARK4RERATE_Job_Shell_Sh -n <APPLICATION_ID> -c "-profileFile <The name of the profile file> -runningMode <Running mode> -envVar<Environment variables> -debugPort <Debug port number>" -e <EWD_EXE> -l</pre>  <p><i>Note: The -e parameter is optional. It is used in environments with Availability Manager. The -l parameter is also optional. It indicates that the process must run with the last working properties directory without creating a new one.</i></p>
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_MARK4RERATE_Job_Shell_Sh
Executable Name:	N/A

Examples:

- *BL_MARK – ADJ1UHMARK4RD (behaving as a job as part of an operational map)*
 - `ADJ1_MARK4RERATE_Job_Shell_Sh -n BL_MARK557 -c "-profileFile UHImplBllUndoMark -runningMode FORCE_START -envVar_abc 123 -debugPort 1234" -e gn1avm_ewd`

- *MAN_MARK – ADJ1UHMARK4RB (full processing)*
 - ADJ1_MARK4RERATE_Job_Shell_Shi -n MAN_MARK457 -c "-profileFile UHImplManMarkByOffer -runningMode NORMAL -envVar_REQUEST_ID 1" -e gn1avm_ewd
 - ADJ1_MARK4RERATE_Job_Shell_Shi -n MAN_MARK457 -c "-profileFile UHImplManMark -runningMode NORMAL -envVar_FROM_DATE 1/8/2008 -envVar_TO_DATE 30/9/2008 -envVar_OFFER_ID 29777 -envVar_CYCLE_CODE 1 -envVar_MARK" -e gn1avm_ewd

For more information about the activation scripts, see the “[Scripts](#)” section in the “[Introduction](#)” chapter.

Shutdown

The Mark for Rerate daemon can be shut down gracefully using the following command.

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> <procType> <instanceNumber> 100
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

ADJ1_Send_Admin_Command_Sh hpbl820 MAN_MARK 459 100

When the Mark for Rerate job is done, it shuts down on its own.

Dependent Processes

In the Undo map, the following process can be run only after the successful completion of the ADJ1UHMARK4RD process:

- *ADJIRRPEOC – Rerate Prepare for End of Cycle*

Recovery Instructions

This section describes the steps to be taken if the Mark for Rerate process fails.

Recovery in Job Mode

To enable the Mark for Rerate process to recover after failure, the leading query must be sorted. Sorting according to the database index is recommended.

In addition, the profile XML file must specify that the implementation supports recovery. For more information, see *Turbo Charging Update Handler XML Configuration Guide*.

Checkpoints

At each defined interval of time (configurable), a checkpoint is performed. During a checkpoint:

1. The Leading Data Reader is suspended, and the Feeder is emptied of all groups.
2. The groups are moved to the Ready for Work queue (regardless of size) and are processed as usual. The Update Handler Updates (AUH1_UPDATES) control table contains the number of leading data objects that have been processed, per leading query.
3. After this is finished, the Leading Data Reader resumes and continue as usual until the next checkpoint.



Note: Naturally, use of checkpoints has a negative impact on performance because checkpoints require system suspension.



Note: In the out-of-the-box implementation, no recovery is supported.

Recovery in Daemon Mode

In the event of failure, each request adapter is responsible for implementing its own recovery mechanism.

File Adapter

When working with Audit & Control, the File Adapter first extract files in IU (In Use) status. This ensures that if the process crashes, the files that were open at the time of failure are handled again.



Note: For these reasons, there is no need to perform checkpoints in daemon mode (for incremental processing).

Recovery Flow

When the Mark for Rerate process is started with the Recovery flag (only for the Full mode), it first checks whether the profile matches the one currently listed in the Update Handler Control (AUH1_CTRL) control table (status=R for Running) with the same parameters as in the Mark for Rerate's Profile Context Parameters (AUH1_PROF_CTXT_PARAMS) table. If it does, the Mark for Rerate process:

1. Changes the status to E (error) in the Update Handler Control (AUH1_CONTROL) table
2. Checks the Checkpoint (AUH1_CHECKPOINT) table to find out where the previous process stopped
3. Continues from the point at which it stopped

This implies the Mark for Rerate process skips the leading queries and leading data that have already been processed.

8 ADJ1RCN – Update Handler in Reconciliation Mode

This chapter describes the Update Handler in Reconciliation mode.

Description

To achieve the highest performance, the Event Server carries out all of its operations using its internal memory, Amdocs In-Memory Object Storage (AIMOS). However, the reference data that it uses and the usage data that it generates are also stored in the database. To prevent wrong calculations and potential denial of service, it is important to ensure that the data in these two locations is properly synchronized.

Reconciliation is a mode of the Update Handler that extends its functionality to perform reconciliation between the memory of the Event Server and the data extracted from the database. In this mode, the Update Handler takes requests from the ADJ1_REQUESTS table, handles them as specified in the Reconciliation XML file, compares the data, and writes the results of comparison to the ADJ1_DETAILED_CMPR_RSLT table.

Reconciliation may also involve synchronization of data between AIMOS and the database.

This implementation uses the same code as the Update Handler in Incremental mode. Therefore, for information not covered in this chapter, see the “[ADJ1UH – Update Handler in Full and Incremental Mode](#)” chapter.

Process Type

The Update Handler in Reconciliation mode supports the following process types:

- *ADJ1RCND* – Daemon
- *ADJRCNB* – Batch job

Run Frequency

- *Daemon* – Runs continuously and waits for new requests from the ADJ1_REQUESTS table. When a new request with the execution ID that the daemon is handling is inserted into the table, it is processed automatically.
- *Batch job* – Runs by request and goes down when no more requests are found in the ADJ1_REQUESTS table.

Activation

This section describes how to activate the Reconciliation process in various modes.

Reconciliation in Daemon Mode

Command Line:	<pre>ADJ1_RCN_Daemon_Shell_Ssh -n <APPLICATION_ID> -c "-profileFile <The name of the profile file> - runningMode <Running mode> -executionId <Execution ID> -uhType RCN -envVar<Environment variables> - debugPort <Debug port number>" -e <EWD_EXE> -l</pre>  <p>Note: The -e parameter is optional. It is used in environments with Availability Manager. The -l parameter is also optional. It indicates that the process must run with the last working properties directory without creating a new one.</p>
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_RCN_Daemon_Shell_Ssh
Executable Name:	N/A

Example:

```
ADJ1_RCN_Daemon_Shell_Ssh -n RCN430 -c "-profileFile RCN_Implementation -
runningMode FORCE_START -executionId 77 -uhType RCN -envVar_abc 123 -
debugPort 1234" -e gn1avm_ewd
```

Starting Multiple Reconciliation Daemons

When comparison or update requests are inserted directly into the ADJ1_REQUESTS and ADJ1_REQUEST_PARAMS tables in the Oracle database, multiple Reconciliation daemons can be started, each working on its own batch of requests.

To run more than one daemon connected to the same ADJ1_REQUESTS table (so that all the daemons take their messages from one location):

1. Start each daemon with the **-executionId <Execution ID>** parameter. Define a different execution ID for each daemon.
2. If necessary, change the value of the ADJ1RCN connection to the location of the input database table.

To enable running more than one daemon simultaneously, each daemon must have its own process instance.

Reconciliation in Job Mode

Command Line:	ADJ1_RCN_Job_Shell_Sh -n <APPLICATION_ID> -c "-profileFile <The name of the profile file> -runningMode <Running mode> -executionId <Execution ID> -uhType RCN -envVar<Environment variables> -debugPort <Debug port number>" -e <EWD_EXE> -l
	 Note: The -e parameter is optional. It is used in environments with Availability Manager. The -l parameter is also optional. It indicates that the process must run with the last working properties directory without creating a new one.
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_RCN_Job_Shell_Sh
Executable Name:	N/A

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

The Reconciliation daemon can be shut down gracefully using the following command.

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> <procType> <instanceNumber> 100
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

ADJ1_Send_Admin_Command_Sh hpbl820 RCN 430 100

When the Reconciliation job is done, it shuts down on its own.

Preceding Processes

The following daemons must be active for this process to run:

- ADJ1DB2E – Database to Event Server (DB2E)
- ADJ1EVENTSRV – Event Server

Flow

In Reconciliation mode, requests undergo the following flow.

Request Creation

The user can create comparison requests in either of the following ways:

- *Through the Reconciliation Tool GUI* – For more information, see *Turbo Charging Reconciliation Tool User Guide*.
- *Directly in the Oracle database* – The user inserts requests using SQL scripts into the ADJ1_REQUESTS and ADJ1_REQUESTS_PARAMS tables. The requests are based on the request types and their parameters defined in the ADJ1_REQUEST_TYPES and ADJ1_REQUEST_TYPES_PARAMS tables. For more information on these tables, see the *Turbo Charging Data Model*. The following rules apply:
 - All requests to be processed in the same batch must be created with the same execution ID. If the execution ID is not specified, the default value of ‘0’ is used.
 - If several daemons are connected to the same database instance, each daemon must work on its own execution ID.
 - Request ID constitutes the primary key of the ADJ1_REQUESTS table.

For sample SQL scripts, see the “Reconciliation Requests” appendix.

The following request types are supported:

- Compare requests
- Remove requests
- Update requests

These requests can be defined at different levels, for example, by subscriber or by customer. For more information, see *Turbo Charging Reconciliation Tool User Guide*.

If the comparison is to be run for a vast amount of data, a special customizable class divides the comparison request into smaller requests, where each request depends on the results of the preceding one. For more information, see *Turbo Charging Customization Guide*.

Subsequent sections describe the compare, remove, and update requests in more detail.

Compare Requests

The user can create compare requests for the following types of data:

- Subscriber data
- Resources
- Accumulators (cycle accumulators, customer accumulators, and subscriber accumulators)
- Customer’s cycle history

The attributes that are compared for each entity are specified in the RCN_Implementation.xml file. The following attributes are compared for each entity out of the box:

- *Subscriber:*
 - CYCLE_CODE – PK
 - CUSTOMER_SEGMENT – PK
 - UPDATE_ID
 - CURRENCY_ID
 - BE
 - CUSTOMER_ID
 - SUBSCRIBER_HASH_VALUE



Note: The full primary key of a subscriber consists of SUBSCRIBER_ID, CYCLE_CODE, and CUSTOMER_SEGMENT.

- *Subscriber Offers:*
 - OFFER_EXP_DATE
 - SOURCE_OFFER_AGR_ID
 - SOURCE_OFFER_INSTANCE
 - EFF_ACT_CODE_PROR
 - EXP_ACT_CODE_PROR
 - UPDATE_ID
 - CYCLE_CODE – PK
 - CUSTOMER_SEGMENT – PK



Note: The full primary key of an offer consists of SUBSCRIBER_ID, OFFER_ID, OFFER_INSTANCE, OFFER_EFF_DATE, CYCLE_CODE, and CUSTOMER_SEGMENT.

- *Subscriber Parameters:*
 - EXPIRATION_DATE
 - UPDATE_ID
 - PARAM_TYPE_ID
 - PARAM_VALUE

- CUSTOMER_SEGMENT – PK
- CYCLE_CODE – PK



Note: The full primary key of a subscriber parameter consists of SUBSCRIBER_ID, PARAM_ID, OFFER_INSTANCE, EFFECTIVE_DATE, CYCLE_CODE, and CUSTOMER_SEGMENT.

■ *Resources:*

- SUB_STATUS
- UPDATE_ID
- PAYMENT_CATEGORY
- CUSTOMER_ID
- BILL_CYCLE
- NEW_BILL_CYCLE
- CHG_CYC_REQ_DATE
- LARGE_CUST_IND
- RESOURCE_HASH_VALUE
- SUBSCRIBER_HASH_VALUE
- EXPIRATION_DATE
- SUBSCRIBER_ID
- RESOURCE_SEGMENT – PK



Note: The full primary key of a resource consists of RESOURCE_VALUE, RESOURCE_TYPE, EFFECTIVE_DATE, and RESOURCE_SEGMENT.

■ *Accumulators:*

- CUSTOMER_ID – PK
- OWNER_ID – PK
- ITEM_ID – PK
- ACCUM_TYPE_ID – PK
- OFFER_INSTANCE – PK
- OWNER_TYPE – PK
- DIMENSION_ID – PK



Note: The full primary key of an accumulator consists of CUSTOMER_ID, CYCLE_YEAR, OWNER_ID, OWNER_TYPE, ITEM_ID, OFFER_INSTANCE, DIMENSION_ID, CYCLE_CODE, CYCLE_INSTANCE, and CUSTOMER_SEGMENT.

- *Customer's cycle history (APE1_CUST_CYCLE_HISTORY):*

- NEW_CYCLE_CODE
- OLD_CYCLE_CODE
- LEADING_DATA_ID
- UPDATE_ID
- CUSTOMER_SEGMENT – PK



Note: The full primary key of a customer's cycle history consists of CUSTOMER_ID, EFFECTIVE_DATE, CYCLE_CODE, and CUSTOMER_SEGMENT.

Remove Requests

The Compare Cycle Accumulators request (CMP_CYC_ACCUMS) can show differences in the accumulators between AIMOS and the database. These differences are written into the ADJ1_CMPR_DETAIL_RSLT table. The Remove Accumulator request runs on DIFFER rows that are in 'RD' status and sends a request to the Event Server to remove them from AIMOS. Then, these accumulators can be loaded to AIMOS from the database.

Update Requests

The user can create update requests for the following types of data:

- From the database to AIMOS:
 - Subscriber data
 - Resources
 - Accumulators (for updating specific accumulator attributes)
 - Customer's cycle history
- From AIMOS to the database:
 - Accumulators

It is recommended that users place requests to update an accumulator in memory via the Reconciliation Tool GUI. Such requests include the key of the accumulator and specify the attribute to be updated, the operation required, and the values with which the attribute is to be updated. This information is inserted into the ADJ1_REQUESTS_PARAMS table like for any other request.

For all other update requests, the request to update the data from the database to the memory of the Event Server only requires the user to specify the entity key.

A request to update accumulators from the database to AIMOS cannot include the following attributes:

- Core attributes:
 - Accumulator ID
 - Dimension ID
 - Cross-cycle indicator
- Accumulator keys:
 - Agreement ID
 - Item ID
 - Cycle code
 - Cycle month
 - Customer ID
 - Offer instance
 - Cycle year
 - Customer Segment
 - Owner Type

In addition, whether an attribute can be updated depends on its data type and the operation that is performed on its value. The following table shows the supported data types for each operation.

Operation	Supported	Not Supported	Comments
==	<ul style="list-style-type: none"> ■ STRING ■ BOOLEAN ■ DATE ■ INTEGER ■ UOM INTEGER ■ FLOAT ■ UOM FLOAT 	N/A	Complex attributes are supported if the complex ('X') is specified as a parameter in the ACCUMULATOR_ATTRIBUTES column of the ADJ1_REQUEST_PARAMETERS table.
+=	<ul style="list-style-type: none"> ■ DATE ■ INTEGER ■ UOM INTEGER ■ FLOAT ■ UOM FLOAT 	<ul style="list-style-type: none"> ■ STRING ■ BOOLEAN ■ COMPLEX STRING ■ COMPLEX DATE ■ COMPLEX INTEGER ■ COMPLEX BOOLEAN ■ COMPLEX UOM INTEGER ■ COMPLEX FLOAT ■ COMPLEX UOM FLOAT 	N/A

Operation	Supported	Not Supported	Comments
<code>-=</code>	<ul style="list-style-type: none"> ■ DATE ■ INTEGER ■ UOM INTEGER ■ FLOAT ■ UOM FLOAT 	<ul style="list-style-type: none"> ■ STRING ■ BOOLEAN ■ COMPLEX STRING ■ COMPLEX DATE ■ COMPLEX INTEGER ■ COMPLEX BOOLEAN ■ COMPLEX UOM INTEGER ■ COMPLEX FLOAT ■ COMPLEX UOM FLOAT 	N/A
<code>*=</code>	<ul style="list-style-type: none"> ■ DATE ■ INTEGER ■ UOM INTEGER ■ FLOAT ■ UOM FLOAT ■ COMPLEX INTEGER ■ COMPLEX UOM INTEGER ■ COMPLEX FLOAT ■ COMPLEX UOM FLOAT 	<ul style="list-style-type: none"> ■ STRING ■ BOOLEAN ■ COMPLEX STRING ■ COMPLEX DATE ■ COMPLEX BOOLEAN 	When this operation is used, the values of the attributes are multiplied by the specified number.
<code>/=</code>	<ul style="list-style-type: none"> ■ DATE ■ INTEGER ■ UOM INTEGER ■ FLOAT ■ UOM FLOAT ■ COMPLEX INTEGER ■ COMPLEX UOM INTEGER ■ COMPLEX FLOAT ■ COMPLEX UOM FLOAT 	<ul style="list-style-type: none"> ■ STRING ■ BOOLEAN ■ COMPLEX STRING ■ COMPLEX DATE ■ COMPLEX BOOLEAN 	When this operation is used, the values of the attributes are divided by the specified number.

Process Start-up

At start-up, the Reconciliation tool performs the following:

1. Reads the ADJ1_REQUEST_TYPES and ADJ1_REQUEST_TYPES_PARAMS configuration tables, which specify the existing supported request types and their parameters.
2. Verifies that the requests in the ADJ1_REQUESTS and ADJ1_REQUESTS_PARAMS were inserted according to the definitions in these reference tables.
3. Verifies the requests against the definitions in the implementation XML file.

Main Flow

At runtime, the Reconciliation tool performs the following activities:

1. Manages all the pools in the application:
 - Thread pool
 - Database connection pool
 - Event Server connection pool
2. Selects Ready or Waiting requests from the ADJ1_REQUESTS table according to the specified execution ID, changes their status to In Process, and runs them in a thread taken from the thread pool.

Data Comparison

For data comparison, the Reconciliation tool executes the following flow:

1. Gets the leading data:
 - *For accumulators* – From each Event Server
 - *For all other data* – From the database
2. Extracts the data from the database and the Event Server.
3. Takes a data row from the Event Server and looks for it in the database using the key.
4. Compares the data.
5. Writes the comparison results in the data row itself as one of the following:
 - *EQUAL* – Equal (there are no differences between the data in the database and the data in AIMOS for the object).
 - *DIFFER* – Not equal (there are differences between the data in the database and the data in AIMOS for the object). In this case, the status is passed to the parent request if it exists.



Note: If an error occurs during the compare operation, the Reconciliation tool changes the status of the request in the ADJ1_REQUESTS table to 'ER' and does not retry to process it.

6. After executing all queries and compare actions, performs the following:
 - For requests created in the Oracle database, populates the following result tables:
 - *ADJ1_CMPR_ENTITY_KEY* – Stores the key of each entity that was compared. For example, if the subscribers in a specific cycle are compared, the table presents the data by the subscriber key, offer key, and subscriber parameter key. This table supports various entities whose keys and the number of fields in the key differ. As a result, it is possible to query different parts of the key, for example, to select all entities related to a specific cycle.
 - *ADJ1_CMPR_DETAIL_RSLT* – Stores the detailed results of the comparison. Each entity is presented with the data that was compared for it. The information to be presented in this table is configurable. The user can choose to view all rows or only the ones that show a difference between the data stored in AIMOS and in the database, and display all attributes or only the ones that were not equal (for more information, see the “[Configuration Parameters](#)” section in this chapter).
 - For requests created via the GUI, also presents the results in the corresponding windows. For more information, see *Turbo Charging Reconciliation Tool User Guide*.
- For more information on these tables, see *Turbo Charging Data Model*.

Data Removal

Normally, data removal occurs after data comparison based on the comparison results.

For data removal, the Reconciliation tool executes the following flow:

1. Sends an Accumulator Removal request (RMV_CMP_ACCUMS_MEM) to the Event Server, which removes the accumulator from AIMOS.
2. Changes the status of the request in the ADJ1_REQUESTS table.



Note: If the accumulator does not exist in memory, the Reconciliation tool changes the status of the request in the ADJ1_REQUESTS table to 'ER' and does not retry to process it.

Data Update

The Update Handler in Reconciliation mode enables synchronizing the data between AIMOS and the database. Normally, data update occurs after data comparison based on the comparison results.

For data update, the Reconciliation tool executes the following flow:

1. Depending on the type of the request, performs the following:
 - *Update an entity in memory* – Sends an update request to the Event Server, which loads the data from the database according to the entity key specified in the ADJ1_REQUESTS_PARAMS table.
 - *Update an accumulator in memory* – Sends an update request to the Event Server, which loads the data from the database according to the accumulator and its attribute specified in the ADJ1_REQUESTS_PARAMS table.
 - *Update an accumulator in the database* – Flushes the accumulator from the Event Server to the database according to the specified accumulator key specified in the ADJ1_REQUESTS_PARAMS table.
2. Changes the status of the request in the ADJ1_REQUESTS table.

Parameters

The Update Handler in Reconciliation mode requires the same parameters as the Update Handler in Incremental mode (for more information, see the “Parameters” section in the “ADJ1UH – Update Handler in Full and Incremental Mode” chapter). In addition, it takes the following parameters.

Parameter	Description	Possible Values
Execution ID	Specifies the requests to be handled by a specific instance of the Reconciliation daemon. Execution ID enables parallel handling of requests from the same Requests table. If several daemons are connected to the same database, each daemon must work on its own execution ID.	Any number
UH Type	Specifies the Update Handler mode that is running.	<ul style="list-style-type: none">▪ <i>RCN</i> – Reconciliation▪ <i>UH</i> – Update Handler

Configuration Parameters

The Update Handler in Reconciliation mode has the same configuration parameters as the Update Handler in Incremental mode (for more information, see the “Configuration Parameters” section in the “ADJ1UH – Update Handler in Full and Incremental Mode” chapter).

In addition, it uses the following parameters.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
RCN	uh.compare.request	ignoredFields	UPDATE_ID	<p>The columns that must be ignored by comparison requests. This means that even if there are differences in the values of the ignored columns, the result of the comparison is EQUAL (provided that there are no differences in the other columns).</p> <p>The following considerations must be taken into account while populating this parameter:</p> <ul style="list-style-type: none"> ■ Multiple column names can be specified. In this case, they must be separated by a comma. <p><i>Example:</i> UPDATE_ID, N1, N2</p> <ul style="list-style-type: none"> ■ The columns specified in this parameter are ignored in all types of comparison requests, even if the ignored column appears in different tables. ■ Column names must be specified exactly like they appear in the table. ■ Columns that are mandatory for fetching the data from the database (such as CUSTOMER_ID, SUBSCRIBER_ID, or ITEM_ID) must not be specified as the columns to be ignored.
RCN	uh.connection	maxConnectAttempts	-1	The number of attempts to reconnect to the Event Server if the first attempt has failed. The value of '-1' indicates no reconnection attempts.
RCN	uh.general	batchSize	250	The maximum number of records taken in a single bulk.
RCN	uh.network.connection	initialize	Y	Indicates whether to initialize a network connection to the Event Server.
RCN	uh.rcn	maxBuckets	6	The maximum number of buckets in a group. Each bucket contains a portion of a customer group. Groups of buckets are processed in parallel.
RCN	uh.rcn.general	requestIdCacheSequenceSize	10000	The size of the request ID cache chunk that is retrieved from the database sequence.
RCN	uh.rcn.general	requestIdCacheSequenceSize	10000	The size of the request ID cache chunk that is retrieved from the database sequence.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
RCN	uh.renovator.sleepTime	init	1000	The initial interval (in milliseconds) between reconnection attempts. This interval is increased up to the value of the uh.renovator.sleepTime.max parameter when failures occur.
RCN	uh.renovator.sleepTime	max	30000	The maximum interval (in milliseconds) between reconnection attempts.
RCN	uh.thread.pool	reqParserCoreSize	4	The number of request Parser threads that can run in parallel.
RCN	uh.thread.pool	reqProviderCoreSize	1	Defines the number of provider threads that can run in parallel.
RCN	uh.timeout.response	threshold	600000	The period of time (in milliseconds) for which the Update Handler in Reconciliation mode must wait for a response from the Event Server.
RCN	uh.view.files.rows	mode	DIFF	Defines which attributes are displayed. If it is set to ‘ALL’, all fields are displayed; otherwise, only the ones with the result of DIFFER are shown.
RCN	uh.view.rows	mode	DIFF	Defines which rows are written to the table. If it is set to ‘ALL’, all rows are displayed; otherwise, only the ones with the result of DIFFER are shown.
RCNB	uh.mode	deamon	N	The working mode of the implementation. In batch mode, the value is ‘N’; in daemon mode, the property is inherited from the Update Handler in Incremental mode with the value of ‘Y’.

Database Connections

The Update Handler in Reconciliation mode connects to the same database tables as the Update Handler in Incremental mode (for more information, see the “Database Connections” section in the “ADJ1UH – Update Handler in Full and Incremental Mode” chapter). In addition, it connects to the following tables:

- Turbo Charging Application area:
 - ADJ1_REQUESTS
 - ADJ1_REQUESTS_PARAMS
 - ADJ1_CMPR_DETAIL_RSLT
 - ADJ1_CMPR_ENTITY_KEY
- Turbo Charging Configuration area:
 - ADJ1_REQUEST_TYPES
 - ADJ1_REQUEST_TYPES_PARAMS

Recovery Instructions

The DB Adapter always takes the requests with the In Process status first. If such requests exist at start-up, it means that the process crashed previously and did not complete them. Therefore, these requests are processed again. After completing all the In Process requests, the DB Adapter proceeds to handle new requests.

Important Remarks

This section describes the clean-up activities for the Update Handler in Reconciliation mode.

Clean-up Job

The Clean-up job cleans the comparison results and old requests from the database according to the policy defined in the application. Because the Reconciliation daemon can stay up for long periods of time, the Clean-up job is invoked periodically according to the process configuration.

9 ADJ1UHCUG – Update Handler in CUG Mode

This chapter describes the Update Handler in CUG mode.

Description

Closed User Group (CUG) is a mode of the Update Handler that extends its functionality. The CM_USR_GRP_MEMBERS table of Amdocs Billing Customer Manager includes lists of resources per group ID. All resources in the list are usually entitled for reduced rates or discounts on tariffs. In the CUG mode, the Update Handler updates the APE1_USER_GROUP_MEM table of Turbo Charging with this information from Amdocs Billing Customer Manager according to the implementation. A separate CUG implementation is defined for full processing, when the Update Handler in CUG mode runs as a batch job and extracts the entire CM_USR_GRP_MEMBERS table. For incremental processing (when only the changes in the CM_USR_GRP_MEMBERS table are handled), the Update Handler in CUG mode runs as a daemon and uses the same implementation as the Update Handler in Incremental mode.

This mode uses the same code as the Update Handler in Full or Incremental mode. Therefore, for information not covered in this chapter, see the “ADJ1UH – Update Handler in Full and Incremental Mode” chapter.

Activation

This section describes how to activate the process.

CUG Update Handler in Job Mode

Command Line:	<pre>ADJ1_UH_CUG_Job_Shell_Sh -n <APPLICATION_ID> -c "<The name of the profile file> -runningMode <Running mode> -envVar<Environment variables> -debugPort <Debug port number>" -e <EWD_EXE> -l</pre>  <p>Note: The -e parameter is optional. It is used in environments with Availability Manager. The -l parameter is also optional. It indicates that the process must run with the last working properties directory without creating a new one.</p>
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_UH_CUG_Job_Shell_Sh
Executable Name:	N/A

Example:

```
ADJ1_UH_CUG_Job_Shell_Sh -n UHF_CUG151 -c "UHImplFullModeCUGRecoveryDelete -runningMode FORCE_START -debugPort 1234" -e gn1avm_ewd
```

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

CUG Update Handler in Daemon Mode

For incremental processing, the Update Handler in CUG mode is activated the same way as the Update Handler in Incremental mode. For more information, see the “[Incremental Mode](#)” section in the “[ADJ1UH – Update Handler in Full and Incremental Mode](#)” chapter.

Shutdown

When running as a daemon for incremental processing, the Update Handler in CUG mode can be shut down gracefully using the following command.

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> <procType> <instanceNumber> 100
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

ADJ1_Send_Admin_Command_Sh hpbl820 UHI_CUG 152 100

When the CUG implementation running as a batch job for full processing is done, it shuts down on its own.

10 ADJ1UHMINFULL – Update Handler in Mini-Full Mode

This chapter describes the Update Handler in Mini-Full mode.

Description

Mini-Full is a mode of the Update Handler that extends its functionality. While the Full mode extracts and loads all the data of all the customers and resources to the target database, the Mini-Full mode updates all the data of a specific customer or resource. The leading population to be refreshed is defined in the AUH1_REQUESTS table. This implementation uses the same code as the Update Handler in Full mode. Therefore, for information not covered in this chapter, see the “[ADJ1UH – Update Handler in Full and Incremental Mode](#)” chapter.

Process Type

Batch job

Run Frequency

Runs by request, when refresh of all data is necessary. Run frequency is therefore determined by customer need.

Activation

Command Line:	<pre>ADJ1_UHMINIFULL_Job_Shell_Sh -n <APPLICATION_ID> -c "-profileFile <The name of the profile file> -runningMode <Running mode> - envVar<Environment variables> -debugPort <Debug port number>" -e <EWD_EXE> -l</pre>  Note: The -e parameter is optional. It is used in environments with Availability Manager. The -l parameter is also optional. It indicates that the process must run with the last working properties directory without creating a new one.
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_UHMINIFULL_Job_Shell_Sh
Executable Name:	N/A

Example:

```
ADJ1_UHMINIFULL_Job_Shell_Sh -n UHF_MINI_RT149 -c "-profileFile
UHImplMiniFullRating -runningMode FORCE_START -envVar_REQUEST_ID 123 -
debugPort 1234" -e gn1avm_ewd
```

For more information about the activation scripts, see the “[Scripts](#)” section in the “[Introduction](#)” chapter.

Important Remarks

- The Update Handler in Mini-Full mode and the Update Handler in Synchronous mode can run in parallel. In this case, if customized profiles for the Mini-Full mode are used, the Delete queries must have an addition WHERE clause so that only the records that are older than the number of seconds specified as a context parameter are deleted. For more information, see *Turbo Charging Update Handler Implementation Best Practices* and *Turbo Charging Update Handler XML Configuration Guide*.
- When the home database is restored after it has failed and processing has been moved to an alternate database, the Update Handler in Mini-Full mode is used to copy the customer and resource data from the alternate database back to the home database.

11 ADJ1DB2E – DB2E

This chapter describes the Database to Event Server (DB2E) process.

Description

DB2E is a multi-threaded process that extracts data from the Turbo Charging database tables and sends the data as events to Event Server processes via an internal protocol between DB2E and Event Server, which is not used for network events.

Multiple instances of DB2E may run simultaneously on the same machine or on different machines. Each instance of DB2E handles a different population of resources and customers. That is, two different instances may not handle the same customer or resource in parallel.

DB2E distributes events to both master and replication Event Server processes, which may run on the same machine as DB2E or on a different one.

Process Type

Daemon

Run Frequency

Runs continuously

Activation

Command Line:	ADJ1_DB2E_Daemon_Shell_Sh -n <APPLICATION_ID> -c "-f <CCF_FILE_NAME>" -e <EWD_EXE>  Note: The -e parameter is optional. It is used in environments with Availability Manager.
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_DB2E_Daemon_Shell_Sh
Executable Name:	gcpf1fwcApp

Examples:

- DB2E process instance:
ADJ1_DB2E_Daemon_Shell_Sh -n DB2E800 -c "-f Apr1_DB2E.ccf" -e gn1avm_ewd
- DB2E_ERR (error recycling) process instance:
ADJ1_DB2E_Daemon_Shell_Sh -n DB2E_ERR888 -c "-f Apr1_DB2E.ccf" -e gn1avm_ewd

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> DB2E \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

ADJ1_Send_Admin_Command_Sh 10.232.53.45 DB2E 800
CPF_GracefulShutdownCommand

Preceding Processes

DB2E can run even if the Update Handler is not running parallel to it. If the Update Handler has never run, DB2E sleeps until the Update Handler runs and inserts data into the database. If the Update Handler ran in the past, DB2E works on the data that already exists in the database.

Affected Applications

The process affects the Event Server in the following way.

DB2E sends events that represent updates to resources or subscribers in the database to the Event Servers. The Event Servers update this data in their shared memory. However, if the DB2E process fails, the Event Servers can still work with the data remaining in shared memory or with the data from the database. (If an Event Server needs some specific data, such as a specific subscriber or specific resource data before DB2E has had a chance to send it to the Event Server, the Event Server fetches this specific data directly from the database and puts it into its own shared memory.)

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*

ADJ1DB2E_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1DB2E_DB2E800_20081109_130927_1.log

- *Console log files:*
ADJ1_DB2E_Daemon_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log
Example:
ADJ1_DB2E_Daemon_inner_Sh_DB2E800_20081109_130927.log
- *Operational log files:*
ADJ1DB2E_<APPLICATION_ID>_\${ABP_MARKET}_<Date>_<Time>.log
Example:
ADJ1DB2E_DB2E800_M3G_20081109_130927.log
- *Event log files:*
ADJ1DB2E_EVENT_LOG_<APPLICATION_ID>_%D_%T_%n.log
Example:
ADJ1DB2E_EVENT_LOG_DB2E800_20111218_113001_0.log
- *Environment variable log files:*
ADJ1_DB2E_Daemon_Shell_Sh_<APPLICATION_ID>_<Date>_<Time>.log
Example:
ADJ1_DB2E_Daemon_Shell_Sh_DB2E800_20081109_130927.log
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/.
- *Console log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/.
- *Operational log files* – \$ABP_LOG.
- *Event log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/. It is also possible to load event log data into the APR1_DB2E_EVENT_LOG_LOAD database table by activating the ADJ1_DB2E_EventLogFileDBLoader_Sh script. When this script is run without any parameters, it informs the operator which parameters are required.
- *Environment variable log files* – \$ABP_LOG.
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter).

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information.
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).
- *Operational log files* – The output of operational scripts.
- *Event log files* – Information about the events sent to the Event Server and the answers received from the Event Server for each event. The type of information that must be printed to the event log is specified by the DB2E_EVENT_LOG_FORMAT configuration parameter. For more information, see the “[Configuration Parameters](#)” section in this chapter.
- *Environment variable log files* – Environment variables used by the process.
- *Light Tracer files* – See the “[Light Tracer Files](#)” section in the “[Introduction](#)” chapter.

Flow

This section describes the flow of the DB2E process.

Process Start-up

This section describes the process flow at start-up.

Checking for Upgrade

At start-up, DB2E searches the APR1_ADMIN_CMD_HISTORY table for an entry with PROCESS_ID matching the current DB2E process instance and COMMAND_ID = 87.

- If such an entry exists, and if UNIQUE_ID = 1, an upgrade is not needed.
- If such an entry does not exist, or if UNIQUE_ID = 0, DB2E upgrades the AUH1_UPDATES table.
 - a. For each group under its responsibility, DB2E selects the rows that were probably inserted by the old Update Handler and have not been processed yet (where AUH1_UPDATES.SEGMENTS_GROUP_ID = 0 and AUH1_UPDATES.SEQ_NUM > APR1_DB2E_CTRL.INCR_LAST_SEQ_NUM).
 - b. These rows are duplicated for each group, and values are populated in the table columns that are new in this version. The value of the ORDER_ID field for these rows remains ‘0’.
 - c. After finishing the upgrade, DB2E inserts or updates the relevant entry in the APR1_ADMIN_CMD_HISTORY table with UNIQUE_ID = 1.

It is expected that in the upgrade flow, the Update Handler is restarted with the new binaries prior to DB2E. If DB2E is restarted first, the Update Handler continues to insert rows with the old structure even after DB2E has finished upgrading the AUH1_UPDATES table. In this case, the operator can restart the Update Handler, manually change the value of UNIQUE_ID to ‘0’, and restart DB2E which will upgrade the additional rows.

Setting Geo Redundancy State

If Geo Redundancy is enabled, at start-up DB2E initializes its internal Geo Redundancy state according to the state that was last updated by DB2E in the APR1_ADMIN_CMD_HISTORY table. DB2E searches for an entry with PROCESS_ID matching the current DB2E process instance and COMMAND_ID = 999.

- If such an entry exists, and if UNIQUE_ID = 1, DB2E initializes its internal state as active.
- If such an entry does not exist, or if UNIQUE_ID = 0, DB2E initializes its internal state as standby.

DB2E updates its state in the APR1_ADMIN_CMD_HISTORY table only when the Availability Manager changes the state from active to standby or from standby to active.

DB2E in Regular and Error Recycling mode at the standby site does not process technical events.

- The data from the APR1_DB2E_CTRL table at the active site is replicated to the APR1_DB2E_CTRL_ACTIVE table at the standby site so that technical events can be processed when the standby site becomes active.
- Technical event entries are inserted into the APR1_DB2E_ERR table only at the active site and replicated to the standby site. As a result, these entries are not processed at the standby site until it becomes active.

Regular Mode

At start-up, the DB2E process in Regular mode performs the following:

1. Loads the routing information for resource and customer groups (the information is retrieved from the configuration tables).
2. Checks which groups it must process according to the routing tables.
3. Divides the groups to be processed among the DbReader threads (the threads that fetch resources, subscribers, cycle history, and technical event data from the database).
4. For each group, checks the last order ID, the last update ID, and the mode in which the group was processed in the previous DB2E run.

In general, DB2E starts processing each group of each target Event Server in Incremental mode. With each new run, processing begins with the next order ID of the group after the last incremental order ID from the previous run, according to the DB2E Control table.

5. If the last DB2E run started to process a group in Full mode and did not finish processing it in Full mode, the current DB2E run continues processing the group in Full mode. When the Full mode is completed for the group, DB2E automatically switches back to incremental processing for this group.
6. Inserts a row per group into the DB2E Control table.

Error Recycling Mode

At start-up, the DB2E process in Error Recycling mode performs the following:

1. Unlike regular DB2E process instances, starts processing all the sequence numbers, update IDs, and events in the APR1_DB2E_ERR table.
2. Sends the events to the Event Servers as usual, and waits for answers from the Event Servers.
3. Marks the processed rows in the APR1_DB2E_ERR table as *already processed*, so that they are not reprocessed the next time DB2E starts in Error Recycling mode (a Clean-up process later deletes these rows from the table). New rows may be inserted to the DB2E Error table while it reprocesses the erred events.
4. As soon as it finishes processing the errors, shuts down.

DB2E_ERR can also be configured to automatically switch to either Incremental mode or Full mode for each of the groups after error recycling is finished. However, this option is not recommended and is reserved for special cases only because a regular DB2E instance may be processing the same groups at the same time. This configuration is defined in the SHUT_DOWN_AFTER_ERR_RECYCLE configuration parameter.

Main Loop

Each thread in the process has its own main loop, and all threads work in parallel.

Orderer Thread

The Orderer thread does the following in its main loop:

1. From each instance of the AUH1_UPADTES table (in the Application and Usage database), selects rows whose ORDER_ID is ‘0’
2. Populates the ORDER_ID field in these rows with sequential values for each SEGMENTS_GROUP_ID, CYCLE_CODE, and SEGMENTS_TYPE
3. Updates the customer or resource segment groups in memory with the maximum order ID

DbReader Threads

Each DbReader thread does the following in its main loop, for each connection to the Customer or Resource database:

1. Checks whether the Update Handler has made new incremental updates (has inserted new update IDs into the Update Handler Updates table).
2. For groups in Incremental mode, opens a cursor in the database to extract all the resource or subscriber data containing the new update IDs. For groups in Full mode, opens a cursor in the database to extract all the resource or subscriber data whose update IDs are within the next range of update IDs to process and whose load indicator is set to ‘Y’ (in Incremental flow, this indicator is ignored).
3. Creates events in memory, groups them into bulks, and puts the bulks into queues for further processing by the Sender threads.

Sender Threads

Each Sender thread sends the events that were created by the DbReader threads to several Event Servers. In its main loop, the thread checks to which Event Server the current bulk of events is to be sent and sends it to the correct socket.

Receiver Threads

Each Receiver thread does the following in its main loop:

1. Waits for answers from several Event Servers to events sent by the Sender threads
2. Marks each answer on the relevant event (if the event is still in the memory of DB2E, and if DB2E has not expired it yet)
3. Performs the following, depending on the answer from the Event Server:
 - If all the answers to all the events in the bulk have been returned, frees the bulk of the events to be re-used by the DbReader thread to which the bulk belongs
 - If the Event Server sends a ‘BUSY’ response, suspends further transmission of events to this server (activates the Throttling mechanism).
 - If the Event Server sends a ‘SUSPEND_GROUP’ response, suspends further transmission of events of this group. This may happen when the group is undergoing a planned maintenance activity in the Event Server.

Timeout Reactor Thread

The Timeout Reactor thread performs the following:

1. Wakes up every predefined period of time.
2. If there are suspended Event Servers or groups that must be automatically resumed, resumes them after a predefined period of time.

3. For each bulk of each DbReader thread of the process, gets the current system time.
4. If the bulk still contains events for which the Event Servers have not yet returned answers, checks whether it is time to resend the unanswered events to the Event Servers.
 - If it is time to resend the events, but the number of completed resends exceeds the maximum number of resends allowed, expires the events.
 - If it is time to resend the events, and the number of resends does not exceed the maximum number allowed, resends the events to the Sender thread, which sends them to the Event Servers.

DbUpdater Threads

The DbUpdater threads perform the following for each DbReader thread of the process:

1. Wait for the next sequence number or update ID of the DbReader to be finished. A sequence number or update ID represents a transaction, which includes several events that belong to one customer or resource group and that were sent to one target Event Server. A finished sequence number or update ID is a transaction for which DB2E has received answers to each event or expired some of the unanswered events.
2. For each finished sequence number or update ID, free the sequence number entry and its event bulks to be re-used by the DbReader thread to which the sequence number and the bulks belong.
3. Update the DB2E Control table with the last finished incremental order ID and sequence number, or the last finished full update ID, in the row of the relevant group.

Process Shutdown

Process shutdown depends on the process mode, as follows.

Regular DB2E Instance

Because a DB2E instance that works in Regular mode is a daemon, it only shuts down if it receives a specific manual request to do so. No special actions are performed during shutdown because the DbUpdater threads continuously update the DB2E Control table with the progress of the work that was done by DB2E for each group.

The subsequent DB2E run continues processing each group exactly from the place at which the previous run left off:

- In Incremental mode, according to the last incremental order ID of the group that was recorded in the DB2E Control table in the previous run, or
- In Full mode, according to the last full update ID of the group, that was recorded in the DB2E Control table during the previous run

Error Recycling DB2E Instance

A DB2E instance that works in Error Recycling mode shuts down when it finishes processing all the entries in the APR1_DB2E_ERR table.

Environment Variables

All the environment variables that affect the process are defined automatically by the DB2E activation script.

It is recommended (but not mandatory) that the operator change the default values of the following environment variables before starting DB2E, according to the expected numbers of resource, subscriber, technical, and change cycle events in the system.

Environment Variable	Description	Default Value
ADJ_DB2E_LOGGER_SEVERITY	The minimum severity of log messages to be printed	20000
ADJ_DB2E_TRACER_SEVERITY	The minimum severity of trace messages to be printed	20000
APR_DB2E_LOGGER_ENABLE	Enables the event log output. In production, this parameter must always be set to 'Y'. The DB2E_EVENT_LOG_ENABLED parameter actually controls the printing of messages to the log.	Y
DB2E_CCF_DB_ORDERER_IWDTASK_HANGUP	The time period for which the Orderer task threads can run without sending a heartbeat to the Internal Watchdog. If the task threads do not send a heartbeat to the Internal Watchdog within this time period, the task is considered blocked.	300
DB2E_CCF_DB_ORDERER_IWDTASK_WEIGHT	<p>The weight of the Orderer task. If $(\langle \text{Number of blocked threads} \rangle / \langle \text{Total number of threads} \rangle) * \text{Weight} \geq 100$, the Availability Manager is notified. For example, if a task has four threads, two of which are blocked, the Availability Manager is notified only if the weight is configured as 200 or if all the threads become blocked.</p> <p>To ensure that the Availability Manager kills the DB2E instance even if only one thread is blocked for more than the hang-up time, the task weight must be defined as $\langle \text{Number of Orderer threads} \rangle * 100$.</p>	0
DB2E_CCF_DB_READER_CUST_SEG_IWD_TASK_HANGUP	The time period for which the Customer DbReader task threads can run without sending a heartbeat to the Internal Watchdog. If the task threads do not send a heartbeat to the Internal Watchdog within this time period, the task is considered blocked.	300

Environment Variable	Description	Default Value
DB2E_CCF_DB_READER_CUST_SEG_IWD_TASK_WEIGHT	<p>The weight of the Customer DbReader task. If $(\langle \text{Number of blocked threads} \rangle / \langle \text{Total number of threads} \rangle) * \text{Weight} \geq 100$, the Availability Manager is notified. For example, if a task has four threads, two of which are blocked, the Availability Manager is notified only if the weight is configured as 200 or if all the threads become blocked.</p> <p>To ensure that the Availability Manager kills the DB2E instance even if only one thread is blocked for more than the hang-up time, the task weight must be defined as $\langle \text{Number of Customer DbReader threads} \rangle * 100$.</p>	4400
DB2E_CCF_DB_READER_CUST_SEG_TIMEOUT	The timeout of the Customer DbReader threads.	1
DB2E_CCF_DB_READER_RESOURCE_SEG_IWD_TASK_HANGUP	The time period for which the Resource DbReader task threads can run without sending a heartbeat to the Internal Watchdog. If the task threads do not send a heartbeat to the Internal Watchdog within this time period, the task is considered blocked.	300
DB2E_CCF_DB_READER_RESOURCE_SEG_IWD_TASK_WEIGHT	<p>The weight of the Resource DbReader task. If $(\langle \text{Number of blocked threads} \rangle / \langle \text{Total number of threads} \rangle) * \text{Weight} \geq 100$, the Availability Manager is notified. For example, if a task has four threads, two of which are blocked, the Availability Manager is notified only if the weight is configured as 200 or if all the threads become blocked.</p> <p>To ensure that the Availability Manager kills the DB2E instance even if only one thread is blocked for more than the hang-up time, the task weight must be defined as $\langle \text{Number of Resource DbReader threads} \rangle * 100$.</p>	1200
DB2E_CCF_DB_READER_RESOURCE_SEG_TIMEOUT	The timeout of the Resource DbReader threads.	1
DB2E_CCF_DB_UPDATER_INTERNAL_WATCHDOG_TASK_HANGUP	The time period for which the DbUpdater task threads can run without sending a heartbeat to the Internal Watchdog. If the task threads do not send a heartbeat to the Internal Watchdog within this time period, the task is considered blocked.	300

Environment Variable	Description	Default Value
DB2E_CCF_DB_UPDATER_I WD_TASK_WEIGHT	<p>The weight of the DbUpdater task. If $(\langle \text{Number of blocked threads} \rangle / \langle \text{Total number of threads} \rangle) * \text{Weight} \geq 100$, the Availability Manager is notified. For example, if a task has four threads, two of which are blocked, the Availability Manager is notified only if the weight is configured as 200 or if all the threads become blocked.</p> <p>To ensure that the Availability Manager kills the DB2E instance even if only one thread is blocked for more than the hang-up time, the task weight must be defined as $\langle \text{Number of DbUpdater threads} \rangle * 100$.</p>	3700
DB2E_CUSTOMERS_FULL_ENABLED	Indicates whether handling the Start Full admin command is enabled for customer groups.	Y
DB2E_CCF_DB_UPDATER_TIMEOUT	The timeout of the DbUpdater threads.	1
DB2E_CCF_ORDERER_TIMEOUT	The timeout of the Orderer thread in milliseconds.	1000
DB2E_CCF.REACTOR_TIMEOUT	The timeout of the Reactor threads.	900
DB2E_CUST_DB_READERS_THREADS	The number of threads in the DB2E process that read subscriber, technical event, and cycle change data from the database. These threads also prepare the events for sending to the Event Servers.	44
DB2E_DB_UPDATER_THREADS	The number of threads in the DB2E process that update the progress of DB2E in the APR1_DB2E_CTRL table and insert entries into the APR1_DB2E_ERR table.	$(\text{DB2E_CUST_DB_READERS_THREADS} + \text{DB2E_RES_DB_READERS_THREADS}) * 2 / 3$
DB2E_EVENT_LOG_ENABLED	<p>Determines whether the event log is printed. Event logs are necessary for investigations. Therefore, it is recommended that event logs in their short format be enabled in production environments (in other words, with the default value of the DB2E_EVENT_LOG_FORMAT environment variable).</p> <p>By default, event logs are written to the <code>\$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/directory</code>. It is also possible to load event log data into the APR1_DB2E_EVENT_LOG_LOAD database table by activating the ADJ1_DB2E_EventLogFilesDBLoader_Sh script. When this script is run without any parameters, it informs the operator which parameters are required.</p>	Y

Environment Variable	Description	Default Value
DB2E_EVENT_LOG_FORM AT	<p>The format of the event log:</p> <ul style="list-style-type: none"> ■ short (default and recommended for production environments for better performance and less disk space usage) – Each line in the event log contains the following fields: <ul style="list-style-type: none"> • EventStatus: <ul style="list-style-type: none"> □ <i>A</i> – The answer to the event arrived on time. □ <i>L</i> – The answer to the event arrived too late (after expiration). □ <i>E</i> – The event has expired. • ServerAnswer: <ul style="list-style-type: none"> □ <i>0</i> – NO_ANSWER □ <i>1</i> – ANSWER_OK □ <i>2</i> – ANSWER_ERROR □ <i>3</i> – ANSWER_SUSPEND_GROUP □ <i>4</i> – ROUTING_MAP_VERSION_MISMATCH □ <i>5</i> – SERVER_IS_IN_DB_ONLY_MODE □ <i>6</i> – ANSWER_SERVER_BUSY • EventType: <ul style="list-style-type: none"> □ <i>0</i> – RESOURCE □ <i>1</i> – CHANGE_CYCLE □ <i>2</i> – SUBSCRIBER □ <i>3</i> – TECH_EVENT □ <i>4</i> – ROLL_ACCUM • TargetServerId • UpdateMode: <ul style="list-style-type: none"> □ <i>I</i> – Incremental □ <i>F</i> – Full • EventSendTimeNanoSinceProcessStart • EventUniqueId • UpdateId 	short

Environment Variable	Description	Default Value
	<ul style="list-style-type: none"> • EntityId: <ul style="list-style-type: none"> ▫ If EventType is ‘2’ or ‘3’, <SUBSCRIBER_ID> ▫ If EventType is ‘4’, <OWNER_ID> ▫ If EventType is ‘1’, <CUSTOMER_ID> ▫ If EventType is ‘0’, ‘0’ • CustomerId • CycleCode • ResourceValue (empty if this is not a RESOURCE event) • ResourceType (‘0’ if this is not a RESOURCE event) <p><i>Example:</i> A;1;0;3000;l;39863335900;282550259100001;33051;0;9;3;0010000040133;2;</p> <ul style="list-style-type: none"> ▪ <i>long</i> – In addition to the fields that appear in the short format, each line in the event log contains the following fields: <ul style="list-style-type: none"> • RoutingVersion • CursorOpenTimeNanoSinceProcessStart • EventCreationTimeNanoSinceProcessStart • AnswerOrExpirationTimeNanoSinceProcessStart • GroupId • EventSizeIncludingHeader <p><i>Example:</i> E;0;2;3000;l;44944919800;281350259100003;33110;5893;5884;1;;0;0;37902305700;44944590300;53653186200;30002;416;</p>	
DB2E_LAG_IN_AUH1UPDATES_MSEC	The number of milliseconds for which DB2E must wait before picking up the updates from the Update Handler. This variable is not in use.	30000
DB2E_MAX_ORDERING_ROWS	The maximum number of rows in the AUH1_UPDATES table that the Orderer thread can order at a time.	5000
DB2E_NO_COMMIT_TIMEOUT_DELTA_MSEC	The frequency at which DB2E must commit the updates in the APR1_DB2E_CTRL and APR1_DB2E_ERR tables.	1000
DB2E_RES_DB_READERS_THREADS	The number of threads in the DB2E process that read resource data from the database. These threads also prepare the resource events for sending to the Event Servers.	12

Environment Variable	Description	Default Value
DB2E_RESOURCES_FULL_ENABLED	Indicates whether handling the Start Full admin command is enabled for resource groups.	Y
PERFORM_TECH_EVENT_TABLE_UPDATE	Determines whether DB2E must update events in the APE1_TECH_EVENTS table with an indication that a successful answer was received for an event from the Event Server.	N

Parameters

The following parameters must be passed with correct values to the activation script for the DB2E process to run properly.

Parameter	Description	Default Value
APPLICATION_ID	The process instance name	N/A
CCF_FILE_NAME	The name of an XML file that contains the process configuration	APR_DB2E.ccf

Configuration Parameters

This section describes the functional and performance tuning parameters.

Main Data Structure

To tune the performance configuration parameters, it is essential to understand the following data structure that is used by DB2E.

The purpose of this structure is to hold information about all the sequence numbers with the corresponding order IDs, update IDs, bulks, and events that are being handled by DB2E in a given time frame.

The DbReader array contains an entry for each DbReader thread in the process.

Each entry in the DbReader array points to a fixed size array of bulks and to a fixed size array of sequence numbers.

Each sequence number entry points to an array of bulk numbers that are being used by this sequence number (the bulk numbers are indexes of entries in the bulk array of the DbReader).

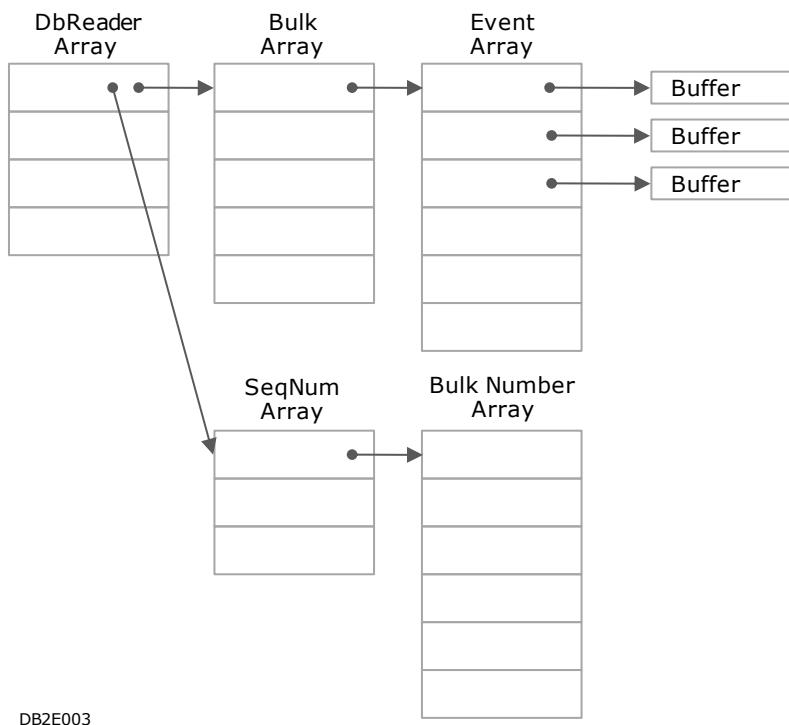
Each bulk entry points to a fixed size array of events.

Each event entry points to a buffer that contains the data of the event. The event data buffer can grow during runtime, if necessary.

The number of DbReaders, the maximum number of sequence numbers per DbReader, the maximum number of bulks per DbReader, and the maximum number of events in a bulk are known during initialization of the process. Therefore, the main data structure is allocated once at process initialization and does not grow during run time (except for the event data buffers, if necessary).

Figure 11.1 shows the main data structure of DB2E.

Figure 11.1 DB2E Process Main Data Structure



The data structure is subscribed to the Global Services of the DB2E process at process initialization. Therefore, the data structure can be used in any DB2E thread or class.

Bulks for which all events were processed successfully (all answers were received from Event Servers) are released for reuse immediately after the last answer is received by the Receiver thread.

Bulks in which some events have not been answered yet, but the bulk has expired, are marked as expired. However, they cannot be released for reuse until all the bulks in the same sequence number have been handled. This is due to the need to write the erred events from the bulk or the entire sequence number to the error table when the sequence number has been handled.

Sequence number entries for which all handling has finished (that is, the handling of all the bulks of the sequence number has completed, and the erred events or the entire sequence number has already been written to the APR1_DB2E_ERR table) are immediately released for reuse by a DbUpdater thread. When the DbUpdater thread releases a sequence number entry for reuse, it also releases all of its bulks for reuse.

Each event within a bulk has a buffer that is allocated once and reused. When the event buffer is too small to contain a specific event, the buffer is re-allocated with a bigger size and then reused.

Functional and Performance Tuning Parameters

The configuration parameters (properties) of the DB2E are defined in the database.

The following table shows the performance tuning (configuration) parameters of the DB2E. These parameters are defined in the APR1_CONF_SECTION_PARAM table using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*).

Some of the parameters are optional and have default values.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE	Description
DB2E	config	ADDITIONAL_TIME_PER_EVENT_FOR_RESEND_INTERVAL_NSEC	Default value: 1000000 nanoseconds	<p>The TIME_INTERVAL_FOR_RESEND[RES CUST]_ULK parameters represent the minimum time to wait per bulk. This parameter allows adding more waiting time per event in the bulk.</p> <p>The actual waiting time before resending a bulk is calculated as follows:</p> $<\text{TIME_INTERVAL_FOR_RESEND[RES/CUST]}_\text{ULK parameter}> + (<\text{actual num of events in bulk}> * <\text{ADDITIONAL_TIME_PER_EVENT_FOR_RESEND_INTERVAL_NSEC}>)$

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE	Description
DB2E	config	BULK_ENTRIES_PER_DB_READER	<p>Default value: 80000</p> <p>The minimum value is the maximum number of records that can be inserted and updated by the Update Handler with one update ID.</p> <p>A higher value may result in better performance.</p> <p>The recommended value is:</p> <p><i>4 * <The maximum number of records that can be inserted and updated by the Update Handler with one update ID></i>, or higher. However, each event in each bulk may consume memory. Therefore, if the DB2E process consumes too much memory, reduce this value, but not below the minimum.</p>	The number of bulk entries that are pre-allocated and used by each DbReader thread. Each entry is reused for holding and processing a new bulk as soon as the handling of the old bulk is finished.
DB2E	config	CUSTOMERS_FULL_ENABLED	\${DB2E_CUSTOMERS_FULL_ENABLED}	Indicates whether the handling of the Start Full admin command is enabled for customer groups.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE	Description
DB2E	config	DB2E_EVNT_LOG_ENABLED	\${DB2E_EVNT_LOG_ENABLED}	<p>Determines whether the event log is printed. Event logs are necessary for investigations. Therefore, it is recommended that event logs in their short format be enabled in production environments (in other words, with the default value of the DB2E_EVENT_LOG_FORMAT environment variable).</p> <p>By default, event logs are written to the \$ABP_APPL_ROOT/log/<APPLICATION_ID>/<process start time stamp>/directory. It is also possible to load event log data into the APR1_DB2E_EVENT_LOG_LOAD database table by activating the ADJ1_DB2E_EventLogFilesDBLoader_Sh script, When this script is run without any parameters, it informs the operator which parameters are required.</p>

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE	Description
DB2E	config	DB2E_EVNT_LOG_FORMAT	\${DB2E_EVNT_LOG_FORMAT}	<p>The format of the event log:</p> <ul style="list-style-type: none"> ▪ short (default and recommended for production environments for better performance and less disk space usage) – Each line in the event log contains the following fields: <ul style="list-style-type: none"> • EventStatus: <ul style="list-style-type: none"> ▫ <i>A</i> – The answer to the event arrived on time. ▫ <i>L</i> – The answer to the event arrived too late (after expiration). ▫ <i>E</i> – The event has expired. • ServerAnswer: <ul style="list-style-type: none"> ▫ <i>0</i> – NO_ANSWER ▫ <i>1</i> – ANSWER_OK ▫ <i>2</i> – ANSWER_ERROR ▫ <i>3</i> – ANSWER_SUSPEND_GROUP ▫ <i>4</i> – ROUTING_MAP_VERSION_MISMATCH ▫ <i>5</i> – SERVER_IS_IN_DB_ONLY_MODE ▫ <i>6</i> – ANSWER_SERVER_BUSY • EventType: <ul style="list-style-type: none"> ▫ <i>0</i> – RESOURCE ▫ <i>1</i> – CHANGE_CYCLE ▫ <i>2</i> – SUBSCRIBER ▫ <i>3</i> – TECH_EVENT ▫ <i>4</i> – ROLL_ACCUM • TargetServerId • UpdateMode: <ul style="list-style-type: none"> ▫ <i>I</i> – Incremental ▫ <i>F</i> – Full • EventSendTimeNanoSinceProcessStart • EventUniqueId

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE	Description
				<ul style="list-style-type: none"> • UpdateId • EntityId: <ul style="list-style-type: none"> ▫ If EventType is ‘2’ or ‘3’, <SUBSCRIBER_ID> ▫ If EventType is ‘4’, <OWNER_ID> ▫ If EventType is ‘1’, <CUSTOMER_ID> ▫ If EventType is ‘0’, ‘0’ • CustomerId • CycleCode • ResourceValue (empty if this is not a RESOURCE event) • ResourceType (‘0’ if this is not a RESOURCE event) <p><i>Example:</i> A;1;0;3000;l;39863335900;2825502591000001;3305 1;0;9;3;00100000040133;2;</p> <ul style="list-style-type: none"> ▪ <i>long</i> – In addition to the fields that appear in the short format, each line in the event log contains the following fields: <ul style="list-style-type: none"> • RoutingVersion • CursorOpenTimeNanoSinceProcessStart • EventCreationTimeNanoSinceProcessStart • AnswerOrExpirationTimeNanoSinceProcessStart • GroupId • EventSizeIncludingHeader <p><i>Example:</i> E;0;2;3000;l;44944919800;2813502591000003;3311 0;5893;5884;1;;0;0;37902305700;44944590300;5365 3186200;30002;416;</p>

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE	Description
DB2E	config	DB2E_MAX_ORDERING_ROWS	\${DB2E_MAX_ORDERING_ROWS}	The maximum number of rows in the AUH1_UPDATES table that the Orderer thread can order at a time.
DB2E	config	DB2E_NO_COMMIT_TIMEOUT_DELTA_MSEC	\${DB2E_NO_COMMIT_TIMEOUT_DELTA_MSEC}	The frequency at which DB2E must commit the updates in the APR1_DB2E_CTRL and APR1_DB2E_ERR tables.
DB2E	config	DB2E_PARAM_NAME_SLEEP_TIME_WAITING_FOR_FREE_SEQ_NUM_NSEC	Default value: 1	Defines the amount of time, in nanoseconds, that a DbReader thread sleeps when it must wait for a free sequence number object to reuse it.
DB2E	config	EVENT_ENTRIES_PER_BULK	Default value: 1	The maximum number of events that one bulk can contain.
DB2E	config	EXPIRE_CHANGE_CYCLE_EVENT_TIME_SEC	10	The number of seconds after which DB2E stops waiting for an answer to a change cycle event from the Event Server and consider the event expired.
DB2E	config	EXPIRE_RESOURCE_EVENT_TIME_SEC	10	The number of seconds after which DB2E stops waiting for an answer to a resource event from the Event Server and consider the event expired.
DB2E	config	EXPIRE_SUBSCRIBER_EVENT_TIME_SEC	10	The number of seconds after which DB2E stops waiting for an answer to a subscriber event from the Event Server and consider the event expired.
DB2E	config	EXPIRE_TECH_EVENT_TIME_SEC	120	The number of seconds after which DB2E stops waiting for an answer to a technical event from the Event Server and consider the event expired.  <i>Note: Normally (depending on the implementation), the Event Server returns an answer to a technical event only after persisting to the database, so DB2E must wait for the answer for a reasonable amount of time.</i>

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE	Description
DB2E	config	FULL_MAX_EVENTS_FROM_CURSOR	1000000	The maximum number of events that are created from a single cursor opened by the DbReader thread, for a customer or resource group in Full mode. The value of the parameter might affect the performance of Full processing in DB2E. The higher the value of this parameter, the fewer cursors are opened to process the whole group in Full mode.
DB2E	config	INCR_MAX_EVENTS_FROM_CURSOR	10000	The maximum number of events that are created from a single cursor opened by the DbReader thread, for a customer or resource group in Incremental mode. The value of the parameter might affect the performance of Incremental processing in DB2E.
DB2E	config	FULL_MAX_UPDATE_IDS_IN_CURSOR	Default value: 100000000	This parameter influences the number of UPDATE_IDs that can be processed at once by the DbReader thread in Full mode.
DB2E	config	INCR_CUST_SLEEP_TIME_WAITING_FOR_NEW_UPDATES_SEC_MAX	Default value: 7	The maximum number of seconds between the polls of the AUH1_UPDATES table if there is no new customer data updated by the Update Handler. When DB2E finds new data in the table, the sleep time between the polls is reduced to 0. If there is no new data, this time gradually grows until the maximum value.
DB2E	config	INCR_MAX_CUST_SEG_CURSOR	Default value: 500000 Recommended value: The maximum number of records that can be inserted and updated by the Update Handler with one update ID.	This parameter influences the number of update IDs that can be processed at once by the DbReader thread in Incremental mode. It is the maximum sum of AUH1_UPDATES.NUM_OF_RECORDS that can be processed at once. This parameter is for subscriber, change cycle, and technical events.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE	Description
DB2E	config	INCR_MAX_RES_SEG_CURS OR	Default value: 100000 Recommended value: The maximum number of records that can be inserted and updated by the Update Handler with one update ID.	This parameter influences the number of update IDs that can be processed at once by the DbReader thread in Incremental mode. It is the maximum sum of AUH1_UPDATES.NUM_OF_RECORDS that can be processed at once. This parameter is for resource events.
DB2E	config	INCR_RES_SLEEP_TIME_WAITING_FOR_NEW_UPDATES_SEC_MAX	Default value: 7	The maximum number of seconds between the polls of the AUH1_UPDATES table if there is no new resource data updated by the Update Handler. When DB2E finds new data in the table, the sleep time between the polls is reduced to 0. If there is no new data, this time gradually grows until the maximum value.
DB2E	config	IS_TEST_MODE_NO_SEND_RECEIVE	Default value: N Possible values: Y or N.	Indicates whether DB2E must run in testing mode, in which DB2E acts as usual (fetches events from the database, creates bulks of events in memory, and updates the DB2E Control table), but does not send events and does not receive answers from any server.
DB2E	config	LAG_IN_AUH1UPDATES_MSEC	\${DB2E_LAG_IN_AUH1UPDATES_MSEC}	The number of milliseconds for which DB2E must wait before picking up the updates from the Update Handler. This parameter is not in use.
DB2E	config	MAX_NUM_OF_EVENTS_IN_PROCESSING_PER_TARGET	Default value: 500	The approximate number of events in processing, per target Event Server, at any given time. This parameter has a double purpose: <ul style="list-style-type: none">■ To avoid overloading the Event Servers with too many events from DB2E when the Event Servers cannot keep up with the pace of the events■ To avoid creating and keeping too many events in the memory of DB2E at any given time.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE	Description
DB2E	config	MAX_RESENGS_PER_BULK	Default value: 0	The maximum number of resends for a Customer or Resource bulk, not including the first transmission. If a bulk has reached this number of resends and must be resent, the unanswered and rejected events in the bulk are expired. That is, the answered events in the bulk are considered successful, and the unanswered and rejected events are considered erred.
DB2E	config	PERFORM_TECH_EVENT_TABLE_UPDATE	\${PERFORM_TECH_EVENT_TABLE_UPDATE}	Determines whether DB2E must update events in the APE1_TECH_EVENTS table with an indication that a successful answer was received for an event from the Event Server.
DB2E	config	PROCESSING_UNITS_FILTER	Default value: Empty Possible value: <ProcessingUnitID>, <ProcessingUnitID>, <ProcessingUnitID>, ... <i>Example:</i> 100, 300, 310	If this parameter is specified, DB2E produces and transmits events only to these processing unit IDs (Event Servers). If this parameter is not specified, DB2E produces events for all processing unit IDs, according to the processed groups. This parameter is to be used mainly for testing purposes. If the TARGET_HOSTS_FILTER parameter is specified in addition to this parameter, both filters are applied. That is, DB2E produces and transmits events only to those Event Servers that match both filters.
DB2E	config	PROGRESS_INFO_PRINT_FREQUENCY	Default value: 10000	For every PROGRESS_INFO_PRINT_FREQUENCY number of events that are sent to a specific server, DB2E prints a progress information message.
DB2E	config	RECONNECT_SERVER_ATTEMPTS_INTERVAL_SEC	Default value: 300 (5 minutes)	The time interval after which the Receiver thread tries to reconnect to a disconnected Event Server. DB2E continues its attempts to reconnect to the Event Server every defined interval of time until it succeeds.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE	Description
DB2E	config	RESOURCES_FULL_ENABLED	\${DB2E_RESOURCES_FULL_ENA BLED}	Indicates whether the handling of the Start Full admin command is enabled for resource groups.
DB2E	config	SEND_MAX_RETRIES_NUM_WHEN_SOCKET_FULL	0	The maximum number of times that DB2E must try to resend events if the network connection with the Event Server is busy. If the value of the parameter is ‘0’, DB2E continues to try to resend events until it succeeds.
DB2E	config	SEQ_NUM_ALLOWED_ERRORS_PERCENT	Default value: 0 percent	<p>The maximum percent of unanswered and rejected events in a transaction. A transaction consists of events that were created by the same DbReader thread and that belong to the same sequence number or update ID, the same group, and the same target Event Server.</p> <p>Beyond this percent of errors, the transaction is written as erred to the DB2E_ERR table with the update ID.</p> <p>Under this percentage of errors, only the unanswered events of the transaction are written to the DB2E_ERR table (one row per event).</p>
DB2E	config	SEQ_NUM_ENTRIES_PER_DB_READER	Default value: 30	The number of sequence number entries that are pre-allocated and used by each DbReader thread. Each entry is reused for holding and processing a new sequence number as soon as the handling of the old sequence number is finished.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE	Description
DB2E	config	SHUT_DOWN_AFTER_ERR_RECYCLE	<p>This parameter is relevant only for DB2E_ERR process instances with PROCESS_TYPE_ID = 14.</p> <p>Default value: Y</p> <p>Possible values: Y or N</p>  <p><i>Note: If DB2E must run in ERR_RECYCLE mode when another instance of DB2E might process the same group or groups in INCR or FULL mode, this parameter must be Y.</i></p>	<p>Using this parameter, DB2E_ERR can be configured to shut down after error recycling is finished.</p> <p>If the value of this parameter is N (do not shut down), then after recycling is finished, DB2E switches the processing mode of the groups to Incremental or Full and continues the processing (this option is not recommended). Otherwise, it shuts down after the error recycling is finished.</p>
DB2E	config	SLEEP_TIME_WAITING_FOR_FREE_BULK_NSEC	Default value: 0	Defines the amount of time, in nanoseconds, that a DbReader thread sleeps when it must wait for a free bulk of events to reuse it.
DB2E	config	SLEEP_TIME_WHEN_SOCKET_FULL_NSEC	20000000	The interval (in nanoseconds) at which DB2E must try to resend events if the network with the Event Server is busy.
DB2E	config	TARGET_HOSTS_FILTER	<p>Default value: Empty</p> <p>Possible value: 'host1' , 'host2' , 'host3' , ...</p> <p><i>Example:</i> 'hpx411' , '10.334.52.58' , ...</p> <p>Because this string is used in an SQL query, the values must be enclosed in single quotation marks, without redundant spaces in the value, and separated by commas.</p> <p>The values may be a name of a host or an IP address, as long as they match the value in the routing tables.</p>	<p>If this parameter is specified, DB2E produces events only for the Event Servers that run on these hosts.</p> <p>If this parameter is not specified, DB2E produces events for all Event Servers, according to the processed groups.</p> <p>This parameter is to be used mainly for testing purposes.</p> <p>If the PROCESSING_UNITS_FILTER parameter is specified in addition to this parameter, both filters are applied. That is, DB2E produces and transmits events only to those Event Servers that match both filters.</p>

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE	Description
DB2E	config	TECH_EVENT_TABLE_UPDATE_THRESHOLD	100	Defines the maximum number of technical events that can be updated by DB2E in the APE1_TECH_EVENTS table by a single update SQL command (if such updates are enabled).
DB2E	config	TIME_INTERVAL_FOR_RESEND_CUST_BULK_MSEC	Default value: 5000 milliseconds	The time to wait, in milliseconds, from the last time a Customer bulk was sent or resent, before resending the bulk. (A bulk is resent if, for some of the events in the bulk, a reject response or no response at all was received from the Event Server. Only the unanswered and rejected events of the bulk are resent.) If the value of the MAX_RESENGS_PER_BULK parameter is 0, it is used as the time after which the unanswered and rejected events in the bulk expire and are written to the APR1_DB2E_ERR table, instead of being resent to the Event Server.
DB2E	config	TIME_INTERVAL_FOR_RESEND_RES_BULK_MSEC	Default value: 5000 milliseconds	Same as TIME_INTERVAL_FOR_RESEND_CUST_BULK_MSEC, only for Resource bulks.
DB2E	config	UPDATER_SLEEP_TIME_WHEN_NO_FINISHED_SEQ_NSEC	Default value: 0 nanoseconds	The number of nanoseconds for which a DbUpdater thread sleeps when it does not find any finished transaction in a DbReader. A finished transaction is a transaction all of whose events have an answer from Event Servers or have expired.
DB2E	LOGGER	Severity	\${ADJ_DB2E_LOGGER_SEVERITY}	The minimum severity of log messages to be printed.
DB2E	Logger.ADJ.DB2E.EVENTLOG	BufferSize	\${BUFFER_SIZE}	The size of the temporary buffer of the event log.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE	Description
DB2E	Logger.AD J.DB2E.E VENTLOG	Enable	<code> \${APR_DB2E_LOGGER_ENABLE } </code>	Enables the event log output. In production, this parameter must always be set to 'Y'. The DB2E_EVENT_LOG_ENABLED parameter actually controls the printing of messages to the log.
DB2E	Logger.AD J.DB2E.E VENTLOG	Filename	<code> \${ABP_APP_ROOT}/log/\${GN1_T ASK_NAME}_EVENT_LOG_\${AP PLICATION_ID}_%D_%T_%n.log </code>	The name of the event log file.
DB2E	Logger.AD J.DB2E.E VENTLOG	FlushPeriod	<code> \${FLUSH_PERIOD} </code>	The interval at which the data from the temporary buffer is flushed to a file.
DB2E	Logger.AD J.DB2E.E VENTLOG	ImmediateFlush	<code> \${IMMEDIATE_FLUSH} </code>	Indicates whether the data is flushed to a file immediately and not periodically.
DB2E	Logger.AD J.DB2E.E VENTLOG	Layout	<code> Db2e_EventLog_Layout </code>	The layout type that the event log uses to format messages.
DB2E	Logger.AD J.DB2E.E VENTLOG	MaxFileSize	<code> \${MAX_FILE_SIZE} </code>	The maximum size of an event log file.
DB2E	Logger.AD J.DB2E.E VENTLOG	Mode	<code> \${MODE} </code>	The working mode of the event log: synchronous or asynchronous.
DB2E	Logger.AD J.DB2E.E VENTLOG	Outputs	<code> DB2E_EVENTLOG </code>	The output object assigned to the event log.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE	Description
DB2E	Logger.AD J.DB2E.E VENTLOG	Severity	0	The message severity threshold for the event log.
DB2E	TRACER	Severity	<code> \${ADJ_DB2E_TRACER_SEVERITY}</code>	The minimum severity of trace messages to be printed.
DB2E, DB2E ERR, ROLL ACCUM	config	UH_MODE_AT_STARTUP	<ul style="list-style-type: none"> ■ <i>INCR</i> – For DB2E process instances with PROCESS_TYPE_ID = 8 ■ <i>ERR_RECYCLE</i> – For DB2E_ERR process instances with PROCESS_TYPE_ID = 14 ■ <i>ROLL_ACCUM</i> – For ROLL_ACCUM process instances with PROCESS_TYPE_ID = 9 <p>No other values are supported for each of the PROCESS_TYPE_IDS above.</p>	<p>DB2E instances with PROCESS_TYPE_ID = 8 always start processing each group in the same mode (Incremental or Full) as the one in which they finished processing this group in the previous run. If DB2E processes a group for the first time, the Incremental mode is used.</p> <p>The mode for processing specific groups (or group ranges) can be changed to Full at runtime by sending one of the following admin commands:</p> <ul style="list-style-type: none"> ■ START_FULL_BY_GROUPS_COMMAND ■ START_FULL_BY_TARGET_SERVER_COMMAND <p>While working in Full mode for the groups that were specified in the command, DB2E continues working in Incremental mode for the remaining groups. At the end of processing in Full mode, the mode automatically changes back to Incremental for the relevant groups.</p>

In addition, the process uses Light Tracer configuration parameters, which are defined in the ADJ1_CONF_SECTION_PARAM table. For more information, see [Light Tracer Parameters](#) in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

For each process instance, an entry with the name of the process instance must be created in the APR1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
DB2E800	DB2E

Any parameter to be overridden at the process instance level must be added to the APR1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
DB2E800	config	SEQ_NUM_ALLOWED_ERRORS_PERCENT	30

For more information on parameters and the properties mechanism, see the “Configuration Parameter Mechanism” appendix.

Configuration Files

The DB2E requires a configuration file (a .ccf file) that is defined via an internal configuration tool. This XML file is supplied as one of the executable files of the product. It must not be changed.

Database Connections

During initialization, the process connects to many reference tables. For a complete description of each table, see *Turbo Charging Data Model*.

After initialization, DB2E connects primarily to the following tables.

Database Type	Table Name	Connection Type (Update, Insert, and so on)	Comments
Application	APR1_ADMIN_CMD_HISTORY	Select, Update, Insert	This table is used for saving the DB2E state with respect to an upgrade from an old version and to the active or standby state in Geo Redundancy.
Application	APR1_DB2E_CTRL	Select, Insert, Update	N/A
Application	APR1_DB2E_CTRL_ACTIVE	Select	This table exists in Geo Redundancy sites. At the standby site, the table contains replicated data from the APR1_DB2E_CTRL table, which is located at the active site.
Application	APR1_DB2E_ERR	Select, Insert, Update	N/A

Database Type	Table Name	Connection Type (Update, Insert, and so on)	Comments
Application	AUH1_UPDATES	Select, Insert, Update	The Orderer task selects rows with ORDER_ID = 0, updates the ORDER_ID, and may insert rows in the upgrade flow.
Customer	APE1_SUBSCR_DATA	Select	N/A
Customer	APE1_SUBSCR_OFFERS	Select	N/A
Customer	APE1_SUBSCR_PARAMS	Select	N/A
Resource	AGD1_RESOURCES	Select	N/A
Usage	APE1_TECH_EVENTS	Select	N/A

Admin Commands

DB2E supports the following admin commands. For information on how to execute the commands, see the “[Handling Admin Commands](#)” section in the “[High Availability](#)” chapter.

Command	Description	Parameters
SUSPEND_GROUPS_COMMAND	Suspends event transmission for specific groups of resources or customers.	<p>--targetServerProcessGroup “<Target Event Server process group>” --segmentsType “<Resources or Customers>” --cycleCode “<Cycle code>” --fromGroup “<From group number>” --toGroup “<To group number>” [--groupRole <“Group role>”] --cycleCode “<Cycle code>” --fromGroup “<From group number>” --toGroup “<To group number>” [--groupRole <“Group role>”]...</p> <p>where group role is one of the following:</p> <ul style="list-style-type: none"> ■ 0 – Signifies that the data is for an Event Server that is acting as the master for the group ■ 1 – Signifies that the data is for an Event Server that is acting as the first replication for the group <p>All the groups in the range specified by one combination of the --fromGroup, --toGroup, and --cycleCode parameters must have the same group role. If the group role is not specified, it is assumed that the data is for an Event Server that is acting as the master for the group.</p> <p><i>Examples:</i></p> <ul style="list-style-type: none"> • ADJ1_Send_Admin_Command_Sh 10.232.53.45 DB2E 800 SUSPEND_GROUPS_COMMAND --targetServerProcessGroup “3000” --segmentsType “Resources” --cycleCode “0” --fromGroup “30” --toGroup “36” --groupRole “1” --cycleCode “0” --fromGroup “51” --toGroup “51” --groupRole “0” • ADJ1_Send_Admin_Command_Sh 10.232.53.45 DB2E 800 SUSPEND_GROUPS_COMMAND --targetServerProcessGroup “3000” --segmentsType “Customers” --cycleCode “1” --fromGroup “18” --toGroup “30”

Command	Description	Parameters
RESUME_GROUPS_COMMAND	<p>Resumes event transmission for specific groups of resources or customers.</p>	<p>--targetServerProcessGroup "<Target Event Server process group>" --segmentsType "<Resources or Customers>" --cycleCode "<Cycle code>" --fromGroup "<From group number>" --toGroup "<To group number>" [--groupRole "<Group role>"] --cycleCode "<Cycle code>" --fromGroup "<From group number>" --toGroup "<To group number>" [--groupRole "<Group role>"]...</p> <p>where group role is one of the following:</p> <ul style="list-style-type: none"> ■ <i>0</i> – Signifies that the data is for an Event Server that is acting as the master for the group ■ <i>1</i> – Signifies that the data is for an Event Server that is acting as the first replication for the group <p>All the groups in the range specified by one combination of the --fromGroup, --toGroup, and --cycleCode parameters must have the same group role. If the group role is not specified, it is assumed that the data is for an Event Server that is acting as the master for the group.</p> <p><i>Examples:</i></p> <ul style="list-style-type: none"> • ADJ1_Send_Admin_Command_Sh 10.232.53.45 DB2E 800 RESUME_GROUPS_COMMAND --targetServerProcessGroup "3000" --segmentsType "Resources" --cycleCode "0" --fromGroup "30" --toGroup "36" --groupRole "0" --cycleCode "0" --fromGroup "51" --toGroup "51" --groupRole "1" • ADJ1_Send_Admin_Command_Sh 10.232.53.45 DB2E 800 RESUME_GROUPS_COMMAND --targetServerProcessGroup "3000" --segmentsType "Customers" --cycleCode "1" --fromGroup "18" --toGroup "30"

Command	Description	Parameters
SUSPEND_TARGET_SERVER_COMMAND	<p>Suspends production and transmission of events to a specific Event Server.</p> <p>DB2E treats the groups handled by a specific Event Server according to its responsibility for the groups (master or replication) in the routing map that DB2E receives from the Availability Manager. In other words, if an Event Server is responsible for one customer group as the master and for one resource group as the replication, and if the --segmentsType parameter is “All”, DB2E sets the processing mode to Full both for the customer group and for the resource group, even though the role was not specified explicitly in the command.</p>	<p>--segmentsType “<Resources, Customers, or All>” --targetServerProcessGroup “<Target Event Server process group>”</p> <p> Note: If the --segmentsType parameter is Resources, only resource groups of this Event Server are suspended. If it is Customers, only customer groups of this Event Server are suspended. If it is “All”, all the groups of this Event Server are suspended.</p> <p>Examples:</p> <ul style="list-style-type: none"> ADJ1_Send_Admin_Command_Sh 10.232.53.45 DB2E 800 SUSPEND_TARGET_SERVER_COMMAND --segmentsType “All” --targetServerProcessGroup “3000” ADJ1_Send_Admin_Command_Sh 10.232.53.45 DB2E 800 SUSPEND_TARGET_SERVER_COMMAND --segmentsType “Customers” --targetServerProcessGroup “3100”
RESUME_TARGET_SERVER_COMMAND	<p>Resumes production and transmission of events to a specific Event Server.</p> <p>DB2E treats the groups handled by a specific Event Server according to its responsibility for the groups (master or replication) in the routing map that DB2E receives from the Availability Manager. In other words, if an Event Server is responsible for one customer group as the master and for one resource group as the replication, and if the --segmentsType parameter is “All”, DB2E sets the processing mode to Full both for the customer group and for the resource group, even though the role was not specified explicitly in the command.</p>	<p>--segmentsType “<Resources, Customers, or All>” --targetServerProcessGroup “<Target Event Server process group>”</p> <p> Note: If the --segmentsType parameter is Resources, only resource groups of this Event Server are resumed. If it is Customers, only customer groups of this Event Server are resumed. If it is “All”, all the groups of this Event Server are resumed.</p> <p>Examples:</p> <ul style="list-style-type: none"> ADJ1_Send_Admin_Command_Sh 10.232.53.45 DB2E 800 RESUME_TARGET_SERVER_COMMAND --segmentsType “All” --targetServerProcessGroup “3000” ADJ1_Send_Admin_Command_Sh 10.232.53.45 DB2E 800 RESUME_TARGET_SERVER_COMMAND --segmentsType “Customers” --targetServerProcessGroup “3100”

Command	Description	Parameters
START_FULL_BY_GROUPS_COMMAND	<p>Sets the processing mode of groups to Full. Upon receiving this command, DB2E changes the mode of the specified groups to Full, and starts sending events for all the resources and customers that belong to these groups to the Event Servers. This may cause transmission of a large amount of data. Therefore, it is recommended that the operator use this command very rarely.</p> <p>When full processing for a group is completed, DB2E sends the corresponding admin command to the Availability Manager.</p> <p>When processing in Full mode is finished for all the groups, DB2E automatically changes the mode of all the processed groups back to Incremental.</p> <p>If the cycle code is on an alternate database, DB2E ignores this command and issues an alert.</p>	<p>--targetServerProcessGroup "<Target Event Server process group>" --segmentsType "<Resources or Customers>" --cycleCode "<Cycle code>" --fromGroup "<From group number>" --toGroup "<To group number>" [--groupRole "<Group role>"] --cycleCode "<Cycle code>" --fromGroup "<From group number>" --toGroup "<To group number>" [--groupRole "<Group role>"]...</p> <p>where group role is one of the following:</p> <ul style="list-style-type: none"> ■ 0 – Signifies that the data is for an Event Server that is acting as the master for the group ■ 1 – Signifies that the data is for an Event Server that is acting as the first replication for the group <p>All the groups in the range specified by one combination of the --fromGroup, --toGroup, and --cycleCode parameters must have the same group role. If the group role is not specified, it is assumed that the data is for an Event Server that is acting as the master for the group.</p> <p><i>Examples:</i></p> <ul style="list-style-type: none"> • ADJ1_Send_Admin_Command_Sh 10.232.53.45 DB2E 800 START_FULL_BY_GROUPS_COMMAND --targetServerProcessGroup "3000" --segmentsType "Resources" --cycleCode "0" --fromGroup "30" --toGroup "36" --groupRole "1"--cycleCode "0" --fromGroup "1" --toGroup "1" --groupRole "0" • ADJ1_Send_Admin_Command_Sh 10.232.53.45 DB2E 800 START_FULL_BY_GROUPS_COMMAND --targetServerProcessGroup "3000" --segmentsType "Customers" --fromGroup "1" --toGroup "100000"

Command	Description	Parameters
START_FULL_BY_TARGET_SERVER_COMMAND	<p>Sets the processing mode of groups to Full. Upon receiving this command, DB2E changes the mode of all the groups of the specified Event Server to Full, and starts sending events for all the resources and customers that belong to these groups to the specified Event Server. This may cause transmission of a large amount of data. Therefore, it is recommended that the operator use this command very rarely.</p> <p>When full processing for an Event Server process group is completed, DB2E sends the corresponding admin command to the Availability Manager.</p> <p>When processing in Full mode is finished for all the groups, DB2E automatically changes the mode of all the processed groups back to Incremental.</p> <p>DB2E treats the groups handled by a specific Event Server according to its responsibility for the groups (master or replication) in the routing map that DB2E receives from the Availability Manager. In other words, if an Event Server is responsible for one customer group as the master and for one resource group as the replication, and if the –segmentsType parameter is “All”, DB2E sets the processing mode to Full both for the customer group and for the resource group, even though the role was not specified explicitly in the command.</p> <p>If the cycle code is on an alternate database, DB2E ignores this command and issues an alert.</p>	<p>--segmentsType “<Resources, Customers, or All>” --targetServerProcessGroup “<Target Event Server process group>”</p> <p><i>Examples:</i></p> <ul style="list-style-type: none"> • ADJ1_Send_Admin_Command_Sh 10.232.53.45 DB2E 800 START_FULL_BY_TARGET_SERVER_COMMAND --segmentsType “Resources” --targetServerProcessGroup “3000” • ADJ1_Send_Admin_Command_Sh 10.232.53.45 DB2E 800 START_FULL_BY_TARGET_SERVER_COMMAND --segmentsType “All” --targetServerProcessGroup “3100”

Command	Description	Parameters
SET_LIGHT_TRACER_COMMAND	<p>Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters.</p>	<ul style="list-style-type: none"> ▪ Activation parameters: <ul style="list-style-type: none"> • <i>turnTracer “on/off”</i> – Activates or stops the Light Tracer • <i>--turnAssert “on/off”</i> – Activates or stops Assertions • <i>--turnLogOnTrace “on/off”</i> – Activates or stops writing log messages in Light Tracer files • <i>--turnFlowTracer/turnAll “on/off”</i> – Activates or stops tracing a single flow or event that is transferred between machines • <i>--turnAll “on/off”</i> – Activates or stops all of the above. The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter). ▪ <i>--moduleID “ALL”</i> ▪ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • <i>--threadInfo “Y/N”</i> – Specifies whether trace or assertion messages include the thread information. • <i>--timestampInd “Y/N”</i> – Specifies whether trace or assertion messages include the time stamp. • <i>--contextInd “Y/N”</i> – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • <i>--noOfMessages “<number>”</i> – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down. • <i>--period “5/10/20/60/120/480/1440”</i> – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated. <p> Note: If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached.</p>

Command	Description	Parameters
		<ul style="list-style-type: none"> • <code>--assertAbortInd "Y/N"</code> – Specifies whether the application is to be aborted if the assertion condition fails. ▪ <code>--commandType "changeBufferSize"</code> – Resizes the Light Tracer buffer at run time. <ul style="list-style-type: none"> • <code>--bufferSize "<number>"</code> – The new size for the Light Tracer buffer. <p style="margin-left: 40px;"><i>Examples:</i></p> <ul style="list-style-type: none"> • <code>ADJ1_Send_Admin_Command_Sh hpbl820 DB2E 800 SET_LIGHT_TRACER_COMMAND --turnTracer "on" --moduleID "ALL" --threadInfo "y"</code> • <code>ADJ1_Send_Admin_Command_Sh hpbl820 DB2E_ERR 888 SET_LIGHT_TRACER_COMMAND --turnTracer "off" --moduleID "ALL"</code>
<code>SET_TRACE_SQL_COMMAND</code>	<p>Activates or stops SQL tracing at runtime. The command name with its parameters must be enclosed in quotation marks.</p> <p><i>Example:</i></p> <pre>ADJ1_Send_Admin_Command_Sh illin1027 <process instance> "SET_TRACE_SQL_COMMAND --traceSqlOn \"yes\" --duration \"1\""</pre>	<p><code>--traceSqlOn "<yes or no>" --duration "<minutes>"</code></p> <p>If the <code>IS_DURATION_REQUIRED</code> environment variable is set to 'Y', the <code>--duration</code> parameter is mandatory. If the duration specified in the command exceeds the allowed maximum value, SQL tracing is activated for the maximum allowed period of time only (60 minutes).</p> <p> Note: The <code>SHUT_DOWN_CHECK_FREQ</code> parameter of the Light Tracer in the <code>ADJ1_CONF_SECTION_PARAM</code> table determines the frequency (defined in the number of messages) at which the SQL Trace Manager checks whether the period specified in the <code>--duration</code> parameter has passed. Therefore, SQL tracing is stopped after the specified duration or after this number of messages has been reached, whichever comes later.</p>
<code>SWITCH_LIGHT_TRACER_ON ALL</code>	Activates the Light Tracer without the ability to specify the mode or send parameters.	N/A

Command	Description	Parameters
SWITCH_LIGHT_TRACER_OFF ALL	Stops the Light Tracer.	N/A
CPF_GracefulShutdownCommand	Shuts down DB2E gracefully.	N/A

Troubleshooting

The following important or frequent errors can occur during the process run.

Error	Preventing Action
A DB2E instance starts to run and immediately goes down.	Search for the ERROR or FATAL string in the DB2E log. The first ERROR or FATAL message probably describes the problem.
DB2E prints the following error: UDP Server could not be initialized...	<p>There is probably another DB2E instance (with the same instance number) already running on the same machine. Running more than one DB2E instance with the same name on the same machine is not allowed.</p> <p>If you need to run more than one DB2E on the same machine, change the name of one of the instances.</p>
DB2E shuts down after it has processed some events.	Search for the ERROR or FATAL string in the DB2E log. The first ERROR or FATAL message probably describes the problem.
DB2E inserts too many rows into the APR1_DB2E_ERR table.	<p>Probably one or more of the target Event Servers are not responding, either because they were disconnected, or they are working too slowly.</p> <ol style="list-style-type: none"> 1. Check whether all the Event Servers are up and running. 2. Check whether the Event Servers are too busy processing events from the network elements and therefore, have no time to handle events from DB2E. If the event servers are loaded with work, increase the value of the following parameters: <ul style="list-style-type: none"> • TIME_INTERVAL_FOR_RESEND_CUST_BULK_MSEC • TIME_INTERVAL_FOR_RESEND_RES_BULK_MSEC
DB2E does not send any events.	<p>The reason for that may be one or more of the following:</p> <ul style="list-style-type: none"> ▪ All of the target Event Servers are disconnected from DB2E, or none of the groups that DB2E must process are active. (Check the values of the IS_ACTIVE and IS_SERVER_ACTIVE fields in the APR1_DB2E_CTRL table.) ▪ There may not be any new sequence numbers to process in the AUH1_UPDATES table. (The Update Handler inserts new updates into this table for DB2E to process.) ▪ The UPDATE_ID values in the AUH1_UPDATES table does not match any of the UPDATE_IDS in the Resources, Subscribers, Customer Cycle History, or Technical Events tables. ▪ The filter parameters in the APR1_CONF_SECTION_PARAM table may have filtered out all of the groups that must be processed by this instance of DB2E, so the DB2E instance does not have any groups to process. Check the APR1_DB2E_CTRL table to see whether the current DB2E instance has inserted any rows at all to this table. If it has not, DB2E has no groups to process.

Recovery Instructions

When DB2E starts, its recovery mechanism is automatically activated. Therefore, no special actions are needed after DB2E is killed or has crashed.

After a DB2E process shuts down, regardless of whether it shut down gracefully or was stopped immediately, the DB2E Control table in the database maintains, per group, a record of the last incremental order ID and sequence number that was fully processed by DB2E. If the shutdown happened when the group was in Full mode, the table maintains, per group, a record of the last full update ID that was fully processed by DB2E.

When DB2E is restarted, it checks the last order ID or update ID for each group to be processed and continues processing the group from there.

In the new run, DB2E may resend some events that were sent in the previous run. This occurs if the events belong to order IDs or update IDs that were not wholly finished in the previous run. Reprocessing of change cycle, resource, and subscriber events by the Event Servers does not harm the data in the shared memory. Technical events are rejected as duplicates by the Event Server if they were already processed once, so resending them from DB2E does not cause any damage.

Important Remarks

This section provides additional information on DB2E.

Suspending and Resuming Event Transmission for Specific Groups

Groups of resources or customers can be suspended. When this happens, DB2E stops producing and sending events that belong to the resources or customers of these groups.

The suspension requires a manual request (admin command) from the operator. The suspension command is useful, for example, if a group is being moved from one Event Server to another.



Note: The suspension command can only be performed while DB2E is running.

Groups that were suspended can be resumed by another manual request (command) from the operator.



Note: For a detailed description of the Suspend Groups and Resume Groups commands, see the “Admin Commands” section in this chapter.

Suspending and Resuming Event Transmission to a Specific Event Server

The production and transmission of events to a specific Event Server can be suspended. When this happens, DB2E stops producing and sending events to the indicated Event Server.

The suspension requires a manual request (admin command) from the operator. The suspension command is useful, for example, if the Event Server is down for some reason.



Note: The suspension command can only be performed while DB2E is running. In addition, before DB2E is started, a filter of processing unit IDs (list of target Event Servers for which events are to be produced) can be specified as a configuration parameter.

The production and transmission of events to an Event Server that was suspended can be resumed by another manual request (command) from the operator.



Note: For a detailed description of the Suspend Target Server and Resume Target Server commands, see the “Admin Commands” section in this chapter.

Interface with Availability Manager

DB2E interfaces with the Availability Manager in the following cases:

- The Availability Manager is responsible for starting the DB2E process (through Amdocs Monitoring & Control).
- The Availability Manager updates DB2E with real-time information about the changes in the master/replication responsibilities of Event Servers for customer and resource groups. DB2E sends events both to the master and the replication Event Servers.
- Amdocs Monitoring & Control sends administrative commands to DB2E through the Availability Manager. These commands include, for example, Start Full Mode, Suspend Groups, Resume Groups, Suspend Target Server, Resume Target Server, and Shutdown.

Defining Multiple Instances of DB2E

Multiple instances of DB2E may run simultaneously on the same machine, or on different machines.

Each DB2E instance must be defined in the routing tables (a set of tables that are used by the Availability Manager) as a separate instance of the DB2E process. The PROCESS_INSTANCE_ID field in the routing tables must contain the number of the instance, and it must be the same number as in the APPLICATION_ID parameter at the process start-up command line.

Example:

If the APPLICATION_ID is DB2E800, the PROCESS_INSTANCE_ID is 800.

Each instance must be also defined in the APR1_CONF_HIERARCHY table in a separate row, where the PROCESS_NAME field is the name of the instance, such as DB2E800. The PROCESS_GROUP field in the APR1_CONF_HIERARCHY table must contain the following value: ‘DB2E,APR,ADJ:ADJ1_CONF_SECTION_PARAM’.

Defining Groups to be Handled by Each DB2E Instance

The following subsections provide guidelines for defining the groups to be handled by each DB2E instance.

Definition in Group Tables

Several instances of the DB2E process may run simultaneously on the same machine, or on different machines.

The resource and customer groups must be divided among the DB2E instances. That is, each DB2E instance must have its own population of resources and customers to handle, and two separate instances must not handle the same groups.

In environments with the Availability Manager, the Availability Manager notifies DB2E about which Event Server is acting as the master and which Event Server is acting as the replication for a group. In environments without the Availability Manager, the population (resource and customer groups) that a specific instance of DB2E handles depends on the configuration in the System Process Groups Configuration (GN1_SYS_SEG_PROC_CFG) table.

In general, the GN1_SYS_SEG_PROC_CFG table must contain at least three rows for each customer or resource group that DB2E must handle:

- A row that contains the group ID and the process group number (PROCESS_EXEC_ID) of the DB2E instance that handles the group
- A row that contains the group ID and the process group number (PROCESS_EXEC_ID) of the master Event Server that handles the group
- A row that contains the group ID and the process group number (PROCESS_EXEC_ID) of the replication Event Server that handles the group

Group Filter Parameters

The following filter parameters can be used for testing purposes or in special situations when the operator wishes to prevent a specific DB2E instance from processing several groups without changing the groups’ configuration in the GN1_SYS_SEG_PROC_CFG table.

In addition to the configuration in the GN1_SYS_SEG_PROC_CFG table, the selection of resource and customer groups to be handled by a specific DB2E instance depends on the values of the following configuration parameters in the APR1_CONF_SECTION_PARAM table. Each parameter can be defined for each DB2E instance (with PARAM_CLASS=“<Db2eProcessInstanceName>”).

The Group Filter parameters are:

- *PROCESSING_UNITS_FILTER* – A list of Event Server process groups for which events can be produced
- *TARGET_HOSTS_FILTER* – A list of target machines to which events can be sent



Note: For a detailed description of the use of these parameters, see the “Configuration Parameters” section in this chapter.

Checking DB2E Backlog

This section contains the queries that return the number of events that a specific DB2E instance must still send to Event Servers, for groups in Incremental or Full mode.

Important notes:

- Every time that a DB2E instance starts, it inserts new entries into the APR1_DB2E_CTRL table. These entries contain the Unix PID of the DB2E process. Each entry in the table represents a customer or a resource group that is being handled by a specific Event Server. The values of the INCR_LAST_SEQ_NUM and FULL_LAST_UPDATE_ID fields of the group are initially copied from the entries that were inserted in the previous run of DB2E. These values continue to grow on the new entries as processing advances. The existing entries in the APR1_DB2E_CTRL table that were inserted by earlier runs of DB2E are not updated anymore. Only the entries that are inserted by the currently running DB2E process are updated with the new INCR_LAST_SEQ_NUM and FULL_LAST_UPDATE_ID values. In other words, the current DB2E process continues the work of the previous DB2E process, but the progress is updated only in the newest rows in the APR1_DB2E_CTRL table.
- The queries listed in this section must be executed only *after* the DB2E process has started and inserted entries with its own PID into the APR1_DB2E_CTRL table.
- If the APR1_DB2E_CTRL table and the data tables, such as AGD1_RESOURCES and APE1_tables, are in different databases (usually, the APR1_DB2E_CTRL table is in the APP database, and APE1_tables are in the USG database), a synonym must be created from each USG database to the APR1_DB2E_CTRL table, and the queries must be executed from each USG database.

The queries about the Full mode assume that DB2E has already started processing in Full mode for all the groups that must be processed in this mode.

To check which groups DB2E has already started processing in Full mode, run the following query:

```
SELECT cycle_code, segments_group_id, segments_type
from APR1_DB2E_CTRL
where full_last_update_id >= 0
and ctrl.PID = (select distinct(pid)
    from apr1_db2e_ctrl
    where start_time = (select max(start_time)
        from apr1_db2e_ctrl
        where db2e_proc_unit_id = :db2e_proc_instance_number -- For example, if
DB2E_1594, then 1594
    ) );
```

To check the number of events that DB2E has already sent to the Event Servers, run the following command:

```
tail -f <the last DB2E log in the $ABP_APPL_ROOT/log/<APPLICATION_ID>/<process start time stamp>/ directory>"
```

This command prints PROGRESS INFO messages every X events (containing the approximate number of events sent to each Event Server). To use this option, you must set the Logger Severity to 20000 (INFO) or lower.

Checking Backlog for Subscribers in Incremental Mode

To check whether the Incremental processing of subscribers in customer groups has finished, run the following query on each CUSTUSG database:

```
SELECT COUNT (1)
FROM APE1_SUBSCR_DATA data,
     APR1_DB2E_CTRL ctrl,
     AUH1_UPDATES updates
WHERE (updates.ORDER_ID > ctrl.INCR_LAST_ORDER_ID OR  updates.ORDER_ID = 0)
      AND data.CUSTOMER_SEGMENT BETWEEN ctrl.FROM_SEGMENT_ID  AND ctrl.TO_SEGMENT_ID
      AND data.cycle_code = ctrl.cycle_code
      AND data.update_ID = updates.UPDATE_ID
      AND ctrl.cycle_code = updates.cycle_code
      AND ctrl.segments_group_id = updates.segments_group_id
      AND ctrl.SEGMENTS_TYPE = updates.SEGMENTS_TYPE
      AND updates.SEGMENTS_TYPE = 'C'
      AND ctrl.PID =
          (SELECT DISTINCT (pid)
           FROM apr1_db2e_ctrl
           WHERE start_time =
              (SELECT MAX (start_time)
               FROM apr1_db2e_ctrl
               WHERE db2e_proc_unit_id = :db2e_proc_instance_number -- For example,
if DB2E_1594, then 1594
          )
       );
;
```

When this query returns 0, it means that all subscribers belonging to the groups that are processed in Incremental mode have been sent to the Event Servers.

Checking Backlog for Resources in Incremental Mode

To check whether the Incremental processing of resources has finished, run the following query:

```
SELECT COUNT (1)
FROM AGD1_RESOURCES data,
     APR1_DB2E_CTRL ctrl,
     AUH1_UPDATES updates
WHERE (updates.ORDER_ID > ctrl.INCR_LAST_ORDER_ID OR  updates.ORDER_ID = 0)
      AND data.RESOURCE_SEGMENT BETWEEN ctrl.FROM_SEGMENT_ID  AND ctrl.TO_SEGMENT_ID
      AND data.update_ID = updates.UPDATE_ID
      AND ctrl.cycle_code = updates.cycle_code
      AND ctrl.segments_group_id = updates.segments_group_id
      AND ctrl.SEGMENTS_TYPE = updates.SEGMENTS_TYPE
      AND updates.SEGMENTS_TYPE = 'R'
      AND updates.cycle_code = 0
      AND ctrl.PID =
          (SELECT DISTINCT (pid)
           FROM apr1_db2e_ctrl
           WHERE start_time =
```

```
(SELECT MAX (start_time)
   FROM apr1_db2e_ctrl
  WHERE db2e_proc_unit_id = :db2e_proc_instance_number -- For example,
if DB2E_1594, then 1594
)
);
```

When this query returns 0, it means that all resources belonging to the groups that are processed in Incremental mode have been sent to the Event Servers.

Checking Backlog for Subscribers in Full Mode

To check how many subscribers DB2E must still send to the Event Servers in Full mode, run the following query on each CUSTUSG database:



Note: This query checks only the subscribers of customer groups which DB2E has already started processing in Full mode.

```
select count(1)
from APE1_SUBSCR_DATA data,
     APR1_DB2E_CTRL ctrl
Where
    data.CUSTOMER_SEGMENT between
        ctrl.FROM_SEGMENT_ID and
        ctrl.TO_SEGMENT_ID
and ctrl.cycle_code <> 0
and data.cycle_code = ctrl.cycle_code
and ctrl.SEGMENTS_TYPE = 'C'
and data.update_id > ctrl.full_last_update_id
and ctrl.full_last_update_id >= 0 -- select only groups for which FULL started
and ctrl.PID = (select distinct(pid)
                from apr1_db2e_ctrl
                where start_time = (select max(start_time)
                                     from apr1_db2e_ctrl
                                     where db2e_proc_unit_id = :db2e_proc_instance_number -- For example, if
DB2E_1594, then 1594
                )
);
;
```

When this query returns 0, it means that all subscribers belonging to the groups that were processed in Full mode have been sent to the Event Servers.

Checking Backlog for Resources in Full Mode

To check how many resources DB2E must still send to the Event Servers in Full mode, run the following query:



Note: This query checks only the resources of resource groups which DB2E has already started processing in Full mode.

```
select count(1)
from AGD1_RESOURCES data,
     APR1_DB2E_CTRL ctrl
Where
    data.RESOURCE_SEGMENT between
```

```
ctrl.FROM_SEGMENT_ID and
ctrl.TO_SEGMENT_ID
and 0 = ctrl.cycle_code
and ctrl.SEGMENTS_TYPE = 'R'
and data.update_id > ctrl.full_last_update_id
and ctrl.full_last_update_id >= 0 -- select only groups for which FULL started
and ctrl.PID = (select distinct(pid)
    from apr1_db2e_ctrl
    where start_time = (select max(start_time)
        from apr1_db2e_ctrl
        where db2e_proc_unit_id = :db2e_proc_instance_number -- For example, if
DB2E_1594, then 1594
    )
);
```

When this query returns 0, it means that all resources belonging to the groups that were processed in Full mode have been sent to the Event Servers.

12 ADJ1GAT2ESRV – GAT2E

This chapter describes the Generic Applicative Tables (GAT) to Event Server (GAT2E) process.

Description

GAT2E is a multi-threaded process that detects the changes in the generic applicative table data of Turbo Charging, handles these updates, and distributes the table data to the Event Servers via an internal protocol in either full or incremental (changes only) mode. GAT2E distributes the data to the Event Servers in its logical cluster that use a specific generic applicative table. These can be master or replication Event Servers.

GAT2E can distribute events to different Event Server processes, which may run on the same machine as GAT2E or on a different one.

Process Type

Daemon

Run Frequency

Runs continuously

Activation

Command Line:	ADJ1_GAT2E_Daemon_Shell_Sh -n <APPLICATION_ID> -c "-f <CCF_FILE_NAME>" -e <EWD_EXE>  Note: The -e parameter is optional. It is used in environments with Availability Manager.
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_GAT2E_Daemon_Shell_Sh
Executable Name:	gcpf1fwcApp

Example:

```
ADJ1_GAT2E_Daemon_Shell_Sh -n GAT2E137 -c "-f APR1_GAT2E.ccf" -e  
gn1avm_ewd
```

For more information about the activation scripts, see the “[Scripts](#)” section in the “[Introduction](#)” chapter.

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> GAT2E \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

```
ADJ1_Send_Admin_Command_Sh 10.232.53.45 GAT2E 137  
CPF_GracefulShutdownCommand
```

Preceding Processes

GAT2E can run even if the Update Handler is not running in parallel. If the Update Handler has never run, GAT2E sleeps until the Update Handler runs and inserts data into the database. If the Update Handler ran in the past, GAT2E works on the data that already exists in the database.

Affected Applications

The process affects the Event Server in the following way.

GAT2E sends events that represent updates to specific generic applicative table data used by Event Server. The Event Servers update this data in their shared memory. However, if the GAT2E process fails, the Event Servers can still work with the data remaining in shared memory or with the data from the database. (If an Event Server needs some specific data before it was updated by GAT2E, the Event Server fetches this specific data directly from the database and puts it into its own shared memory.)

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*

ADJ1GAT2ESRV_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1GAT2ESRV_GAT2E137_20081109_130927_1.log

- *Console log files:*

ADJ1_GAT2E_Daemon_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1_GAT2E_Daemon_inner_Sh_GAT2E137_20081109_130927.log

- *Operational log files:*

ADJ1GAT2ESRV_<APPLICATION_ID>_\${ABP_MARKET}_<Date>_<Time>.log

Example:

ADJ1GAT2ESRV_GAT2E137_M3G_20081109_130927.log

- *Environment variable log files:*

ADJ1_GAT2E_Daemon_Shell_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1_GAT2E_Daemon_Shell_Sh_GAT2E137_20081109_130927.log

- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG
- *Environment variable log files* – \$ABP_LOG
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter)

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information.
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).

In addition, the console log contains a statistics report for GAT2E. The report includes the following information:

- *ES ID* – The ID of the Event Server
- *T* – The table ID
- *M* – The processing mode:
 - *F* – Full
 - *I* – Incremental
- *RD* – The number of records that are ready for sending to the Event Server
- *ST* – The number of records sent to the Event Server and waiting for a response
- *RCV* – The number of records for which a response was received

- *Last update ID* – The last update ID processed
 - *Processed <number of records>*:
 - In Incremental mode, the number of records sent for the update ID
 - In Full mode, the number of records sent until the update ID
- Example:*
- ES ID = 2205 [T:9][M:I][RD/ST/RCV 0/0/0] (last update id = 119458 processed 62 records);
- *Operational log files* – The output of operational scripts.
 - *Environment variable log files* – Environment variables used by the process.
 - *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Flow

This section describes the flow of the GAT2E process.

Process Start-up

At start-up, the GAT2E process performs the following:

1. Loads the list of tables that are to be handled.
2. Initializes the GAT Manager module.
3. Connects to all active Event Servers in its logical cluster. The logical cluster is determined by the FAILURE_GROUP field in the GN1_SYS_PROC_INST_GRP_CFG table. GAT2E connects only to those Event Servers whose FAILURE_GROUP is the same as its own, regardless of the physical cluster in which the Event Servers reside. There is one active and one back-up GAT2E process per FAILURE_GROUP.
4. For each Event Server, receives the current status of each relevant table in the process of a handshake.
5. Opens a session for each Event Server and starts loading the data:
 - a. If the status of a table is not Empty, starts an incremental load of the table data.
 - b. Otherwise, starts a full load of the table. When the full load is complete, starts an incremental load.

Main Loop

Each thread of the process performs the following in its main loop:

1. Receives a new update for a particular table from the AUH1_UPDATES table.
2. Retrieves from the GAT Manager a serialized message with a new update data for this table.

3. For each session, checks whether the new update is relevant for it (it is relevant if the Event Server uses it).
4. Adds a new message to the queue of incremental events.
5. If the Event Server is not overloaded (controlled by configuration parameters), sends the new message immediately.
6. If the Event Server sends a ‘BUSY’ response, resends the request after a predefined period of time (activates the Throttling mechanism).

If Geo Redundancy is enabled, GAT2E is active at both sites. At the standby site, it performs the same flow as at the active site and sends messages to the Event Server, similar to DB2E.

Process Shutdown

Because GAT2E is a daemon, it only shuts down if it receives a specific manual request to do so. No special actions are performed during shutdown.

Environment Variables

All the environment variables that affect the process are defined automatically by the GAT2E activation script.

GAT2E uses the following environment variables that are initialized by the op_adj_env_sh script.

Variable Name	Description	Valid Values/ Default Value
APR_GAT2E_CLEANUP_IWD_TASK_HANGUP	The time period for which the Clean-up task threads can run without sending a heartbeat to the Internal Watchdog. If the task threads do not send a heartbeat to the Internal Watchdog within this time period, the task is considered blocked.	300
APR_GAT2E_CLEANUP_IWD_TASK_WEIGHT	The weight of the Clean-up task. If $(\text{Number of blocked threads}) / \text{Total number of threads}) * \text{Weight} \geq 100$, the Availability Manager is notified. For example, if a task has four threads, two of which are blocked, the Availability Manager is notified only if the weight is configured as 200 or if all the threads become blocked. The default value of 100 means that only when all the threads are blocked, the Availability Manager is notified.	100
APR_GAT2E_MAINTENANCE_OF_TIME_MNGR_IWD_TAS_K_HANGUP	The time period for which the Maintenance of Time Manager task threads can run without sending a heartbeat to the Internal Watchdog. If the task threads do not send a heartbeat to the Internal Watchdog within this time period, the task is considered blocked.	300

Variable Name	Description	Valid Values/ Default Value
APR_GAT2E_MAINTENANCE_OF_TIME_MNGR_IWD_TASK_WEIGHT	<p>The weight of the Maintenance of Time Manager task. If $(\langle \text{Number of blocked threads} \rangle / \langle \text{Total number of threads} \rangle) * \text{Weight} \geq 100$, the Availability Manager is notified. For example, if a task has four threads, two of which are blocked, the Availability Manager is notified only if the weight is configured as 200 or if all the threads become blocked.</p> <p>The default value of 100 means that only when all the threads are blocked, the Availability Manager is notified.</p>	100
APR_GAT2E_SESSION_MNGR_IWD_TASK_HANGUP	<p>The time period for which the Session Manager task threads can run without sending a heartbeat to the Internal Watchdog. If the task threads do not send a heartbeat to the Internal Watchdog within this time period, the task is considered blocked.</p>	300
APR_GAT2E_SESSION_MNGR_IWD_TASK_WEIGHT	<p>The weight of the Session Manager task. If $(\langle \text{Number of blocked threads} \rangle / \langle \text{Total number of threads} \rangle) * \text{Weight} \geq 100$, the Availability Manager is notified. For example, if a task has four threads, two of which are blocked, the Availability Manager is notified only if the weight is configured as 200 or if all the threads become blocked.</p> <p>The default value of 100 means that only when all the threads are blocked, the Availability Manager is notified.</p>	100
APR_GAT2E_STATISTICS_MNGR_IWD_TASK_HANGUP	<p>The time period for which the Statistics Manager task threads can run without sending a heartbeat to the Internal Watchdog. If the task threads do not send a heartbeat to the Internal Watchdog within this time period, the task is considered blocked.</p>	300
APR_GAT2E_STATISTICS_MNGR_IWD_TASK_WEIGHT	<p>The weight of the Statistics Manager task. If $(\langle \text{Number of blocked threads} \rangle / \langle \text{Total number of threads} \rangle) * \text{Weight} \geq 100$, the Availability Manager is notified. For example, if a task has four threads, two of which are blocked, the Availability Manager is notified only if the weight is configured as 200 or if all the threads become blocked.</p> <p>The default value of 100 means that only when all the threads are blocked, the Availability Manager is notified.</p>	100
APR_GAT2E_UPDATE_MNGR_IWD_TASK_HANGUP	<p>The time period for which the Update Manager task threads can run without sending a heartbeat to the Internal Watchdog. If the task threads do not send a heartbeat to the Internal Watchdog within this time period, the task is considered blocked.</p>	300

Variable Name	Description	Valid Values/ Default Value
APR_GAT2E_UPDATE_MNGR_IWD_TASK_WEIGHT	The weight of the Update Manager task. If (<i><Number of blocked threads> / <Total number of threads></i>) * Weight ≥ 100 , the Availability Manager is notified. For example, if a task has four threads, two of which are blocked, the Availability Manager is notified only if the weight is configured as 200 or if all the threads become blocked. The default value of 100 means that only when all the threads are blocked, the Availability Manager is notified.	100
APR_GAT2E_SESSION_MNGR_NUM_OF_THREADS	The number of GAT2E Session Manager threads	1
APR_GAT2E_UPDATE_MNGR_TIMEOUT	The timeout for the GAT2E Update Manager (in milliseconds)	1000
APR_GAT2E_VIRTUAL_OR_SHARED_MEMORY	Indicates whether the process uses virtual or shared memory	VIRTUAL

For more information on the script, see the “Amdocs Batch Job Manager Application Script” section in the “Introduction” chapter.

Parameters

The following parameters must be passed with correct values to the activation script for the GAT2E process to run properly.

Parameter	Description	Default Value
APPLICATION_ID	The process instance name	N/A
CCF_FILE_NAME	The name of an XML file that contains the process configuration	APR1_GAT2E.ccf

Configuration Parameters

The configuration parameters (properties) of the GAT2E are defined in the database.

The following table shows the performance tuning (configuration) parameters of the GAT2E. These parameters are defined in the APR1_CONF_SECTION_PARAM table using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*).

Some of the parameters are optional and have default values.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
GAT2E	CONFIG	BYTES_HIGH_WATER_MARK	102400	The maximum number of bytes that can be sent to Event Server without receiving a response
GAT2E	CONFIG	ENABLE_DETAILED_STATISTICS	Yes	Specifies whether to add the statistics to the log file
GAT2E	CONFIG	MAX_EVENT_IN_FULL_LOAD_CAC HE	300	The maximum number of events for a table to save after a full load for future reuse
GAT2E	CONFIG	MAX_EVENT_SIZE_IN_BYTES	1024	The maximum size of a message in bytes
GAT2E	CONFIG	MAX_EVENT_SIZE_IN_RECORDS	100	The maximum number of records in a message
GAT2E	CONFIG	RECORDS_HIGH_WATER_MARK	100	The maximum number of records (not messages) that can be sent to the Event Server without receiving a response
GAT2E	CONFIG	RESEND_TIMEOUT_SEC	3	The amount of time in seconds to wait for a response before resending the event
GAT2E	CONFIG	STATISTICS_TIMEOUT_SEC	15	The interval at which statistics are to be taken
GAT2E	config	VIRTUAL_OR_SHARED_MEMORY_U SED	\${APR_GAT2E_ VIRTUAL_OR_ SHARED_MEM ORY}	Indicates whether the process uses virtual or shared memory

In addition, the process uses Light Tracer configuration parameters, which are defined in the ADJ1_CONF_SECTION_PARAM table. For more information, see [Light Tracer Parameters](#) in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

For each process instance, an entry with the name of the process instance must be created in the APR1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
GAT2E137	GAT2E

Any parameter to be overridden at the process instance level must be added to the APR1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
GAT2E137	CONFIG	STATISTICS_TIMEOUT_SEC	10

For more information on parameters and the properties mechanism, see the “Configuration Parameter Mechanism” appendix.

Configuration Files

The GAT2E requires a configuration file (a .ccf file) that is defined via an internal configuration tool. This XML file is supplied as one of the executable files of the product. It must not be changed.

The environment variables that appear in the .ccf file and can be customized are defined in the op_adj_env_sh script (for more information, see the “Environment Variables” section in this chapter).

Database Connections

Currently, GAT2E only works with generic applicative tables that reside in the Resource database.

Admin Commands

GAT2E supports the following admin commands. For information on how to execute the commands, see the “Handling Admin Commands” section in the “High Availability” chapter.

Command	Description	Parameters
GAT2E_SUSPEND	Stops loading data to all Event Servers in its logical cluster.	N/A
GAT2E_RESUME	Resumes loading data to all Event Servers in its logical cluster.	N/A
GAT2E_SUSPEND_TARGET	Stops loading data to a particular Event Server.	--ES <Event Server name, for example, ES100>
GAT2E_RESUME_TARGET	Resumes loading data to a particular Event Server.	--ES <Event Server name, for example, ES100>
GAT2E_REFRESH	Refreshes the configuration of the generic applicative tables. If new tables were added, starts their processing. This command also refreshes the versions of existing tables.	N/A
GAT2E_FULL_CMD	Starts a full load according to the command parameters. When the full load for a specific Event Server is finished, GAT2E sends the corresponding admin command to the Availability Manager.	--table <Table name> --ES <Event Server name> where: <ul style="list-style-type: none">■ Table name is either a list of tables with ‘,’ as a delimiter, or ‘all’.■ Event Server name is either a list of Event Server names (for example, ES100) with ‘,’ as a delimiter, or ‘all’.
SET_LIGHT_TRACER_COMMAND	Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters	<ul style="list-style-type: none">■ Activation parameters:<ul style="list-style-type: none">• <i>turnTracer “on/off”</i> – Activates or stops the Light Tracer• <i>--turnAssert “on/off”</i> – Activates or stops Assertions• <i>--turnLogOnTrace “on/off”</i> – Activates or stops writing log messages in Light Tracer files• <i>--turnFlowTracer/turnAll “on/off”</i> – Activates or stops tracing a single flow or event that is transferred between machines

Command	Description	Parameters
		<ul style="list-style-type: none"> • <code>--turnAll "on/off"</code> – Activates or stops all of the above. <p>The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter).</p> <ul style="list-style-type: none"> ▪ <code>--moduleID "ALL"</code> ▪ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • <code>--threadInfo "Y/N"</code> – Specifies whether trace or assertion messages include the thread information. • <code>--timestampInd "Y/N"</code> – Specifies whether trace or assertion messages include the time stamp. • <code>--contextInd "Y/N"</code> – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • <code>--noOfMessages "<number>"</code> – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down. • <code>--period "5/10/20/60/120/480/1440"</code> – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated. <p> Note: If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached.</p> <ul style="list-style-type: none"> • <code>--assertAbortInd "Y/N"</code> – Specifies whether the application is to be aborted if the assertion condition fails. ▪ <code>--commandType "changeBufferSize"</code> – Resizes the Light Tracer buffer at run time. • <code>--bufferSize "<number>"</code> – The new size for the Light Tracer buffer.

Command	Description	Parameters
SET_TRACE_SQL_COMMAND	<p>Activates or stops SQL tracing at runtime. The command name with its parameters must be enclosed in quotation marks.</p> <p><i>Example:</i> ADJ1_Send_Admin_Command_Sh illin1027 <process instance> "SET_TRACE_SQL_COMMAND --traceSqlOn \"yes\" --duration \"1\""</p>	<p>--traceSqlOn "<yes or no>" --duration "<minutes>"</p> <p>If the IS_DURATION_REQUIRED environment variable is set to "Y", the --duration parameter is mandatory. If the duration specified in the command exceeds the allowed maximum value, SQL tracing is activated for the maximum allowed period of time only (60 minutes).</p>  <p>Note: The SHUT_DOWN_CHECK_FREQ parameter of the Light Tracer in the ADJ1_CONF_SECTION_PARAM table determines the frequency (defined in the number of messages) at which the SQL Trace Manager checks whether the period specified in the --duration parameter has passed. Therefore, SQL tracing is stopped after the specified duration or after this number of messages has been reached, whichever comes later.</p>
SWITCH_LIGHT_TRACER_ON ALL	Activates the Light Tracer without the ability to specify the mode or send parameters	N/A
SWITCH_LIGHT_TRACER_OFF ALL	Stops the Light Tracer	N/A
CPF_GracefulShutdownCommand	Shuts down GAT2E gracefully.	N/A

Examples:

- ADJ1_Send_Admin_Command_Sh 10.232.53.45 GAT2E 137 GAT2E_SUSPEND ES100
- ADJ1_Send_Admin_Command_Sh 10.232.53.45 GAT2E 137 GAT2E_RESUME ES100

Recovery Instructions

In the case of a failure, no special actions are required. GAT2E receives the latest status from each Event Server and proceeds accordingly.

Important Remarks

This section provides additional information on GAT2E.

Applicative Errors

If invalid data from GAT2E causes an applicative error in an Event Server, a new record is added to ADJ1_GAT_ERRORS table with the table ID and error description.

Interface with Availability Manager

GAT2E behaves the same when it works with or without the Availability Manager.

It attempts to connect to all the Event Servers in its logical cluster as specified in the configuration tables.

13 ADJ1RRPSRV – Rerate Prepare

This chapter describes the Rerate Prepare process.

Description

The Rerate Prepare process initiates the rerating procedure for customers whose balance or accumulators may be incorrect.

The process can work in the following modes:

- Ongoing
- On Demand

Process Type

Daemon

Run Frequency

Run frequency depends on the running mode:

- *Ongoing* – Runs continuously
- *On Demand* – Runs by request

Activation



Caution: When the home database is up after a failure, the operator must not activate the Rerate Prepare process until the Copy Cycle Usage process, which copies the data from the alternate database to the home database, is complete.

Command Line:	ADJ1_RRP_Daemon_Shell_Script -n <APPLICATION_ID> -c "-f <CCF_FILE_NAME>" -e <EWD_EXE>
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_RRP_Daemon_Shell_Script
Executable Name:	gcpf1fwcApp

Example:

ADJ1_RRP_Daemon_Shell_Script -n RRP1600 -c "-f APR1_RRP.ccf" -e gn1avm_ewd

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> RRP \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

ADJ1_Send_Admin_Command_Sh hpbl820 RRP 1600
CPF_GracefulShutdownCommand

Affected Applications

The process affects the following applications:

- *Event Server* – Receives events that are to be rerated from the Rerate Prepare process
- *File2E* – Receives files with events that were reguided and are to be sent to the correct Event Servers for processing

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*

ADJ1RRPSRV_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1RRPSRV_RRP1600_20081109_130927_1.log

- *Console log files:*

ADJ1_RRP_Daemon_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1_RRP_Daemon_inner_Sh_RRP1600_20081109_130927.log

- *Operational log files:*

ADJ1RRPSRV_<APPLICATION_ID>_\${ABP_MARKET}_<Date>_<Time>.log

Example:

ADJ1RRPSRV_RRP1600_M3G_20081109_130927.log

- *Environment variable log files:*

ADJ1_RRP_Daemon_Shell_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1_RRP_Daemon_Shell_Sh_RRP1600_20081109_130927.log

- *Latency log files:*

ADJ1RRPSRV_LATENCY_REPORT_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1RRPSRV_LATENCY_REPORT_RRP1600_20081109_130927_1.log

- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG
- *Environment variable log files* – \$ABP_LOG
- *Latency log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter)

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands)
- *Operational log files* – The output of operational scripts
- *Environment variable log files* – Environment variables used by the process

- *Latency log files* – The following comma-separated details (with a header) about each session rerated by Rerate Prepare and the Event Server:
 - Rerate session ID
 - Counts of events per status:
 - *Total*
 - *Rated*
 - *Rejected*
 - *Reguided*
 - *Dropped*
 - *Updated* – The number of events that were updated in the database after rerating
 - *Rolled* – The number of events that were rolled to the next cycle after rerating
 - Time measurements for each logical task within Rerate Prepare:
 - *Total Handling Time* – The total handling time of the session.
 - *Pending* – The time spent by the Rerate Prepare process in a pending state.
 - *Init* – The time it took the Event Server to check whether it can handle this session.
 - *Quarantine* – The time for which the Event Server did not process this session because the subscriber was locked. This situation can occur during end-of-cycle processing, for example, if two or more Rerate Prepare for End of Cycle processes are connected to the same Event Server.
 - *Extract* – The time it took the Rerate Prepare process to extract the data after opening a cursor in the APE1_RATED_EVENT table.
 - *Open* – The time it took to prepare the session for rerate by the Event Server, from creating a request until sending the events of a session to the Event Server.
 - *Rerate* – The time it took the Rerate Prepare and the Event Server to rerate the session, starting from the time when the events of the session were sent to the Event Server.
 - *Reguide* – The time it took to reguide the session, that is, to transmit the session to another Event Server.
 - *Radar* – The time it took the Rerate Prepare process to report the data in Pre-Balance reporting.
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Output Files

This section describes the output files of the process.

Reguided Events

This section describes the Reguided Events file.

Name

The Rerate Prepare process produces Reguided Events files, which follow this naming convention:

%PrefixName_%FileName_%CustomerID_%SubscriberID_%ProcessID_%ThreadID_%
n.dat

where:

- *PrefixName* – A configurable string defined in the REGUIDE_OUTPUT_FILE_PREFIX configuration parameter. If this parameter is missing, the default ‘RRP’ string is used.
- *FileName* – A configurable string defined in the REGUIDE_OUTPUT_FILE_NAME configuration parameter. If this parameter is missing, the default ‘ReguidedEvents’ string is used.
- *CustomerID* – The ID of the customer to which the reguided events belong.
- *SubscriberID* – The ID of the subscriber to which the reguided events belong. For reguided events that took place at the customer level, the default value is ‘0’ (zero).
- *ProcessID* – The process ID of the current Rerate Prepare process.
- *ThreadID* – The ID of the thread that created the current output file.
- *n* – A runtime counter that prevents the creation of files with the same name.

Example:

RRP_ReguidedEvents_123456789_123456789_10_1_0.dat

Location

The file is located under the directory defined in the REGUIDE_OUTPUT_DIR_NAME configuration parameter.

Contents

The output file contains reguided events in the native OCI format.

Rerated Events

This section describes the Rerated Events file.

Name

The Rerate Prepare process produces Rerated Events files, which follow this naming convention:

rr_events_%ProcessID_%n.dat

where:

- *ProcessID* – The process ID of the current Rerate Prepare process
- *n* – A runtime counter that prevents the creation of files with the same name

Example:

rr_events_10_0.dat

Location

The file is located under the directory defined in the EXTRACT_OUTPUT_DIR_NAME configuration parameter.

Contents

The output file contains rerated events in the native OCI format.

Flow

This section describes the process flow of the Rerate Prepare process in various modes.

Population Tables

The Rerate Prepare process uses two rerate population tables:

- *Intermediate* – Subscriber Rerate (APE1_SUBSCRIBER_RERATE)
- *Summarized* – Rerate Population (APE1_RERATE_POPULATION)

The intermediate Subscriber Rerate table can be populated with data from many different processes. If a process assumes that one of its operations may invalidate the current accumulator value, it inserts the details of the related customer or subscriber into the intermediate table. Thus, one customer or subscriber may appear several times in the intermediate table. Moreover, one customer or subscriber may appear with different rerate levels: rerate, reguide, or prerate.

The Rerate Prepare process summarizes these records and inserts only one record per customer, with the highest rerate level found in the intermediate table for this customer, into the summarized Rerate Population table.

Mid-Cycle Rerate

This section describes the process flow for Ongoing Rerate and Rerate on Demand.

These two modes differ in the trigger for rerating:

- *Ongoing Rerate* – Rerating can be started only if the Event Server is not under a heavy load. It is performed for all open cycles handled by this Event Server. This rerating mode is continuous.
- *Rerate on Demand* – Rerating is started only when the process receives a command from the operator. In this mode, rerating is limited to a specific cycle. When rerating in this mode is finished, the Rerate Prepare process becomes idle.

Process Start-up

At start-up, the mid-cycle Rerate Prepare process rolls all data of customers or subscribers for whom rerating has not been completed from the summarized Rerate Population (APE1_RERATE_POPULATION) table to the intermediate Subscriber Rerate (APE1_SUBSCRIBER_RERATE) table. This is done because this data may already be obsolete.

If at start-up, the Rerate Prepare process receives an indication from the Availability Manager that one of its cycles was moved to an alternate database and is still being copied back to the home database, it shuts down and sends a message to the Availability Manager so that the Availability Manager does not recover it.

Main Loop

The Mid-Cycle Rerate Prepare process (in Ongoing Rerate and Rerate on Demand) performs the following as part of the main loop:

1. Brings new rerate population data from the intermediate Subscriber Rerate table to the summarized Rerate Population table. In Ongoing mode, the data can belong to any cycle that is handled by the related Event Server. In Rerate on Demand, all the data belongs to a specific cycle.

The following considerations apply:

- The system maintains a blacklist table for customers that must be excluded from rerate in a given cycle and run mode. The Rerate Prepare process does not select such population for rerate. For example, a communication service provider can use this table to determine whether postpaid customers with open non-persistent sessions must be rerated.
 - The Subscriber Rerate table includes a counter that counts the number of attempts made to rerate a subscriber's events. If this number is greater than a predefined value, the subscriber is not selected for rerate until the counter is changed manually to a number that is lower than the predefined value.
2. Based on the data in the Rerate Population table, initiates rerate sessions in the Event Server until the maximum number of sessions is reached. A rerate session represents an interaction between the Rerate Prepare process and the Event Server for a finite period of time. As a result of this interaction, a customer's or subscriber's events are rerated.

3. Checks whether the Event Server is available for rerate. If the Event Server is in one of the following states, rerating does not start:
 - Persistence Writer throttling
 - Session throttling
 - Queue throttling
 - Recovery from a failover
4. Selects the relevant events from the database, writes them into a memory-mapped file, and notifies the Event Server of their location.
5. During rerating, updates its progress in the summarized Rerate Population table.
6. After processing each session, writes its data into the latency log.
7. Maintains the maximum number of open sessions by replacing the finished rerate session with a new one.
8. In Ongoing mode, if the summarized Rerate Population table is empty, brings new rerate population data from the intermediate Subscriber Rerate table.

Process Shutdown

Because the Rerate Prepare process is a daemon, it only shuts down if it receives a specific manual request to do so. Before shutting down, the Rerate Prepare process completes all ongoing rerate sessions.



Note: If at the end of a cycle, the Rerate Prepare process receives the Refresh Cycle admin command for a cycle code and cycle month under its responsibility, it shuts down immediately (kill -9). When the process is restarted (usually, by the External Watchdog), it loads the updated Cycle State table, identifies the cycle instance as closed and ignores the requests for this cycle instance. These requests are handled by the Rerate Prepare for End of Cycle job.

Environment Variables

Rerate Prepare uses the following environment variables that are initialized by the op_adj_env_sh script.

Variable Name	Description	Valid Values/ Default Value
APR_EXTRACT_EVENTS_BULK_NUM	The number of records fetched from the APE1_RATED_EVENT table at once	100
APR_EXTRACT_MAX_FILE_SIZE	The maximum size of files that contain the data extracted from the APE1_RATED_EVENT table	1000000

Variable Name	Description	Valid Values/ Default Value
APR_NON_REGUIDABLE_EVENT_IDS	A comma-separated list of IDs of events that cannot be reguided (for example, technical events). If this parameter is populated with event IDs, only the specified events are considered non-reguidable. If this parameter is empty, events whose resource ID is ‘0’ or empty are considered non-reguidable.	0
APR_POP_PREP_BULK	The number of records moved to the summarized APE1_RERATE_POPULATION table from the intermediate APE1_SUBSCRIBER_RERATE table	100
APR_POP_RETRIEVAL_BULK	The number of records loaded to the memory of the process at once	20
APR_RR_LATENCY_ENABLED	Indicates whether messages must be printed to the latency log	false
APR_RRP_POPULATION_RET_TIMEOUT_UT	The timeout between one population retrieval from the APE1_SUBSCRIBER_RERATE table and the next (in seconds)	1000
APR_RRP_REGUIDE_ROUTING_CRITERIA	The routing criteria for the Reguided Events file	rc1
APR_RRPOG_DATA_EXTRACTOR_NUM_OF_THREADS	The number of Data Extractor threads	1
APR_RRPOG_MAX_CUSTOMERS_THRESHOLD	The number of customers for which events must be extracted.	1
APR_RRPOG_MAX_FILE_NUMBER_PER_SESSION	The maximum number of files with extracted data that each rerate session can use at the same time	20
STATEMENT_HINT	The hint for the SQL statement used for Event Extracts	/*+ USE_CONCAT */

For more information on the script, see the “Amdocs Batch Job Manager Application Script” section in the “Introduction” chapter.

Parameters

The following parameters must be set for the Rerate Prepare process to run properly.



Note: In Mid-Cycle Rerate, cycle-related information is retrieved from the configuration of the Event Server that is bound with Rerate Prepare.

Parameter	Description	Default Value
APPLICATION_ID	The process instance	RRP1600
CCF_FILE_NAME	An XML file that contains the relevant configuration	APR1_RRP.ccf

Configuration Parameters

The configuration parameters for the Rerate Prepare process are defined in the APR1_CONF_SECTION_PARAM table and can be modified using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*).

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	config	ACTIVITY_SOURCE_DEFAULT	RR	The default activity source.
APR	config	DUPCHECK_MODE_IS_IOT	N	Indicates whether the Duplicate Check mechanism works with the special index-organized table (IOT).
APR	config	RADAR_ENABLED	\${RADAR_ENABLE_D}	Indicates whether Pre-Balance reporting is enabled. The value of this parameter is an environment variable that must be defined in runtime environments. Its default value is 'N'.
RRP	config	EXPIRED_SESSION_DELETE_HINT_1	/* */	The hint for the SQL statement used for deleting expired sessions.
RRP	config	EXPIRED_SESSION_DELETE_HINT_2	/* */	The hint for the SQL statement used for deleting expired sessions.
RRP	config	EXPIRED_SESSION_INSERT_HINT	/* */	The hint for the SQL statement used for inserting expired sessions.
RRP	config	NEW_SESSION_DELETE_HINT_1	/* */	The hint for the SQL statement used for deleting new sessions.
RRP	config	NEW_SESSION_DELETE_HINT_2	/* */	The hint for the SQL statement used for deleting new sessions.
RRP	config	NEW_SESSION_INSERT_HINT	/* */	The hint for the SQL statement used for inserting new sessions.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
RRP	config	NON_REGUIDABLE_EVENT_IDS	\${APR_NON_REGUIDABLE_EVENT_IDS }	A comma-separated list of IDs of events that cannot be reguided (for example, technical events). If this parameter is populated with event IDs, only the specified events are considered non-reguidable. If this parameter is empty, events whose resource ID is '0' or empty are considered non-reguidable.
RRP	config	OG_RETRIEVER_HINT	/* */	The hint for selecting the population from the APE1_RERATE_POPULATION table.
RRP	config	SEGMENT_DELETE_HINT	/* */	The hint for the SQL statement used for deleting segments.
RRP	config	SEGMENT_INSERT_HINT	/* */	The hint for the SQL statement used for inserting segments.
RRP	config	UNCOMPLETED_SESSION_DELETE_HINT_1	/* */	The hint for the SQL statement used for deleting incomplete sessions.
RRP	config	UNCOMPLETED_SESSION_DELETE_HINT_2	/* */	The hint for the SQL statement used for deleting incomplete sessions.
RRP	config	UNCOMPLETED_SESSION_INSERT_HINT	/* */	The hint for the SQL statement used for inserting incomplete sessions.
RRP	config	VIRTUAL_OR_SHARED_MEMORY_USED	VIRTUAL	Specifies whether the process uses the shared or virtual memory for routing information.
RRP	EXTRACT_DATA_PARAMS	ADDITIONAL_WHERE_CLAUSE	AND RE.EVENT_STATE = 'N'	Specifies the additional WHERE clause for the SQL statement used for Event Extracts.
RRP	EXTRACT_DATA_PARAMS	EXTRACT_EVENTS_BULK_NUM	\${APR_EXTRACT_EVENTS_BULK_NUM }	The number of records fetched from the APE1_RATED_EVENT table at once.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
RRP	EXTRACT_DATA_PARAMS	EXTRACT_MAX_FILE_SIZE	\${APR_EXTRACT_MAX_FILE_SIZE}	The maximum size of files that contain the data extracted from the APE1_RATED_EVENT table.
RRP	EXTRACT_DATA_PARAMS	EXTRACT_OUTPUT_DIR_NAME	\${ABP_AP4_ROOT}/interfaces/output/rerate	The directory where the files that contain rerated events are created.
RRP	EXTRACT_DATA_PARAMS	REGUIDE_OUTPUT_DIR_NAME	\${ABP_AP4_ROOT}/interfaces/output/reguide	The directory where the files that contains reguided events are created.
RRP	EXTRACT_DATA_PARAMS	REGUIDE_OUTPUT_FILE_NAME	ReguidedEvents	The root of the file name of the file that contains reguided events.
RRP	EXTRACT_DATA_PARAMS	REGUIDE_OUTPUT_FILE_PREFIX	RRP_	The prefix of the name of the file that contains reguided events.
RRP	Logger.ADJ.RERATE.LATENCY	Enable	Y	Enables the latency log output. This parameter must always be set to 'Y'. The RR_LATENCY_REPORT_ENABLED parameter actually controls the printing of messages to the log.
RRP	Logger.ADJ.RERATE.LATENCY	Outputs	RRPGNRL	The output object assigned to the latency log.
RRP	Logger.ADJ.RERATE.LATENCY	Severity	0	The message severity threshold for the latency log.
RRP	LOGGER.output.RRPGNR	BufferSize	\${BUFFER_SIZE}	The size of the temporary buffer of the latency log.
RRP	LOGGER.output.RRPGNR	Filename	\${ABP_AP4_LOG}/\${GN1_TASK_NAME}_LATENCY_REPORT_\${APPLICATION_ID}_%D_%T_%n.log	The name of the latency log file.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
RRP	LOGGER.output.RRPGNR	FlushPeriod	\${FLUSH_PERIOD}	The interval at which the data from the temporary buffer is flushed to a file.
RRP	LOGGER.output.RRPGNR	ImmediateFlush	\${IMMEDIATE_FLUSH}	Indicates whether the data is flushed to a file immediately and not periodically.
RRP	LOGGER.output.RRPGNR	Layout	RR_Latency_Layout	The layout type that the latency log uses to format messages.
RRP	LOGGER.output.RRPGNR	MaxFileSize	\${MAX_FILE_SIZE}	The maximum size of a latency log file.
RRP	LOGGER.output.RRPGNR	Mode	\${MODE}	The working mode of the latency log: synchronous or asynchronous.
RRP	RR_PARAMS	RR_LATENCY_REPORT_ENABLED	\${APR_RR_LATENCY_ENABLED}	Indicates whether the latency log is initially turned on.
RRP	SESSION_MANAGER_PARAMS	RERATE_SUSPENSION_PERIOD	300	The number of seconds before the process attempts to establish a new rerate session if the previous session was rejected with the Server Too Busy error. (Relevant for the Mid-Cycle mode only.)
RRP	SESSION_MANAGER_PARAMS	SESSION_QUARANTINE_PERIOD	5	The number of seconds before the process attempts to re-establish a failed rerate session if the reason for the failure reason was Customer Locked.
RRP	SESSION_PROGRESS_TRACKER_PARAMS	MAX_SESSION_INIT_TIME	300	The maximum time for the Event Server to initialize a session.
RRP	SESSION_PROGRESS_TRACKER_PARAMS	MAX_SESSION_OPEN_TIME	300	The maximum time for the Event Server to open a session.
RRP	SESSION_PROGRESS_TRACKER_PARAMS	MAX_SESSION_UPDATE_TIME	300	The time interval for the Event Server to send an update of the progress of the rerate session.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
RRP_OG	EXTRACT_DATA_PARAMS	EXTRACT_MAX_CUSTOMERS_THRESHOLD	\${APR_RRPOG_MAX_CUSTOMERS_THRESHOLD}	The maximum number of customers whose rerate data is retrieved from the APE1_RATED_EVENT table at once.
RRP_OG	EXTRACT_DATA_PARAMS	EXTRACT_MAX_FILE_NUMBER_PER_SESSION	\${APR_RRPOG_MAX_FILE_NUMBER_PER_SESSION}	The maximum number of files with extracted data that each rerate session can use at the same time.
RRP_OG	EXTRACT_DATA_PARAMS	STATEMENT_HINT	\${STATEMENT_HINT}	The hint for the SQL statement used for Event Extracts.
RRP_OG	PREPARATOR_PARAMS	MAX_NUM_OF_RERATE_TRIES	3	A customization option that enables changing the default maximum number of times that a request can be selected for rerate. The value must be between 1 and 99. If this parameter is set to a higher value, 99 is used by default.
RRP_OG	PREPARATOR_PARAMS	POPULATION_PREPARATION_BULK	\${APR_POP_PREP_BULK}	The number of records moved to the summarized Rerate Population table from the intermediate Subscriber Rerate table.
RRP_OG	RETRIEVER_PARAMS	POPULATION_RETRIEVAL_BULK	\${APR_POP_RETRIEVAL_BULK}	The number of records loaded to the memory of the process at once.
RRP_OG	RETRIEVER_PARAMS	WORK_MODE	ON_GOING	The working mode of the process. Valid values: <ul style="list-style-type: none">■ ON_GOING■ ON_DEMAND■ EOC

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
RRP_OG	RR_PARAMS	BANNED_CYCLES_LIST	0	Lists the cycle codes of all technical cycles separated by a comma (‘,’). These cycles are not to be processes by the Ongoing Rerate.
RRP_OG	RR_PARAMS	MAX_SESSION_COUNT	20	The maximum number of sessions that the process can establish against the Event Server.

In addition, the process uses Light Tracer configuration parameters, which are defined in the ADJ1_CONF_SECTION_PARAM table. For more information, see [Light Tracer Parameters](#) in the “[Configuration Parameters](#)” section in the “[ADJ1EVENTSRV – Event Server](#)” chapter.

For each process instance, an entry with the name of the process instance must be created in the APR1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
RRP1600	RRP

Any parameter to be overridden at the process instance level must be added to the APR1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
RRP1600	EXTRACT_DATA_PARAMS	REGUIDE_OUTPUT_FILE_PREFIX	RG_

For more information on parameters and the properties mechanism, see the “[Configuration Parameter Mechanism](#)” appendix.

Configuration Files

The Rerate Prepare process requires a configuration file (.ccf) that is defined via an internal configuration tool.

It also requires parameters from Audit & Control, which appear in the configuration file.

The following parameters can be configured in the APR1_RRP.ccf file.

Property Name	Related Internal Module	Description	Default Value
Data Group	Reguider strategy	The data group of the output file. This parameter is used to distinguish between files that are to be processed by different processes of the same type. It is a column in the AC1_CONTROL table.	4444
File Alias	Reguider strategy	The file alias of the output file.	TCOCIUSAGE
File Union	Reguider strategy	Indicates whether the Event Server must check for duplicates of the event.	TCOCIUSAGE
Routing Criteria	Reguider strategy	The number of instances of each acceptor type. Each acceptor handles one file.	rc1

The environment variables that appear in the .ccf file and can be customized are defined in the op_adj_env_sh script (for more information, see the “[Environment Variables](#)” section in this chapter).

Database Connections

During initialization, the process connects to many reference tables. For a complete description of each table, see *Turbo Charging Data Model*.

After initialization, the Rerate Prepare process connects primarily to the following tables.

Database Type	Table Name	Connection Type (Update, Insert, and so on)	Comments
Customer	APE1_RERATE_POPULATION	Select, Update, Delete	N/A
Customer	APE1_SUBSCRIBER_RERATE	Select, Update, Delete	N/A
Usage	AC1_CONTROL	Insert	N/A
Usage	APE1_EVENT_DUP_KEYS	Delete	N/A
Usage	APE1_EVENT_DUP_KEYS_EXT	Delete	This table is used only if the USE_NEW_IOT_TABLE environment variable is set to 'Y'.
Usage	APE1_RATED_EVENT	Update	It is assumed that the Usage database contains a synonym to the AC1_CONTROL table.
Usage	APE1_RR_BLACK_LIST	Select	N/A

Admin Commands

The Rerate Prepare process supports the following admin commands. For information on how to execute the commands, see the “Handling Admin Commands” section in the “High Availability” chapter.

Command	Description	Parameters
START_RERATE_DMND_COMMAND	Starts Rerate on Demand	--cycleCode “<Cycle code>” --cycleInstance “<Cycle instance>” -- cycleYear “<Cycle year>”
STOP_RERATE_DMND_COMMAND	Stops Rerate on Demand	--cycleCode “<Cycle code>” --cycleInstance “<Cycle instance>” -- cycleYear “<Cycle year>”
RESUME_RERATE_COMMAND	Resumes Ongoing Rerate	N/A
SUSPEND_RERATE_COMMAND	Suspends Ongoing Rerate	N/A
REFRESH_CYCLE_DATA_COMMAND	Refreshes cycle data from the ADJ1_CYCLE_STATE table	--cycleCode “<Cycle code>” --cycleInstance “<Cycle instance>”
REFRESH_ALL_GENCODE_AND_IMPL_TABLES_COMMAND	Refreshes generated libraries and reference data	N/A
RESUME_FROM_REFRESH_ALL_COMMAND	Enables Rerate Prepare to work with the new implementation after generated libraries and reference data have been refreshed	N/A
SET_LIGHT_TRACER_COMMAND	Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters	<ul style="list-style-type: none"> ■ Activation parameters: <ul style="list-style-type: none"> • <code>--turnTracer "on/off"</code> – Activates or stops the Light Tracer • <code>--turnAssert "on/off"</code> – Activates or stops Assertions • <code>--turnLogOnTrace "on/off"</code> – Activates or stops writing log messages in Light Tracer files • <code>--turnFlowTracer/turnAll "on/off"</code> – Activates or stops tracing a single flow or event that is transferred between machines • <code>--turnAll "on/off"</code> – Activates or stops all of the above. <p>The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter).</p>

Command	Description	Parameters
		<ul style="list-style-type: none"> ■ --moduleID “ALL” ■ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • --threadInfo “Y/N” – Specifies whether trace or assertion messages include the thread information. • --timestampInd “Y/N” – Specifies whether trace or assertion messages include the time stamp. • --contextInd “Y/N” – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • --noOfMessages “<number>” – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down. • --period “5/10/20/60/120/480/1440” – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated. ■  Note: If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached. • --assertAbortInd “Y/N” – Specifies whether the application is to be aborted if the assertion condition fails. ■ --commandType “changeBufferSize” – Resizes the Light Tracer buffer at run time. • --bufferSize “<number>” – The new size for the Light Tracer buffer.

Command	Description	Parameters
SET_TRACE_SQL_COMMAND	<p>Activates or stops SQL tracing at runtime.</p> <p>The command name with its parameters must be enclosed in quotation marks.</p> <p><i>Example:</i></p> <pre>ADJ1_Send_Admin_Command_Sh illin1027 <process instance> "SET_TRACE_SQL_COMMAND --traceSqlOn \"yes\" --duration \"1\""</pre>	<p>--traceSqlOn \"<yes or no>\" --duration \"<minutes>\"</p> <p>If the IS_DURATION_REQUIRED environment variable is set to 'Y', the --duration parameter is mandatory. If the duration specified in the command exceeds the allowed maximum value, SQL tracing is activated for the maximum allowed period of time only (60 minutes).</p>  <p>Note: The SHUT_DOWN_CHECK_FREQ parameter of the Light Tracer in the ADJ1_CONF_SECTION_PARAM table determines the frequency (defined in the number of messages) at which the SQL Trace Manager checks whether the period specified in the --duration parameter has passed. Therefore, SQL tracing is stopped after the specified duration or after this number of messages has been reached, whichever comes later.</p>
SWITCH_LIGHT_TRACER_ON ALL	Activates the Light Tracer without the ability to specify the mode or send parameters	N/A
SWITCH_LIGHT_TRACER_OFF ALL	Stops the Light Tracer	N/A
RR_LATENCY_REPORT_ENABLED	Activates the latency log	N/A
CPF_GracefulShutdownCommand	Shuts down the Rerate Prepare process gracefully	N/A

Examples:

- ADJ1_Send_Admin_Command_Sh hpbl820 RRP 1600 START_RERATE_DMND_COMMAND –cycleCode “100” –cycleInstance “9” –cycleYear “2008”
- ADJ1_Send_Admin_Command_Sh hpbl820 RRP 1600 RESUME_RERATE_COMMAND

Recovery Instructions

Rerun the process.

14 ADJ1RRPOPREP – Rerate Population Report

This chapter describes Rerate Population Report process.

Description

In the End-of-Cycle map, the Rerate Population Report process extracts customer population that is marked for rerate and prepares tasks for the Rerate Prepare for End of Cycle (ADJ1RRPEOC) process.

In the Undo map, the Rerate Population Report does not send the population to Amdocs Invoicing. It is used internally only to divide the population among multiple Rerate Prepare for End of Cycle processes.

Process Type

Batch job

Run Frequency

By request

Participation in Generic Maps

This section contains information on the generic maps in which the process participates.

Map Name

- End-of-Cycle (EOC)
- Undo

For more information about these maps, see *Amdocs Billing Operator Guide*.

Severity Level

Severity Level 2 – The map must hold until the job is fixed.

Activation

Command Line:	RunJobs ADJ1RRPOPREP BYREQ
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_ReratePopulationReport_Job_Sh
Executable Name:	gcpf1fwcApp

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> RRP_POPREP \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

ADJ1_Send_Admin_Command_Sh hpbl820 RRP_POPREP 265
CPF_GracefulShutdownCommand

Preceding Processes

The following job must run successfully before this process is activated in the End-of-Cycle map:

- *ADJ1CYCLEMNT* – Cycle Maintenance

Dependent Processes

The following processes can be run only after the successful completion of this process:

- In the End-of-Cycle map:
 - *ADJ1RRPEOC* – Rerate Prepare for End of Cycle
 - *ADJ1RCPEXT* – Customer Group Accumulators Extract
 - *ADJ1ACGRPLSTN* – Accumulator Group Listener
- In the Undo map:
 - *ADJ1RRPEOC* – Rerate Prepare for End of Cycle
 - *ADJ1EXTCLEANE* – Extract Clean-up

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*

ADJ1RRPOPREP_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1RRPOPREP_RRP_OP_REP265_20081109_130927_1.log

- *Console log files:*

ADJ1_ReratePopulationReport_Job_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1_ReratePopulationReport_Job_inner_Sh_RRP_OP REP265_20081109_130927.log

- *Operational log files:*

ADJ1RRPOPREP_<Job Record Name>_\${ABP_MARKET}_<Date>_<Time>.log

Example:

ADJ1RRPOPREP_BYREQ_M3G_20081109_130927.log

- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter)

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information.
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).
- *Operational log files* – The output of operational scripts.
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Output Files

This section describes the output files.

Name

ReratePopulationReportData_<DateTime>_<PID>_<ThreadID>_<FileNo>.dat

Location and Format

- *Location* – \$ABP_APP_ROOT/interfaces/output/rrp_rep
- *Format* – Semicolon-delimited file

Contents

The output file contains the customers that are marked for rerate.

Flow

In the End-of-Cycle map, the Rerate Population Report process performs the following steps:

1. Extracts the customer population that is marked for rerate according to the cycle code and cycle instance from the APE1_SUBSCRIBER_RERATE and APE1_RERATE_POPULATION tables and sends the list of customers to Amdocs Invoicing.
2. Inserts the information about the Oracle partition containing the rated events of the customers that are undergoing rerate in the specified cycle to the APR1_EOC_RERATE_TASKS table.

In the Undo map, the process only divides the population among multiple Rerate Prepare for End of Cycle processes by populating the APR1_EOC_RERATE_TASKS table with the Oracle partitions that are involved in the rerate for the specified cycle. In this map, it does not send the customer population to Amdocs Invoicing.

Environment Variables

The Rerate Population Report uses the following environment variables.

Variable Name	Description	Valid Values/ Default Value
ROUTING_CRITERIA	The routing criteria for registering files to Audit & Control	RPR1_TO_BL1

Parameters

The following parameters must be set for the process to run properly. When the job runs as part of a map, the Turbo Charging Cycle Bill Run script populates these parameters automatically. For more information on this script, see the “ADJ1_AP4_CycleBillRun_Sh – Turbo Charging Cycle Bill Run” chapter.

Parameter	Description	Default Value
APPLICATION_ID	The ID of the job that enables it to get its parameters.	N/A
CCF_FILE_NAME	An XML file that contains the relevant configuration.	N/A
CYCLE_CODE	The cycle code for which data is to be extracted.	N/A
CYCLE_INST	The cycle instance for which data is to be extracted.	N/A
CYCLE_YEAR	The cycle year for which data is to be extracted.	N/A
EXTR_INPUT_MAP_KEY	The input map key. This key is used as a filter. Only the record with this value appears in the DATA_GROUP field of the AC1_CONTROL table.	N/A
MAP_MODE	<p>The context in which the process is activated:</p> <ul style="list-style-type: none"> ■ EOC ■ UNDO <p>Other possible values are:</p> <ul style="list-style-type: none"> ■ RERUN ■ QA ■ NONE (if the job is run manually and not as part of a map) 	N/A
RUN_REQUEST_ID	The request that is used to expand the input map key and insert it as part of the data group into the AC1_CONTROL table.	N/A

Configuration Parameters

In Turbo Charging, the configuration parameters (properties) of the Rerate Population Report process are defined in the database.

The following table shows the performance tuning (configuration) parameters of the Rerate Population Report process. These parameters are defined in the APR1_CONF_SECTION_PARAM table using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*).



Note: Other parameters must not be changed.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
RRP_REP	FILE_OUTPUT	FILE_DIRECTORY	\${ABP_APP_ROOT}/interfaces/output/rp_rep	The default directory where the output file is created.
RRP_REP	FILE_OUTPUT	FILE_NAME	ReratePopulationReportData.dat	The naming convention for the output file.
RRP_REP	POPULATION_REPORT	CYCLE_CODE	\${CYCLE_CODE}	The cycle code for which the customers that are marked for rerate are extracted.
RRP_REP	POPULATION_REPORT	CYCLE_INSTANCE	\${CYCLE_INST}	The cycle instance for which the customers that are marked for rerate are extracted.
RRP_REP	POPULATION_REPORT	CYCLE_YEAR	\${CYCLE_YEAR}	The cycle year for which the customers that are marked for rerate are extracted.
RRP_REP	POPULATION_REPORT	DISTRIBUTE_WORK	\${POPREP_DISTRIBUTE_WORK}	The flag that indicates whether the Rerate Population Report or the Rerate Error Report distributes the work to the Rerate job, as follows: <ul style="list-style-type: none">■ Yes – Rerate Error Report■ No – Rerate Population Report

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
RRP_REP	POPULATION_REPORT	FILE_TYPE	\${FILE_TYPE}	<p>The type of files to insert into the AC1_CONTROL table:</p> <ul style="list-style-type: none"> ■ <i>Undo map</i> – NONE ■ <i>EOC map</i> – BEFORE_RERATE <p>This parameter indicates whether the population is sent to Amdocs Invoicing. It is populated with the correct value when Amdocs Invoicing triggers the map.</p>
RRP_REP	POPULATION_REPORT	INPUT_MAP_KEY	\${EXTR_INPUT_MAP_KEY}	The input map key. This key is used as a prefix for the DATA_GROUP field of the AC1_CONTROL table.
RRP_REP	POPULATION_REPORT	RUN_REQUEST_ID	\${RUN_REQUEST_ID}	The request that is used to expand the input map key and insert it as part of the data group into the AC1_CONTROL table.

In addition, the process uses Light Tracer configuration parameters, which are defined in the ADJ1_CONF_SECTION_PARAM table. For more information, see [Light Tracer Parameters](#) in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

For each process instance, an entry with the name of the process instance must be created in the APR1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
RRP_OP REP265	RRP_OP REP

Any parameter to be overridden at the process instance level must be added to the APR1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
RRP_OP REP265	FILE_OUTPUT	FILE_NAME	Example.dat

For more information on parameters and the properties mechanism, see the “Configuration Parameter Mechanism” appendix.

Configuration Files

The Rerate Population Report process requires a configuration file (.ccf) that is defined via an internal configuration tool.

Database Connections

The process connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	AC1_CONTROL	Insert
Usage	APE1_RERATE_POPULATION	Select
Usage	APE1_RATED_EVENT	Select
Usage	APE1_SUBSCRIBER_RERATE	Select
Usage	APR1_EOC_RERATE_TASKS	Insert

Admin Commands

The Rerate Population Report supports the following admin commands. For information on how to execute the commands, see the “Handling Admin Commands” section in the “High Availability” chapter.

Command	Description	Parameters
SET_LIGHT_TRACER_COMMAND	Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters	<ul style="list-style-type: none"> ■ Activation parameters: <ul style="list-style-type: none"> • <i>turnTracer “on/off”</i> – Activates or stops the Light Tracer • <i>--turnAssert “on/off”</i> – Activates or stops Assertions • <i>--turnLogOnTrace “on/off”</i> – Activates or stops writing log messages in Light Tracer files • <i>--turnFlowTracer/turnAll “on/off”</i> – Activates or stops tracing a single flow or event that is transferred between machines • <i>--turnAll “on/off”</i> – Activates or stops all of the above. ■ The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter). ■ <i>--moduleID “ALL”</i> ■ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • <i>--threadInfo “Y/N”</i> – Specifies whether trace or assertion messages include the thread information. • <i>--timestampInd “Y/N”</i> – Specifies whether trace or assertion messages include the time stamp. • <i>--contextInd “Y/N”</i> – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • <i>--noOfMessages “<number> ”</i> – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down.

Command	Description	Parameters
		<ul style="list-style-type: none"> • <code>--period "5/10/20/60/120/480/1440"</code> – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated. <p> Note: If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached.</p> <ul style="list-style-type: none"> • <code>--assertAbortInd "Y/N"</code> – Specifies whether the application is to be aborted if the assertion condition fails. ▪ <code>--commandType "changeBufferSize"</code> – Resizes the Light Tracer buffer at run time. • <code>--bufferSize "<number>"</code> – The new size for the Light Tracer buffer.
SWITCH_LIGHT_TRACER_ON ALL	Activates the Light Tracer without the ability to specify the mode or send parameters.	N/A
SWITCH_LIGHT_TRACER_OFF ALL	Stops the Light Tracer	N/A
CPF_GracefulShutdownCommand	Shuts down the Rerate Population Report gracefully.	N/A

Examples:

- `ADJ1_Send_Admin_Command_Sh hpbl820 RRP_POP REP 265 SET_LIGHT_TRACER_COMMAND -turnTracer "on" -moduleID "ALL" -threadInfo "y"`
- `ADJ1_Send_Admin_Command_Sh hpbl820 RRP_POP REP 265 SWITCH_LIGHT_TRACER_OFF ALL`

Recovery Instructions

Perform the following steps to rerun the process:

1. Delete the records according to the cycle code and cycle instance that you are running.
2. Rerun the job.

15 ADJ1RRPEOC – Rerate Prepare for End of Cycle

This chapter describes the Rerate Prepare for End of Cycle process.

Description

The Rerate Prepare for End of Cycle process initiates the rerating procedure for customers whose balance or accumulators may be incorrect at the time of cycle closure. The process works on closed cycles.

Process Type

Batch job

Run Frequency

Every cycle

Participation in Generic Maps

This section contains information on the generic maps in which the process participates.

Map Name

The process participates in the following operational maps:

- End-of-Cycle (EOC) map
- Undo map

For more information about these maps, see *Amdocs Billing Operator Guide*.

Severity Level

Severity Level 2 – The map must hold until the job is fixed.

Activation

Command Line:	ADJ1_RRP_EOC_Daemon_Shell_Sh -n RRPEOC1900 -c "-f APR1_RRP_EOC.ccf" <APPLICATION_ID> -c "-f <CCF_FILE_NAME>" -e <EWD_EXE>  Note: The -e parameter is optional. It is used in environments with Availability Manager.
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_RRP_EOC_Daemon_Shell_Sh
Executable Name:	gcpf1fwcApp

Example:

```
ADJ1_RRP_EOC_Daemon_Shell_Sh -n RRPEOC1900 -c "-f APR1_RRP_EOC.ccf"  
-e gn1avm_ewd
```

For more information about the activation scripts, see the “[Scripts](#)” section in the “[Introduction](#)” chapter.

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> RRPEOC \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

```
ADJ1_Send_Admin_Command_Sh hpbl820 RRPEOC 1900
CPF_GracefulShutdownCommand
```

Preceding Processes

The following jobs must run successfully before this process is activated:

- In the End-of-Cycle map:
 - *ADJ1RRPOPREP* – Rerate Population Report
- In the Undo map:
 - *ADJ1UHMARK4RD* – Undo Mark for Rerate
 - *ADJ1RRPOPREP* – Rerate Population Report

Dependent Processes

The following processes depend on the successful completion of this process:

- End-of-Cycle map:
 - *ADJ1ERRREP* – Rerate Error Report
 - *ADJ1CRA* – Copy Rolling Accumulators
 - *ADJ1EVGRPLSTN* – Event Group Listener
 - *ADJ1DISPENTFR* – Rerate Prepare for End of Cycle Finish Notification
 - *ADJ1PRCPEXT* – Customer Group Accumulators Extract Post Rerate
- Undo map:
 - *ADJ1CRA* – Copy Rolling Accumulators
 - *ADJ1ERRREP* – Rerate Error Report
 - *ADJ1DISPENTFR* – Rerate Prepare for End of Cycle Finish Notification
 - *ADJ1EXTCLEANE* – Extract Clean-up

Affected Applications

The process affects the following applications:

- *Event Server* – Receives events that are to be rerated from the Rerate Prepare process
- *File2E* – Receives files with events that were reguided and are to be sent to the correct Event Servers for processing

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*
ADJ1RRPEOC_<APPLICATION_ID>_%D_%T_%n.log
Example:
ADJ1RRPEOC_RRPEOC1900_20081109_130927_1.log
- *Console log files:*
ADJ1_RRP_EOC_Daemon_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log
Example:
ADJ1_RRP_EOC_Daemon_inner_Sh_RRPEOC1900_20081109_130927.log
- *Operational log files:*
ADJ1RRPEOC_<APPLICATION_ID>_\${ABP_MARKET}<Date>_<Time>.log
Example:
ADJ1RRPEOC_RRPEOC1900_M3G_20081109_130927.log
- *Environment variable log files:*
ADJ1_RRP_EOC_Daemon_Shell_Sh_<APPLICATION_ID>_<Date>_<Time>.log
Example:
ADJ1_RRP_EOC_Daemon_Shell_Sh_RRPEOC1900_20081109_130927.log
- *Latency log files:*
ADJ1RRPEOC_LATENCY_REPORT_<APPLICATION_ID>_%D_%T_%n.log
Example:
ADJ1RRPEOC_LATENCY_REPORT_RRPEOC1900_20081109_130927_1.log
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG
- *Environment variable log files* – \$ABP_LOG
- *Latency log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see [Light Tracer Parameters](#) in the “[Configuration Parameters](#)” section in the “[ADJ1EVENTSRV – Event Server](#)” chapter)

For information about the configuration parameters that define the location of log and trace files, see the “[Configuration Parameters](#)” section in the “[ADJ1EVENTSRV – Event Server](#)” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information.
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).
- *Operational log files* – The output of operational scripts.
- *Environment variable log files* – Environment variables used by the process.
- *Latency log files* – The following comma-separated details (with a header) about each session rerated by Rerate Prepare and the Event Server:
 - Rerate session ID
 - Counts of events per status:
 - *Total*
 - *Rated*
 - *Rejected*
 - *Reguided*
 - *Dropped*
 - *Updated* – The number of events that were updated in the database after rerating
 - *Rolled* – The number of events that were rolled to the next cycle after rerating

- Time measurements for each logical task within Rerate Prepare:
 - *Total Handling Time* – The total handling time of the session.
 - *Pending* – The time spent by the Rerate Prepare process in a pending state.
 - *Init* – The time it took the Event Server to check whether it can handle this session.
 - *Quarantine* – The time for which the Event Server did not process this session because the subscriber was locked. This situation can occur during end-of-cycle processing, for example, if two or more Rerate Prepare for End of Cycle processes are connected to the same Event Server.
 - *Extract* – The time it took the Rerate Prepare process to extract the data after opening a cursor in the APE1_RATED_EVENT table.
 - *Open* – The time it took to prepare the session for rerate by the Event Server, from creating a request until sending the events of a session to the Event Server.
 - *Rerate* – The time it took the Rerate Prepare and the Event Server to rerate the session, starting from the time when the events of the session were sent to the Event Server.
 - *Reguide* – The time it took to reguide the session, that is, to transmit the session to another Event Server.
 - *Radar* – The time it took the Rerate Prepare process to report the data in Pre-Balance reporting.
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Output Files

This section describes the output files of the process.

Reguided Events

This section describes the Reguided Events file.

Name

The Rerate Prepare process produces Reguided Events files, which follow this naming convention:

%PrefixName_%FileName_%CustomerID_%SubscriberID_%ProcessID_%ThreadID_%
n.dat

where:

- *PrefixName* – A configurable string defined in the REGUIDE_OUTPUT_FILE_PREFIX configuration parameter. If this parameter is missing, the default ‘RRP’ string is used.
- *FileName* – A configurable string defined in the REGUIDE_OUTPUT_FILE_NAME configuration parameter. If this parameter is missing, the default ‘ReguidedEvents’ string is used.
- *CustomerID* – The ID of the customer to which the reguided events belong.
- *SubscriberID* – The ID of the subscriber to which the reguided events belong. For reguided events that took place at the customer level, the default value is ‘0’ (zero).
- *ProcessID* – The process ID of the current Rerate Prepare process.
- *ThreadID* – The ID of the thread that created the current output file.
- *n* – A runtime counter that prevents the creation of files with the same name.

Example:

RRP_ReguidedEvents_123456789_123456789_10_1_0.dat

Location

The file is located under the directory defined in the REGUIDE_OUTPUT_DIR_NAME configuration parameter.

Contents

The output file contains reguided events in the native OCI format.

Rerated Events

This section describes the Rerated Events file.

Name

The Rerate Prepare process produces Rerated Events files, which follow this naming convention:

rr_events_%ProcessID%_n.dat

where:

- *ProcessID* – The process ID of the current Rerate Prepare process.
- *n* – A runtime counter that prevents the creation of files with the same name.

Example:

rr_events_10_0.dat

Location

The file is located under the directory defined in the EXTRACT_OUTPUT_DIR_NAME configuration parameter.

Contents

The output file contains rerated events in the native OCI format.

Flow

This section describes the process flow of the Rerate Prepare process in various modes.

Population Tables

The Rerate Prepare process uses two rerate population tables:

- *Intermediate* – Subscriber Rerate (APE1_SUBSCRIBER_RERATE)
- *Summarized* – Rerate Population (APE1_RERATE_POPULATION)

The intermediate Subscriber Rerate table can be populated with data from many different processes. If a process assumes that one of its operations may invalidate the current accumulator value, it inserts the details of the related customer or subscriber into the intermediate table. Thus, one customer or subscriber may appear several times in the intermediate table. Moreover, one customer or subscriber may appear with different rerate levels: rerate, reguide, or prerate.

The Rerate Prepare process summarizes these records and inserts only one record per customer, with the highest rerate level found in the intermediate table for this customer, into the summarized Rerate Population table.

End-of-Cycle Rerate

This section describes the process flow for End-of-Cycle Rerate.

Process Start-up

For the sake of scalability, a given cycle instance can be processed by more than one processing pair (a Rerate Prepare process and an Event Server). The work is divided among the various processing pairs according to the partitions of the Rated Events table. This division into the Oracle partitions enables achieving good rerate performance, while retrieving the rated events from the database and updating them there.

At start-up, the Rerate Prepare for End of Cycle process determines which Oracle partitions it is going to handle, by updating one or more rows of the APR1_EOC_RERATE_TASKS table with the system ID of the Rerate Prepare for End of Cycle process. Each record in this table contains the name of the Oracle partition and the range of customer segments that belong to this partition.

Each time the Rerate Prepare for End of Cycle process finishes rerating one or few partitions, it retrieves the next partitions that have not yet been handled.

Main Loop

The Rerate Prepare for End of Cycle process performs the following as part of the main loop:

1. Brings new rerate population data from the intermediate Subscriber Rerate table to the summarized Rerate Population table. In this mode, all such data belongs to the cycle that is being closed.

The system maintains a blacklist table for customers that must be excluded from rerate in a given cycle and run mode. If the specified run mode is EOC, the Rerate Prepare for End of Cycle process does not select such population for rerate.

2. Based on the data in the Rerate Population table, opens rerate sessions in the Event Server until the maximum number of sessions is reached. A rerate session represents an interaction between the Rerate Prepare process and the Event Server for a finite period of time. As a result of this interaction, a customer's or subscriber's events are rerated.

In contrast to the Mid-Cycle mode, no session initialization is required. This is because in the End-of-Cycle mode, the Rerate Prepare for End of Cycle process works with a dedicated Event Server that is not handling ongoing events, so it is always available for rerating. Because there is no session initialization, the Rerate Prepare for End of Cycle process can prepare more sessions in advance than the number of sessions actually sent to the Event Server, thereby optimizing performance.

3. Maintains the maximum number of open sessions by replacing the finished rerate session with a new one that was prepared in advance.
4. During rerating, updates its progress in the summarized Rerate Population table.
5. After processing each session, writes its data into the latency log.
6. When all customers of the specific Oracle partition have been rerated, takes the next Oracle partition to handle.

Process Shutdown

The Rerate Prepare for End of Cycle process shuts down if all Oracle partitions have been handled.

Environment Variables

Rerate Prepare for End of Cycle uses the following environment variables that are initialized by the op_adj_env_sh script.

Variable Name	Description	Valid Values/ Default Value
APR_EOC_POP_PREP_BULK	The number of records moved to the summarized Rerate Population table from the intermediate Subscriber Rerate table	1000
APR_EOC_POP_RETRIEVAL_BULK	The number of records loaded to the memory of the process at once	1000

Variable Name	Description	Valid Values/ Default Value
APR_EXTRACT_EVENTS_BULK_NUM	The number of records fetched from the APE1_RATED_EVENT table at once	100
APR_EXTRACT_MAX_FILE_SIZE	The maximum size of files that contain the data extracted from the APE1_RATED_EVENT table	1000000
APR_NON_REGUIDABLE_EVENT_IDS	A comma-separated list of IDs of events that cannot be reguided (for example, technical events). If this parameter is populated with event IDs, only the specified events are considered non-reguidable. If this parameter is empty, events whose resource ID is '0' or empty are considered non-reguidable.	0
APR_RR_LATENCY_ENABLED	Indicates whether messages must be printed to the latency log	false
APR_RRP_REGUIDE_ROUTING_CRITE_RIA	The routing criteria for the Reguided Events file	rc1
APR_RRPEOC_DATA_EXTR_NUM_OF_THREADS	The number of Data Extractor threads	20
APR_RRPEOC_MAX_CUSTOMERS_THRESHOLD	The number of customers for which events must be extracted.	1
APR_RRPEOC_MAX_EXTRACT_SESSION_COUNT	The maximum number of sessions that the process can prepare by extracting events to files.	120
APR_RRPEOC_MAX_FILE_NUMBER_PER_SESSION	The maximum number of files with extracted data that each rerate session can use at the same time	20
APR_RRPEOC_QUEUE_MAX_SIZE	The maximum size of the queue	200000
STATEMENT_HINT	The hint for the SQL statement used for Event Extracts	/*+ USE_CONCAT */

For more information on the script, see the “Amdocs Batch Job Manager Application Script” section in the “Introduction” chapter.

Parameters

The following parameters must be set for the Rerate Prepare for End of Cycle process to run properly.



Note: In End-of-Cycle Rerate, cycle-related information is retrieved from the configuration of the Rerate Prepare for End of Cycle process.

Parameter	Description	Default Value
APPLICATION_ID	The process instance	RRP1900
CCF_FILE_NAME	An XML file that contains the relevant configuration	APR1_RRP_EOC.ccf

Configuration Parameters

The configuration parameters for the Rerate Prepare for End of Cycle process are defined in the APR1_CONF_SECTION_PARAM table and can be modified using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*).

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
RRP	config	EOC_RETRIEVER_HINT	/* */	The hint for selecting the population from the APE1_RERATE_POPULATION table.
RRP	config	EXPIRED_SESSION_DELETE_HINT_1	/* */	The hint for the SQL statement used for deleting expired sessions.
RRP	config	EXPIRED_SESSION_DELETE_HINT_2	/* */	The hint for the SQL statement used for deleting expired sessions.
RRP	config	EXPIRED_SESSION_INSERT_HINT	/* */	The hint for the SQL statement used for inserting expired sessions.
RRP	config	NEW_SESSION_DELETE_HINT_1	/* */	The hint for the SQL statement used for deleting new sessions.
RRP	config	NEW_SESSION_DELETE_HINT_2	/* */	The hint for the SQL statement used for deleting new sessions.
RRP	config	NEW_SESSION_INSERT_HINT	/* */	The hint for the SQL statement used for inserting new sessions.
RRP	config	NON_REGUIDABLE_EVENT_IDS	\${APR_NON_REGUIDABLE_EVENT_IDS}	A comma-separated list of IDs of events that cannot be reguided (for example, technical events). If this parameter is populated with event IDs, only the specified events are considered non-reguidable. If this parameter is empty, events whose resource ID is ‘0’ or empty are considered non-reguidable.
RRP	config	SEGMENT_DELETE_HINT	/* */	The hint for the SQL statement used for deleting segments.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
RRP	config	SEGMENT_INSERT_HINT	/* */	The hint for the SQL statement used for inserting segments.
RRP	config	UNCOMPLETED_SESSION_DELETE_HINT_1	/* */	The hint for the SQL statement used for deleting incomplete sessions.
RRP	config	UNCOMPLETED_SESSION_DELETE_HINT_2	/* */	The hint for the SQL statement used for deleting incomplete sessions.
RRP	config	UNCOMPLETED_SESSION_INSERT_HINT	/* */	The hint for the SQL statement used for inserting incomplete sessions.
RRP	config	VIRTUAL_OR_SHARED_MEMORY_USED	VIRTUAL	Specifies whether the process uses the shared or virtual memory for routing information.
RRP	EXTRACT_DATA_PARAMS	ADDITIONAL_WHERE_CLAUSE	AND RE.EVENT_STATE = 'N'	Specifies the additional WHERE clause for the SQL statement used for Event Extracts.
RRP	EXTRACT_DATA_PARAMS	EXTRACT_EVENTS_BULK_NUM	\${APR_EXTRACT_EVENTS_BULK_NUM}	The number of records fetched from the APE1_RATED_EVENT table at once.
RRP	EXTRACT_DATA_PARAMS	EXTRACT_MAX_FILE_SIZE	\${APR_EXTRACT_MAX_FILE_SIZE}	The maximum size of files that contain the data extracted from the APE1_RATED_EVENT table.
RRP	EXTRACT_DATA_PARAMS	EXTRACT_OUTPUT_DIR_NAME	\${ABP_AP_R_ROOT}/interfaces/output/re rate	The directory where the files that contain rerated events are created.
RRP	EXTRACT_DATA_PARAMS	REGUIDE_OUTPUT_DIR_NAME	\${ABP_AP_R_ROOT}/interfaces/output/re guide	The directory where the files that contain reguided events are created.
RRP	EXTRACT_DATA_PARAMS	REGUIDE_OUTPUT_FILE_NAME	ReguidedEvents	The root of the name of the file that contains reguided events.
RRP	EXTRACT_DATA_PARAMS	REGUIDE_OUTPUT_FILE_PREFIX	RRP_-	The prefix of the file name of the file that contains reguided events.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
RRP	Logger.ADJ.RERATE.LATENCY	Enable	Y	Enables the latency log output. This parameter must always be set to 'Y'. The RR_LATENCY_REPORT_ENABLED parameter actually controls the printing of messages to the log.
RRP	Logger.ADJ.RERATE.LATENCY	Outputs	RRPGNRL	The output object assigned to the latency log.
RRP	Logger.ADJ.RERATE.LATENCY	Severity	0	The message severity threshold for the latency log.
RRP	LOGGER.output.RRPGNR	BufferSize	\${BUFFER_SIZE}	The size of the temporary buffer of the latency log.
RRP	LOGGER.output.RRPGNR	Filename	\${ABP_APPL_LOG}/ \${GN1_TASK_NAME}_LATENCY_REPORT_\${APPLICATION_ID}_%D_%T_%n.log	The name of the latency log file.
RRP	LOGGER.output.RRPGNR	FlushPeriod	\${FLUSH_PERIOD}	The interval at which the data from the temporary buffer is flushed to a file.
RRP	LOGGER.output.RRPGNR	ImmediateFlush	\${IMMEDIATE_FLUSH}	Indicates whether the data is flushed to a file immediately and not periodically.
RRP	LOGGER.output.RRPGNR	Layout	RR_Latency_Layout	The layout type that the latency log uses to format messages.
RRP	LOGGER.output.RRPGNR	MaxFileSize	\${MAX_FILE_SIZE}	The maximum size of a latency log file.
RRP	LOGGER.output.RRPGNR	Mode	\${MODE}	The working mode of the latency log: synchronous or asynchronous.
RRP	RR_PARAMS	RR_LATENCY_REPORT_ENABLED	\${APR_RR_LATENCY_ENABLED}	Indicates whether the latency log is initially turned on.
RRP	SESSION_PROGRESS_TRACKER_PARAMS	MAX_SESSION_OPEN_TIME	300	The maximum time for the Event Server to open a session.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
RRP	SESSION_PROGRESS_TRACKER_PARAMS	MAX_SESSION_UPDATE_TIME	300	The time interval for the Event Server to send an update of the progress of the rerate session.
RRP_EOC	EOC_SECTION	CYCLE_CODE	N/A	The code of the cycle that is being closed.
RRP_EOC	EOC_SECTION	CYCLE_INSTANCE	N/A	The instance of the cycle that is being closed.
RRP_EOC	EOC_SECTION	CYCLE_YEAR	N/A	The year of the cycle that is being closed.
RRP_EOC	EXTRACT_DATA_PARAMS	EXTRACT_MAX_CUSTOMERS_THRESHOLD	\${APR_RRPOG_MAX_CUSTOMERS_THRESHOLD}	The number of customers for which events must be extracted.
RRP_EOC	EXTRACT_DATA_PARAMS	EXTRACT_MAX_FILE_NUMBER_PER_SESSION	\${APR_RRPEOC_MAX_FILE_NUMBER_PER_SESSION}	The maximum number of files with extracted data that each rerate session can use at the same time.
RRP_EOC	EXTRACT_DATA_PARAMS	STATEMENT_HINT	\${STATEMENT_HINT}	The hint for the SQL statement used for Event Extracts.
RRP_EOC	PREPARATOR_PARAMS	NUM_PARALLEL_TASKS	6	The number of tasks that can be loaded from the APR1_EOC_RERATE_TASKS table in parallel.
RRP_EOC	PREPARATOR_PARAMS	POPULATION_PREPARATION_BULK	\${APR_RRP_EOC_POP_PREP_BULK}	The number of records moved to the summarized Rerate Population table from the intermediate Subscriber Rerate table.
RRP_EOC	RETRIEVER_PARAMS	POPULATION_RETRIEVAL_BULK	\${APR_EOC_POP_RETRIEVAL_BULK}	The number of records loaded to the memory of the process at once.
RRP_EOC	RETRIEVER_PARAMS	RETRIEVALS_NUMBER	3	The number of attempts to rerate the same customer.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
RRP_EOC	RETRIEVER_PARAMS	WORK_MODE	EOC	The working mode of the process. Valid values: <ul style="list-style-type: none">■ ON_GOING■ ON_DEMAND■ EOC
RRP_EOC	RR_PARAMS	GRACEFULL_SHUT_DOWN_NUM_RETRY	3	The maximum number of attempts to send a graceful shutdown message to the End-of-Cycle Event Server.
RRP_EOC	RR_PARAMS	MAX_SESSION_COUNT	75	The maximum number of sessions that the process can establish against the Event Server.
RRP_EOC	RR_PARAMS	WAIT_FOR_GRACEFULL_SHUT_DOWN_RSP	100	The number of seconds for which the process must wait for a graceful shutdown response from the End-of-Cycle Event Server.
RRP_EOC	SESSION_MANAGER_PARAMS	MAX_EXTRACT_SESSION_COUNT	<code>#{APR_RRPEOC_MAX_EXTRACT_SESSION_COUNT}</code>	The maximum number of sessions that the process can prepare by extracting events to files.

In addition, the process uses Light Tracer configuration parameters, which are defined in the ADJ1_CONF_SECTION_PARAM table. For more information, see [Light Tracer Parameters](#) in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

For each process instance, an entry with the name of the process instance must be created in the APR1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
RRPEOC1900	RRPEOC

Any parameter to be overridden at the process instance level must be added to the APR1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
RRPEOC 1900	EXTRACT_DATA_PARAMS	REGUIDE_OUTPUT_FILE_PREFIX	RG_

For more information on parameters and the properties mechanism, see the “Configuration Parameter Mechanism” appendix.

Configuration Files

The Rerate Prepare for End of Cycle process requires a configuration file (.ccf) that is defined through an internal configuration tool.

It also requires parameters from Audit & Control, which appear in the configuration file.

The following parameters can be configured in the APR1_RRP_EOC.ccf file.

Property Name	Related Internal Module	Description	Default Value
Data Group	Reguider strategy	The data group of the output file. This parameter is used to distinguish between files that are to be processed by different processes of the same type. It is a column in the AC1_CONTROL table.	4444
File Alias	Reguider strategy	The file alias of the output file.	TCOCIUSAGE
File Union	Reguider strategy	Indicates whether the Event Server must check for duplicates of the event.	TCOCIUSAGE
Routing Criteria	Reguider strategy	The number of instances of each acceptor type. Each acceptor handles one file.	rc1

The environment variables that appear in the .ccf file and can be customized are defined in the op_adj_env_sh script (for more information, see the “Environment Variables” section in this chapter).

Database Connections

During initialization, the process connects to many reference tables. For a complete description of each table, see *Turbo Charging Data Model*.

After initialization, the Rerate Prepare for End of Cycle process connects primarily to the following tables.

Database Type	Table Name	Connection Type (Update, Insert, and so on)	Comments
Application	APR1_EOC_RERATE_TASKS	Select, Update, Delete	N/A
Customer	APE1_RERATE_POPULATION	Select, Update, Delete	N/A
Customer	APE1_SUBSCRIBER_RERATE	Select, Update, Delete	N/A
Usage	AC1_CONTROL	Insert	N/A
Usage	APE1_RATED_EVENT	Update	It is assumed that the Usage database contains a synonym to the AC1_CONTROL table.
Usage	APE1_RR_BLACK_LIST	Select	N/A

Admin Commands

The Rerate Prepare for End of Cycle process supports the following admin commands. For information on how to execute the commands, see the “[Handling Admin Commands](#)” section in the “[High Availability](#)” chapter.

Command	Description	Parameters
SET_LIGHT_TRACER_COMMAND	Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters	<ul style="list-style-type: none"> ▪ Activation parameters: <ul style="list-style-type: none"> • <i>turnTracer “on/off”</i> – Activates or stops the Light Tracer • <i>--turnAssert “on/off”</i> – Activates or stops Assertions • <i>--turnLogOnTrace “on/off”</i> – Activates or stops writing log messages in Light Tracer files • <i>--turnFlowTracer/turnAll “on/off”</i> – Activates or stops tracing a single flow or event that is transferred between machines • <i>--turnAll “on/off”</i> – Activates or stops all of the above. ▪ The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter). ▪ <i>--moduleID “ALL”</i> ▪ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • <i>--threadInfo “Y/N”</i> – Specifies whether trace or assertion messages include the thread information. • <i>--timestampInd “Y/N”</i> – Specifies whether trace or assertion messages include the time stamp. • <i>--contextInd “Y/N”</i> – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • <i>--noOfMessages “<number>”</i> – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down. • <i>--period “5/10/20/60/120/480/1440”</i> – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated. <p> Note: If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached.</p>

Command	Description	Parameters
		<ul style="list-style-type: none"> • <code>--assertAbortInd "Y/N"</code> – Specifies whether the application is to be aborted if the assertion condition fails. ▪ <code>--commandType "changeBufferSize"</code> – Resizes the Light Tracer buffer at run time. <ul style="list-style-type: none"> • <code>--bufferSize "<number>"</code> – The new size for the Light Tracer buffer.
SET_TRACE_SQL_COMMAND	<p>Activates or stops SQL tracing at runtime. The command name with its parameters must be enclosed in quotation marks.</p> <p><i>Example:</i> <code>ADJ1_Send_Admin_Command_Sh illin1027 <process instance></code> <code>"SET_TRACE_SQL_COMMAND --traceSqlOn \"yes\" --duration \"1\""</code></p>	<code>--traceSqlOn "<yes or no>" --duration "<minutes>"</code> If the IS_DURATION_REQUIRED environment variable is set to 'Y', the --duration parameter is mandatory. If the duration specified in the command exceeds the allowed maximum value, SQL tracing is activated for the maximum allowed period of time only (60 minutes). <div style="margin-left: 20px;">  Note: The SHUT_DOWN_CHECK_FREQ parameter of the Light Tracer in the ADJ1_CONF_SECTION_PARAM table determines the frequency (defined in the number of messages) at which the SQL Trace Manager checks whether the period specified in the --duration parameter has passed. Therefore, SQL tracing is stopped after the specified duration or after this number of messages has been reached, whichever comes later. </div>
SWITCH_LIGHT_TRACER_ON ALL	Activates the Light Tracer without the ability to specify the mode or send parameters.	N/A
SWITCH_LIGHT_TRACER_OFF ALL	Stops the Light Tracer	N/A
RR_LATENCY_REPORT_ENABLED	Activates the latency log	N/A
CPF_GracefulShutdownCommand	Shuts down the Rerate Prepare process gracefully.	N/A

Examples:

- `ADJ1_Send_Admin_Command_Sh hpbl820 RRPEOC 1900 SET_LIGHT_TRACER_COMMAND –turnTracer "on" –moduleId "ALL" –threadInfo "y"`
- `ADJ1_Send_Admin_Command_Sh hpbl820 RRPEOC 1900 SWITCH_LIGHT_TRACER_OFF ALL`

Recovery Instructions

Rerun the process.

16 ADJ1DISPENTFR – Rerate Prepare for End of Cycle Finish Notification

This chapter describes the Rerate Prepare for End of Cycle Finish Notification job.

Description

The Rerate Prepare for End of Cycle Finish Notification job informs the Dispatcher for End of Cycle job that the latter can exit if there are no more relevant entries in the APR1_DISPATCHER_CTRL table.

Process Type

Batch job

Run Frequency

Every cycle

Participation in Generic Maps

This section contains information on the generic maps in which the process participates.

Map Name

The process participates in the following operational maps:

- End-of-Cycle (EOC)
- Undo

For more information about these maps, see *Amdocs Billing Operator Guide*.

Severity Level

Severity Level 1 – The map can continue to the next job without fixing this job.

Activation

Command line:	RunJobs ADJ1DISPENTFR BYREQ
Amdocs Monitoring & Control:	No
Script Name:	ADJ1_Dispatcher_Ntfr_Job_Sh
Executable Name:	NA

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Preceding Processes

The following job must run successfully before this process is activated:

- *ADJIRRPEOC* – Rerate Prepare for End of Cycle

Affected Applications

The process affects the Dispatcher for End of Cycle (ADJ1DISPEOC) job. Dispatcher for End of Cycle shuts down if it has finished its tasks for the cycle, and it finds the notification in the APR1_OP_MAP_EVENTS table indicating that no more tasks are expected.

Flow

The Rerate Prepare for End of Cycle Finish Notification job inserts a special record into the APR1_OP_MAP_EVENTS table. This record informs the Dispatcher for End of Cycle job that it can exit when there are no more relevant entries in the APR1_DISPATCHER_CTRL table.

Parameters

The following job parameters must be set for the job to run properly. When the job runs as part of a map, the Turbo Charging Cycle Bill Run script populates these parameters automatically. For more information on this script, see the “ADJ1_APB_CycleBillRun_Sh – Turbo Charging Cycle Bill Run” chapter.

Parameter	Description	Default Value
CYCLE_CODE	The cycle code.	N/A
CYCLE_INST	The cycle instance.	N/A
CYCLE_YEAR	The cycle year.	N/A
MAP_MODE	The context (map) in which the process is currently running. Valid values are: <ul style="list-style-type: none">■ NONE (if the job is run manually and not as part of a map)■ EOC■ UNDO	N/A

Database Connections

The job connects to the following table during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	APR1_OP_MAP_EVENTS	Insert

Recovery Instructions

Rerun the job.

17 ADJ1ERRREP – Rerate Error Report

This chapter describes the Rerate Error Report process.

Description

The Rerate Error Report process extracts the customer population whose events failed in rerate.

Process Type

Batch job

Run Frequency

By request

Participation in Generic Maps

This section contains information on the generic maps in which the process participates.

Map Name

- End-of-Cycle (EOC)
- Undo

For more information about these maps, see *Amdocs Billing Operator Guide*.

Severity Level

Severity Level 2 – The map must hold until the job is fixed.

Activation

Command Line:	RunJobs ADJ1ERRREP BYREQ
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_ReratePopulationReport_Job_Sh
Executable Name:	gcpf1fwcApp

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> RRP_ERR REP \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

ADJ1_Send_Admin_Command_Sh hpbl820 RRP_ERR REP 270
CPF_GracefulShutdownCommand

Preceding Processes

The following job must run successfully before this process is activated:

- *ADJ1RRPEOC* – Rerate Prepare for End of Cycle

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*

ADJ1RRPOPREP_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1RRPOPREP_RRP_ERR_REP270_20081109_130927_1.log

- *Console log files:*

ADJ1_ReratePopulationReport_Job_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1_ReratePopulationReport_Job_inner_Sh_RRP_ERR_REP270_20081109_130927.log

- *Operational log files:*

ADJ1ERRREP_<Job Record Name>_\${ABP_MARKET}_<Date>_<Time>.log

Example:

ADJ1ERRREP_BYREQ_M3G_20081109_130927.log

- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter)

For information about the configuration parameters that define the location of log and trace files, see the “[Configuration Parameters](#)” section in the “[ADJ1EVENTSRV – Event Server](#)” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information.
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).
- *Operational log files* – The output of operational scripts.
- *Light Tracer files* – See the “[Light Tracer Files](#)” section in the “[Introduction](#)” chapter.

Output Files

This section describes the output files.

Name

ReratePopulationReportData_<DateTime>_<PID>_<ThreadID>_<FileNo>.dat

Location and Format

- *Location* – \$ABP_APP_ROOT/interfaces/output/rrp_rep
- *Format* – Semicolon-delimited file

Contents

The output file contains the customers that are marked for rerate.

Flow

The Rerate Error Report extracts the customer population that failed in rerate according to the cycle code and cycle instance from the APE1_SUBSCRIBER_RERATE and APE1_RERATE_POPULATION tables.

Environment Variables

The Rerate Error Report uses the following environment variables.

Variable Name	Description	Valid Values/ Default Value
FILE_UNION_FOR_UNDO	The value of the file union for the Undo map: <ul style="list-style-type: none">■ <i>FCURER</i> – FULL CYCLE UNDO■ <i>RERUERR</i> – Other UNDO map modes (FULL_RUN and RUN)	Populated by the ADJ1_APP_CycleBil lRun_Sh script
ROUTING_CRITERIA	The routing criteria for registering files to Audit & Control	RPR1_TO_BL1

Parameters

The following parameters must be set for the process to run properly. When the job runs as part of a map, the Turbo Charging Cycle Bill Run script populates these parameters automatically. For more information on this script, see the “ADJ1_AP4_CycleBillRun_Sh – Turbo Charging Cycle Bill Run” chapter.

Parameter	Description	Default Value
APPLICATION_ID	The ID of the job that enables it to get its parameters.	N/A
CCF_FILE_NAME	An XML file that contains the relevant configuration.	N/A
CYCLE_CODE	The cycle code for which data is to be extracted.	N/A
CYCLE_INST	The cycle instance for which data is to be extracted.	N/A
CYCLE_YEAR	The cycle year for which data is to be extracted.	N/A
EXTR_INPUT_MAP_KEY	The input map key. This key is used as a filter. Only the record with this value appears in the DATA_GROUP field of the AC1_CONTROL table.	N/A
FILE_UNION_FOR_UNDO	The value of the file union for the Undo map	\${FILE_UNION_FOR_UNDO}
MAP_MODE	<p>The context in which the process is activated:</p> <ul style="list-style-type: none"> ■ EOC ■ UNDO <p>Other possible values are:</p> <ul style="list-style-type: none"> ■ RERUN ■ QA ■ NONE (if the job is run manually and not as part of a map) 	N/A
RUN_REQUEST_ID	The request that is used to expand the input map key and insert it as part of the data group into the AC1_CONTROL table.	N/A

Configuration Parameters

In Turbo Charging, the configuration parameters (properties) of the Rerate Error Report process are defined in the database.

The following table shows the performance tuning (configuration) parameters of the Rerate Error Report process. These parameters are defined in the APR1_CONF_SECTION_PARAM table using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*).



Note: Other parameters must not be changed.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
RRP_REP	FILE_OUTPUT	FILE_DIRECTORY	\${ABP_AP_R_ROOT}/interfaces/output/rp_rep	The default directory where the output file is created.
RRP_REP	FILE_OUTPUT	FILE_NAME	ReratePopulationReportData.dat	The naming convention for the output file.
RRP_REP	POPULATION_REPORT	CYCLE_CODE	\${CYCLE_CODE}	The cycle code for which the customers that failed in rerate are extracted.
RRP_REP	POPULATION_REPORT	CYCLE_INSTANCE	\${CYCLE_INST}	The cycle instance for which the customers that failed in rerate are extracted
RRP_REP	POPULATION_REPORT	CYCLE_YEAR	\${CYCLE_YEAR}	The cycle year for which the customers that are marked for rerate are extracted.
RRP_REP	POPULATION_REPORT	DISTRIBUTE_WORK	\${POPREP_DISTRIBUTE_WORK}	The flag that indicates whether the Rerate Population Report or the Error Population Report distributes the work to the Rerate job, as follows: <ul style="list-style-type: none">■ Yes – Error Population Report■ No – Rerate Population Report

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
RRP_REP	POPULATION_REPORT	FILE_TYPE	\${FILE_TYPE}	The type of files to insert into the AC1_CONTROL table: <ul style="list-style-type: none"> ▪ <i>Undo map</i> – AFTER_RERATE_UNDO ▪ <i>EOC map</i> – AFTER_RERATE
RRP_REP	POPULATION_REPORT	INPUT_MAP_KEY	\${EXTR_INPUT_MAP_KEY}	The input map key. This key is used as a prefix for the DATA_GROUP field of the AC1_CONTROL table.
RRP_REP	POPULATION_REPORT	RUN_REQUEST_ID	\${RUN_REQUEST_ID}	The request that is used to expand the input map key and insert it as part of the data group into the AC1_CONTROL table.

In addition, the process uses Light Tracer configuration parameters, which are defined in the ADJ1_CONF_SECTION_PARAM table. For more information, see [Light Tracer Parameters](#) in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

For each process instance, an entry with the name of the process instance must be created in the APR1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
RRP_ERR_REP270	RRP_OP REP

Any parameter to be overridden at the process instance level must be added to the APR1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
RRP_ERR_REP270	FILE_OUTPUT	FILE_NAME	Example.dat

For more information on parameters and the properties mechanism, see the “Configuration Parameter Mechanism” appendix.

Configuration Files

The Rerate Error Report process requires a configuration file (.ccf) that is defined via an internal configuration tool.

Database Connections

The process connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	AC1_CONTROL	Insert
Usage	APE1_RATED_EVENT	Select
Usage	APE1_RERATE_POPULATION	Select
Usage	APE1_SUBSCRIBER_RERATE	Select

Admin Commands

The Rerate Error Report supports the following admin commands. For information on how to execute the commands, see the “Handling Admin Commands” section in the “High Availability” chapter.

Command	Description	Parameters
SET_LIGHT_TRACER_COMMAND	<p>Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters</p>	<ul style="list-style-type: none"> ▪ Activation parameters: <ul style="list-style-type: none"> • <i>turnTracer “on/off”</i> – Activates or stops the Light Tracer • <i>--turnAssert “on/off”</i> – Activates or stops Assertions • <i>--turnLogOnTrace “on/off”</i> – Activates or stops writing log messages in Light Tracer files • <i>--turnFlowTracer/turnAll “on/off”</i> – Activates or stops tracing a single flow or event that is transferred between machines • <i>--turnAll “on/off”</i> – Activates or stops all of the above. The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter). ▪ <i>--moduleId “ALL”</i> ▪ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • <i>--threadInfo “Y/N”</i> – Specifies whether trace or assertion messages include the thread information. • <i>--timestampInd “Y/N”</i> – Specifies whether trace or assertion messages include the time stamp. • <i>--contextInd “Y/N”</i> – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • <i>--noOfMessages “<number>”</i> – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down. • <i>--period “5/10/20/60/120/480/1440”</i> – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated.

Command	Description	Parameters
		 Note: If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached. <ul style="list-style-type: none"> • <code>--assertAbortInd "Y/N"</code> – Specifies whether the application is to be aborted if the assertion condition fails. ▪ <code>--commandType "changeBufferSize"</code> – Resizes the Light Tracer buffer at run time. • <code>--bufferSize "<number>"</code> – The new size for the Light Tracer buffer.
SET_TRACE_SQL_COMMAND	<p>Activates or stops SQL tracing at runtime. The command name with its parameters must be enclosed in quotation marks.</p> <p><i>Example:</i> <code>ADJ1_Send_Admin_Command_Shillin1027 <process instance> "SET_TRACE_SQL_COMMAND --traceSqlOn \"yes\" --duration \"1\""</code></p>	<code>--traceSqlOn "<yes or no>" --duration "<minutes>"</code> If the IS_DURATION_REQUIRED environment variable is set to ‘Y’, the --duration parameter is mandatory. If the duration specified in the command exceeds the allowed maximum value, SQL tracing is activated for the maximum allowed period of time only (60 minutes).  Note: The SHUT_DOWN_CHECK_FREQ parameter of the Light Tracer in the ADJ1_CONF_SECTION_PARAM table determines the frequency (defined in the number of messages) at which the SQL Trace Manager checks whether the period specified in the --duration parameter has passed. Therefore, SQL tracing is stopped after the specified duration or after this number of messages has been reached, whichever comes later.

Command	Description	Parameters
SWITCH_LIGHT_TRACER_ON ALL	Activates the Light Tracer without the ability to specify the mode or send parameters.	N/A
SWITCH_LIGHT_TRACER_OFF ALL	Stops the Light Tracer	N/A
CPF_GracefulShutdownCommand	Shuts down the Rerate Error Report gracefully.	N/A

Examples:

- ADJ1_Send_Admin_Command_Sh hpbl820 RRP_ERR REP 270 SET_LIGHT_TRACER_COMMAND –turnTracer “on” –moduleId “ALL” –threadInfo “y”
- ADJ1_Send_Admin_Command_Sh hpbl820 RRP_ERR REP 270 SWITCH_LIGHT_TRACER_OFF ALL

Recovery Instructions

Rerun the job.

18 ADJ1CLNBLACK – Blacklist Table Clean-up

This chapter describes the Blacklist Table Clean-up job.

Description

The Blacklist Table Clean-up job removes data from the APE1_RR_BLACK_LIST table on demand. The job uses the ADJ1_CYCLE_STATE table to check whether a cycle is closed, and then cleans the APE1_RR_BLACK_LIST table according to the cycle code, cycle instance, and cycle year.

Process Type

Batch job

Run Frequency

By request

Activation

Command Line:	RunJobs ADJ1CLNBLACK BYREQ
Amdocs Monitoring & Control:	No
Script Name:	ADJ1_CleanBlackList_Job_Sh
Executable Name:	N/A

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has successfully completed.

Log files are created and handled in the default way.

Name

ADJ1CLNBLACK _SCRIPT_<Date>_<Time>.log

Example:

ADJ1CLNBLACK _SCRIPT_21110629_165002.log

Location

The files are located in \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/.

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log file contains the process error information.

Flow

The Blacklist Table Clean-up job executes the following steps:

1. Sets all database connections
2. Checks whether the cycle is closed in ADJ1_CYCLE_STATE table.
3. If yes, truncates the relevant partition of the APE1_RR_BLACK_LIST table

Database Connections

The job connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Applicative	ADJ1_CYCLE_STATE	Select
Usage	APE1_RR_BLACK_LIST	Truncate

Important Remarks

- The data belonging to a cycle code is truncated only if the cycle is closed.
- The job does not support databases that are not partitioned.

19 ADJ1RER – Rejected Event Recycler

This chapter describes the Rejected Event Recycler process.

Description

The Rejected Event Recycler (RER) is a Java application used to recycle rejected events.

While processing a particular event, the Event Server may encounter an error. Once the Event Server identifies the event as rejected, the event is stored in the APE1_REJECTED_EVENT table. The events in this table are then manipulated using an error management tool such as Amdocs Error Manager. The Rejected Event Recycler takes these corrected events from the database, formats them using the Text Simple Protocol or a customized protocol, and inserts them into a file. This file is registered with Audit & Control so that it may be further processed by File2E or/and any other applications based on the configuration. The events in this file are subsequently sent to the Event Server for reprocessing.

Process Type

- Batch job
- Daemon

Run Frequency

- *Batch job* –Runs by request and processes the data in each relevant database partition only once. After the data in the partitions has been processed, the process exits.
- *Daemon* – Runs continuously.

Activation

This section describes how to activate Rejected Event Recycler in various modes.



Caution: When the home database is up after a failure, the operator must not activate the Rejected Event Recycler process until the Copy Cycle Usage process has copied the Rejected Event table from the alternate database to the home database.

Batch Job

Command line:	ADJ1_RER_Job_Shell_Sh -n <APPLICATION_ID> -e <EWD_EXE>  Note: The -e parameter is optional. It is used in environments with the Availability Manager.
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_RER_Job_Shell_Sh
Executable Name:	N/A

Example:

ADJ1_RER_Job_Shell_Sh -n RER157

Daemon

Command line:	ADJ1_RER_Daemon_Shell_Sh -n <APPLICATION_ID> -e <EWD_EXE>
	<i>Note: The -e parameter is optional. It is used in environments with the Availability Manager.</i>
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_RER_Daemon_Shell_Sh
Executable Name:	N/A

Example:

ADJ1_RER_Daemon_Shell_Sh -n RER146

For more information about the activation scripts, see the “[Scripts](#)” section in the “[Introduction](#)” chapter.

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> RER \${PROCESS_INSTANCE_NUMBER} 100
Amdocs Monitoring & Control:	Yes

Examples:

- Daemon mode:

ADJ1_Send_Admin_Command_Sh hpx418 RER 146 100

- Batch mode:

ADJ1_Send_Admin_Command_Sh hpx418 RER 157 100

Preceding Processes

The following jobs must run successfully before this process is activated, and the following daemons must be active for this process to run:

- *Event Server* – Inserts the rejected events into the APE1_REJECTED_EVENT table
- *Implementation Compiler* – Populates the APE1_RATED_EVENT_MAP table
- *Error management system (for example, Amdocs Error Manager)* – Makes corrections in the rejected events to prepare them for recycling

Affected Applications

The process affects the following applications:

- *File2E* – Picks up and parses the files registered by the Rejected Event Recycler in Audit & Control.
- *Event Server* – Receives events formatted by the Rejected Event Recycler for reprocessing.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

ADJ1RER_<Process Instance>_<Time Stamp>.log

Location

The files are located in \${ABP_AJTRER_ROOT}/logs.

Contents

The files are standard log4j log files. They contain the error and warning messages that occurred during the Rejected Event Recycler run as well as debugging information if the process is run in debug mode.

Output Files

This section describes the output files of the process.

Rejected Event Files

This section describes the files that contain rejected events.

Name

Output files follow these naming conventions:

- If the MULTIFORMAT parameter is set to ‘Y’: *<File Prefix>_<File Type>_<Serial Number>_<Time in Milliseconds>_<Application ID>_<File Format>.tsp*
- If the MULTIFORMAT parameter is set to ‘N’: *<File Prefix>_<File Type>_<Serial Number>_<Time in Milliseconds>_<Application ID>.tsp*

where:

- *File Prefix* – The value of the rer.ac.OutputFileName.prefix configuration parameter (see the “Configuration Parameters” section in this chapter).
- *File Type* – One of the following:
 - RejectedEvents
 - PurgedEvents
 - Dummy_PurgedEvents
- *File Format* – The format of the records in the output file (such as USAGE_ERR or GD_ERR) if multi-formatting is enabled (see the “Configuration Parameters” section in this chapter).

The names of Pre-Balance files consist of the EVENT_STATE_REASON_CODE prefix and the name of the corresponding Rejected Event file, where EVENT_STATE_REASON_CODE is the flow in which the events were rejected, for example, ‘RR’ (Rerate) or ‘UN’ (Undo).

Location and Format

- *Location* – <rer.logpath>/AuditAndControl/AcFiles.
- *Format* – Each file contains rejected events formatted using the Text Simple Protocol.

Contents

These files contain rejected or purged events formatted using the Text Simple Protocol. Dummy files are empty.

Statistics Files

This section describes the statistics files.

Name

<instance_name>_stats_<sequence_number>.csv

Location and Format

- *Location* – <rer.logpath>/metrics/
- *Format* – Each line contains information in the following format:
<Time Stamp> <amdocs.adjust.rer.RejectedEventsRead value>,<Delta Value>
<amdocs.adjust.rer.RejectedEventsProcessed value>,<Delta Value>

Contents

This file contains the following information:

- Date and time
- Total number of records read from the database
- Delta of records read from the database in delta time
- Total number of records processed
- Delta of records processed in delta time

Flow

The Rejected Event Recycler process executes the following steps.

Initialization

During initialization, the process performs the following tasks:

1. Retrieves the value of command line parameters
2. Initializes the TopicManager for passing messages
3. Initializes the ShutdownManager, which performs orderly shutdown of the application and cleans up the resources that are used by initialized APIs if the process encounters a problem during initialization
4. Initializes Routing APIs
5. Fetches the configuration parameters using the RERProperties class
6. Initializes the Internal Watchdog (IWD)
7. Initializes the EventMappingsManager, which retrieves event attribute mappings from the APE1_RATED_EVENT_MAP table in the reference database
8. Initializes the database connection pool
9. Initializes the Audit & Control APIs
10. Initializes and starts the Reader and Worker thread pools
11. Initializes the AdminCommandManager to receive commands from the admin port
12. Calls the open() method on the Internal Watchdog to indicate that it has completed the initialization

Processing

During processing, the Rejected Event Recycler performs the following flow:

1. A set of Reader tasks is executed in a fixed-size Reader thread pool. Each Reader task performs the following:
 - a. Fetches recycled and purged events from a single database partition. The number of records retrieved by a Reader task in a single query is limited (this number is configurable).
 - b. Puts the records in customer wrapper objects containing the rejected events of a single customer. All customer objects belonging to a single partition are put in the PartitionData object.
 - c. Updates the RecordsRead counter using the metrics APIs.
 - d. Puts the PartitionData object that it created on the processing queue.

2. A set of Worker tasks is executed using a fixed-size Worker thread pool. Worker tasks do the following:
 - a. Pick up a RejectedEventsFile instance from the queue.
 - b. Format the events in it using the Text Simple Protocol.
 - c. Insert the formatted events into a file (if multi-formatting is enabled, create a separate file for each file alias). The records inserted into the file are sorted by start time and then by event ID for each customer. If the Pre-Balance reporting flag is on, pre-balance files are also created based on the EventStateReasonCode configuration parameter.
 - d. Update the processing status of all the records in the file to Complete (CO) and register the file with Audit & Control in the same database transaction.
 - e. Update the RecordsProcessed counter representing number of records processed by the application.
 - f. Unlike Reader tasks that exit after fetching the data from a single partition, exit only during the application shutdown.

Shutdown

Prior to shutting down, the Rejected Event Recycler performs the following tasks:

1. Sets the shutdown flag to communicate the status to the threads
2. Waits for all threads to exit
3. Closes the logs
4. Stops the Internal Watchdog

Environment Variables

The following environment variables affect the process.

Name	Description	Default Value	Method of Changing the Variable
ABP_ENCODING	The encoding of rejected event data. The Rejected Event Recycler uses this encoding to extract data from the APE1_REJECTED_EVENT table and to write the record into the Text Simple Protocol (TSP) file. If this variable is not defined, the default encoding is used.	UTF-8	Define the variable in the .w.ini.pre.env or .local.ini file.

Name	Description	Default Value	Method of Changing the Variable
ART_AMC_START_DB_CONNECT_STRING	A parameter used by the Availability Manager to connect to the appropriate database instance.	N/A	N/A
RER_RUN_BACKGROUND	Indicates whether the Rejected Event Recycler must run in the background.	N	N/A

In addition, the following environment variables control the logging in log4j.

Name	Description	Location	Default Value
log4jFND	The minimum severity of Foundation (FND) messages that are printed to the log. Valid values: <ul style="list-style-type: none">■ ERROR■ WARNING■ DEBUG■ INFO	\${HOME}/.w.ini.pre.env	ERROR
log4jRER	The minimum severity of Rejected Event Recycler (RER) messages that are printed to the log. Valid values: <ul style="list-style-type: none">■ ERROR■ WARNING■ DEBUG■ INFO	\${HOME}/.w.ini.pre.env	ERROR
log4jRTA	The minimum severity of RTA messages that are printed to the log. Valid values: <ul style="list-style-type: none">■ ERROR■ WARNING■ DEBUG■ INFO	\${HOME}/.w.ini.pre.env	ERROR

Parameters

The following parameter must be set for the process to run properly.

Parameter	Description	Possible Values
APPLICATION_ID	The name of the process instance. When a process instance is started from the Availability Manager, this name must match the one in the ADJ1_CONF_HIERARCHY table.	N/A

Configuration Parameters

The configuration parameters of the Rejected Event Recycler are defined in the database.

The following table shows the configuration parameters of the Rejected Event Recycler. The parameters are defined in the ADJ1_CONF_SECTION_PARAM table using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*).

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ADJ	Config	AVM_ENABLED	Y	Indicates whether the Rejected Event Recycler interacts with the Availability Manager.
ADJ	Config	RADAR_ENABLED	\${RADAR_ENABLED}	Indicates whether Pre-Balance reporting is enabled. The value of this parameter is an environment variable that must be defined in runtime environments. Its default value is 'N'.
AJTFND	fnd.connection.test	query	SELECT 1 FROM DUAL	The query for testing the integrity of the database connection object.
AJTFND	fnd.db	fetchSize	50	The size of the bulk that can be fetched.
RER	rer	logpath	\${ABP_AJTRER_ROOT}	The path to the subdirectories containing files with rejected events and statistics.
RER	rer.ac	DataGroup	DG,DG	Comma-separated data groups for each target in Audit & Control for rejected records.
RER	rer.ac	FileAlias	RER_RAW,RER_BIN	Comma-separated file aliases for each target in Audit & Control for rejected records.
RER	rer.ac	ProgramName	RER	The name of the current program.
RER	rer.ac	RoutingCriteria	RC,RC	Comma-separated routing criteria for each target in Audit & Control for rejected records.
RER	rer.ac.OutputFileName	prefix	RER	The prefix for all files created by the process.
RER	rer.ac.purge	DataGroup	PU_DG	The data group in Audit & Control for purged records.
RER	rer.ac.purge	FileAlias	RER_PURGE	The file alias in Audit & Control for purged records.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
RER	rer.ac.purge	RoutingCriteria	PU_RC	The routing criteria in Audit & Control for purged records.
RER	rer.boolean	false	false,f,n,no,0	Various possible values in the database to represent the Boolean false value.
RER	rer.boolean	true	true,t,y,yes,1	Various possible values in the database to represent the Boolean true value.
RER	rer.config	EventStateReasonCodes	RR,UN	The event state reason code for which a separate Pre-Balance file must be created if Pre-Balance reporting is enabled.
RER	rer.config	GroupByCustomer	Y	<ul style="list-style-type: none"> ■ If set to 'Y', indicates that the events of a customer must go to a single file. In this case, the number of files is less than or equal to the number of unique customer IDs. ■ If set to 'N', indicates that the maximum size of a file with rejected events must equal the size set by the rer.file.records.max parameter. Moreover, the records in the file are sorted by event start time. In this case, the minimum number of files equals the number of rejected events divided by the value of the rer.file.records.max parameter. <p>This setting increases the probability of a file containing the events of more than one customer. As a result, these events can be processed in parallel by more than one File2E and Event Server.</p>  <p><i>Note: If after the events are sorted by event start time, consecutive events belong to the same customer, it is highly probable that a file will contain a large number of events of one customer. In this case, performance will not be substantially improved.</i></p>

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
RER	rer.config	HUGE_PARTITION	N	Indicates whether Rejected Event Recycler must stop processing a partition after a certain amount of data and not scan it for a long time.
RER	rer.config	MULTIFORMAT	N	Indicates whether output files contain records of the same format or of different formats: <ul style="list-style-type: none"> ■ <i>Y</i> – The Rejected Event Recycler creates an output file per file alias (target). Using an exit point implementation, the process writes the records to a corresponding file depending on the format. ■ <i>N</i> – The Rejected Event Recycler creates files with the same format for all file aliases (targets).
RER	rer.config	ProcessingStatus	'FI'	Defines which records are to be handled. All records with the specified status are processed. Several values can be specified, separated by a comma (for example, 'FI','CO').
RER	rer.config	RER_WHERE_CLAUSE	<Empty>	The filtering criteria in the form of an SQL query. This additional clause is appended to the WHERE condition of the query that is used to get the required rejected events for processing. As a result, rejected events can be selected by other criteria, such as event type, and not only by status. The value of this parameter must start with either 'AND' or 'OR' as necessary.
RER	rer.config	USE_CONTROL_TABLE	Y	Indicates whether the Rejected Event Recycler must scan the control table ('Y') or poll the Rejected Events table for new records directly ('N').
RER	rer.controltable.polling	interval	300	The time interval (in seconds) at which the ADJ1_RER_CONTROL table is to be polled. This parameter is relevant for the daemon mode only.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
RER	rer.cycle	code	1,2,3,4,5,99	The comma-separated range of cycle codes to be handled by the Rejected Event Recycler. For example, if this parameter is configured as '1-5, 8, 10-15', the process handles cycles 1,2,3,4,5,8,10,11,12,13,14, and 15. Invalid cycle codes in the range specified in the configuration parameter are ignored.
RER	rer.cycle	instance	ALL	The cycle instance(s) to be handled by the Rejected Event Recycler. The value of this parameter can be a specific instance or 'ALL'.
RER	rer.file	writeErrorDesc	N	Indicates whether the error description is written in the output file.
RER	rer.file	writeUOMs	Y	Indicates whether the units of measurement must be written to the output file.
RER	rer.file.records	max	5000	The maximum number of records in the output file containing rejected events.
RER	rer.file.timeout	threshold	60000	The timeout threshold (in milliseconds) for the file closing policy.
RER	rer.purge	filetype	0	For purged events, indicates whether dummy files or actual physical files with purged records formatted using Simple Text Protocol are created. Values are: <ul style="list-style-type: none"> ■ 0 – Dummy files ■ 1 – Physical files with purged records
RER	rer.query.records	max	10000	The maximum number of records to be fetched in a single database query.
RER	rer.record.mandatory.field	Msg_Based_Flow_Key	Recycle	The message-based flow key for formatting events using the Text Simple Protocol. The default value must <i>not</i> be changed.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
RER	rer.resolution	code	ALL	<p>The resolution code(s) to be handled by the Rejected Event Recycler. The value of this parameter can be a specific resolution code or one of the following:</p> <ul style="list-style-type: none"> ■ ‘ALL’ – To process the records with the ‘RR’, ‘RG’ and ‘PU’ resolution code ■ ‘EMPTY’ – To process the records with the NULL resolution code ■ ‘IGNORE’ – To process all records regardless of the resolution code
RER	rer.statistics	enable	Y	Indicates whether the Rejected Event Recycler must provide statistics information.
RER	rer.statistics	numberOfFiles	100	The maximum number of statistics files that can be kept.
RER	rer.statistics	numberOfLines	100	The maximum number of lines in a statistics file.
RER	rer.statistics	TimeInterval	300	The time interval (in seconds) at which the statistics are logged.
RER	rer.thread.pool.purge.worker	max	2	The maximum size of the Purged Worker thread pool.
RER	rer.thread.pool.reader	max	2	The maximum size of the Reader thread pool.
RER	rer.thread.pool.reader	min	1	The minimum size of the Reader thread pool. Currently, this parameter is not used because the thread pool size is fixed.
RER	rer.thread.pool.worker	max	10	The maximum size of the Worker thread pool.
RER	rer.thread.pool.worker	min	1	The minimum size of the Worker thread pool. Currently, this parameter is not used because the thread pool size is fixed.

For each process instance, an entry with the name of the process instance must be created in the ADJ1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
RER146	RER

Any parameter to be overridden at the process instance level must be added to the ADJ1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
RER146	rer.statistics	enable	Y

For more information on parameters and the properties mechanism, see the “Configuration Parameter Mechanism” appendix.

Configuration Files

The log4j.properties file is used to configure the logging properties of the Rejected Event Recycler.

The ADJRERCustomizationRegistration.properties file is used to configure the exit point handler class to be used by the Rejected Event Recycler. For more information, see *Turbo Charging Customization Guide*.

Both the files are located in the <Recycler_home>/public/properties directory.

The user can configure additional core fields to be extracted (the ones that do not exist in the mapping) by modifying the gajrer/public/static/schemas/RER_AdditionalCoreFields.xml file and adding it to the classpath.

Example:

```
<Field column_name="PROCESSING_STATUS" column_attr_type="C" column_attr_name="Processing Status"/>
<Field column_name="RESOLUTION_CODE" column_attr_type="C" column_attr_name="Resolution Code"/>
```

Database Connections

The process connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)	Comments
Configuration	<ul style="list-style-type: none"> ■ AC_PGM_RULES_CONTROL ■ AC_PROGRAMS 	Select	These tables are used by the APIs of Audit & Control.
Configuration	<ul style="list-style-type: none"> ■ ADJ1_CONF_SECTION_PARAM ■ ADJ1_CONF_HIERARCHY 	Select	The Rejected Event Recycler retrieves configuration parameters from these tables.

Database Type	Table Name	Connection Type (Update, Insert, and so on)	Comments
Configuration	<ul style="list-style-type: none"> ■ APE1_UOMS ■ APE1_RATED_EVENT_MAP 	Select	The Rejected Event Recycler retrieves units of measurement and event mappings from these tables, respectively.
Configuration	GN1_SYS_IN_CONNS_CFG	Select	The ADJ1_SetAdminCommandPortNum_Sh script used by ARER1_Init_Sh (the initialization script of the Rejected Event Recycler) reads from this table and exports the admin port number in the ADMIN_COMMAND_PORT_NUM environment variable.
Configuration	<ul style="list-style-type: none"> ■ GN1_TASK_CONNECT ■ GN1_CONNECT_PARAMS 	Select	The GN1SetAllConnPrm script used by ARER1_Start_Sh (the start-up script of the Rejected Event Recycler) retrieves data from these tables and exports the environment variable for database connect codes.
Usage	AC1_CONTROL	Insert	The APIs of Audit & Control insert a record for each file registered by the Rejected Event Recycler into this table.
Usage	ADJ1_RER_CONTROL	Select, Update	The Rejected Event Recycler reads and updates the partition processing information in this table along with Amdocs Error Manager.
Usage	APE1_REJECTED_EVENT	Select, Update	The Rejected Event Recycler selects rejected events from this table and updates the processing status of recycled events.

Admin Commands

The Rejected Event Recycler supports the following admin commands. For information on how to execute the commands, see the “Handling Admin Commands” section in the “High Availability” chapter.

Command	Command Value	Description
■ REFRESH_ALL ■ REFRESH_REFERENCE	■ 1 ■ 2	Reads the updated event mapping data from the APE1_RATED_EVENT_MAP table and unit of measurement data from the APE1_UOMS table.
GRACEFULL_SHUTDOWN	100	Completes the current work, commit the work, and exits.

Examples:

- ADJ1_Send_Admin_Command_Sh hpx418 RER 146 1
- ADJ1_Send_Admin_Command_Sh hpx418 RER 146 2

Recovery Instructions

Rerun the process.

20 ADJ1NTFSRV – Notification

This chapter describes the Notification process.

Description

The Notification process is a multi-threaded component of Turbo Charging that performs the following:

1. Reads notifications and scheduled event transactions inserted into the Notifications table by the Persistence Writer Function of the Event Server.
2. Does the following:
 - For notifications:
 - i. Formats notifications using an extension function of the Implementation Compiler
 - ii. Sends formatted notifications to their relevant targets, such as the Transaction Broker or files
 - For scheduled event transactions:
 - i. Opens the instructions and separates them into fields
 - ii. Updates these fields in the Transactions table of Generic Transaction Scheduler

Normally, the Notification process handles notifications, whereas external records are handled by the Dispatcher process. However, if the number of external records is small, the Notification process may handle them as well. The Event Server publishes the dispatching records to the Dispatcher or Notification process based on the configuration in the Rating Logic Configurator implementation.

The process works according to the following granularity levels, which are specified in the form of a list in the APR1_CONF_SECTION_PARAM table:

- *All* – The process handles notifications for all targets under all cycle codes.
- *Cycle code* – The process handles notifications for all the targets under this cycle code.
- *Cycle code and target ID* – The process handles notifications for a specific target under a specific cycle code. Because the Notification process works on all Transaction Broker targets at once, for these targets, the responsibility of the process is defined as *<cycle>_TRB* (there is no need to list all targets).

The process supports parallelism by activating one or more threads per cycle and target. The threads are allocated using routing keys – granularity values that define sub-partitions of cycle codes and target IDs.

Process Type

Daemon

Run Frequency

Runs continuously

Activation

Command line:	ADJ1_NTF_Daemon_Shell_Sh -n <APPLICATION_ID> -c “-f <CCF_FILE_NAME>” -e <EWD_EXE>
	<i>Note: The -e parameter is optional. It is used in environments with Availability Manager.</i>
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_NTF_Daemon_Shell_Sh
Executable Name:	gcpf1fwcApp

Example:

ADJ1_NTF_Daemon_Shell_Sh -n NTF1510 -c “-f APR1_NTF.ccf” -e gn1avm_ewd

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> NTF \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes

Example:

ADJ1_Send_Admin_Command_Sh 10.232.53.45 NTF 1510
CPF_GracefulShutdownCommand

Preceding Processes

The following jobs must run successfully before this process is activated, and the following daemons must be active for this process to run:

- *Event Server* – Creates notifications and writes them to the Notifications table.
- *Reference Table Synchronization (RTS) process* – Copies the contents of the PC1_DISP_TARGET table into the APR1_DISP_TARGET table. The rules for copying the data are defined by implementation. The Notification process supports only notification, dispatching record, and scheduled event targets, which are defined in the LOGICAL_FORMAT field as ‘N’. External records (‘X’) are processed by the Dispatcher (for more information, see the “ADJ1DISPSRV – Dispatcher” chapter).

Affected Applications

The following applications are affected by the Notification process:

- All applications that subscribe to the notification transactions in the Transaction Broker
- All applications (defined in the NXT_PGM_NAME field of the AC_PGM_RULES_CONTROL table) that read notification files from the Audit & Control table
- Generic Transaction Scheduler, which reads the details of scheduled event transactions from the Transactions table populated by the Notification process

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*
ADJ1NTFSRV_<APPLICATION_ID>_%D_%T_%n.log
Example:
ADJ1NTFSRV_NTF1510_20081109_130927_1.log
- *Console log files:*
ADJ1_NTF_Daemon_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log
Example:
ADJ1_NTF_Daemon_inner_Sh_ NTF1510_20081109_130927.log
- *Operational log files:*
ADJ1NTFSRV_<APPLICATION_ID>_\${ABP_MARKET}_<Date>_<Time>.log
Example:
ADJ1NTFSRV_NTF1510_M3G_20081109_130927.log
- *Environment variable log files:*
ADJ1_NTF_Daemon_Shell_Sh_<APPLICATION_ID>_<Date>_<Time>.log
Example:
ADJ1_NTF_Daemon_Shell_Sh_ NTF1510_20081109_130927.log
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG
- *Environment variable log files* – \$ABP_LOG
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter)

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information.
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).
- *Operational log files* – The output of operational scripts.
- *Environment variable log files* – Environment variables used by the process.
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Output Files

The process creates output files for notifications with target IDs whose LOGICAL_FORMAT = ‘N’ (notification) and CHANNEL = ‘FI’ (file) in the APR1_DISP_TARGET table.

Name

FilePrefix
<CycleID><TargetID>_<RoutingKey>_<ProcessID>_<ThreadID>_<Date>_<Time>.dat

The file’s prefix is defined in the FILE_PREFIX column of the specific target ID in the APR1_DISP_TARGET table.

Location and Format

- *The location of the source file before it is stored in Audit & Control* – Defined in the APR1_CONF_SECTION_PARAM table as \$ABP_APP_ROOT/interfaces/source/
- *The location of the destination file after it inserted into Audit & Control* – \$ABP_APP_ROOT/<path from the FILE_PATH column of the specific target ID in the APR1_DISP_TARGET table>
- *Format – USAGE*

Contents

This file contains the formatted notifications for a specific target. The definitions of the notification format are in the generated code that was created from the Format XML file.

Flow

The Notification process includes the following modules:

- *Manager module* – Controls the work of notifications and scheduled event transactions. This module is a single-thread context.
- *Reader module* – Reads notifications and scheduled event transactions from the Notifications table. The number of Reader threads is defined in configuration.
- *Synchronization module* – Synchronizes the order of notifications and scheduled event transactions per cycle code. This module is a single-thread context.
- *Worker module* – Formats notifications, opens the transaction instructions, and sends them to the correct target. The number of Worker threads is defined in configuration.
- *Recovery module* – Recovers notifications and transaction that were not handled as a result of failure. The number of Recovery threads is the same as that of the Worker threads.

This section describes the flow of the Notification process.

Process Start-up

At start-up, the Notification process performs the following steps:

1. Connects to the relevant Usage database using an OCI connection.
2. Selects all notification and scheduled event targets and their configuration from the APR1_DISP_TARGET table.
3. Loads the formatting libraries of the Implementation Compiler.
4. Runs the Worker and Recovery thread division algorithm. This algorithm makes sure that each worker or recovery thread is responsible for one or more combinations of cycle code, target ID, and routing key (CTR).
5. For each CTR, initializes a map of safe points from the APR1_NOTIFICATIONS_CTRL table, with the last handled notification sequence number (update ID) of the CTR and its processing mode (Regular or Recovery).

Each module in the process has its own start-up flow described in subsequent sections.

Manager Module

The Manager thread performs the following:

1. Initializes the cycle's vector and connects to the relevant Usage databases. It uses the same connection for all cycles in the same Usage database.
2. Using the map of safe points for all CTRs, initializes the safe point of the last handled notification sequence number (update ID) and its day-of-month key per cycle code.

Reader Module

Each Reader thread performs the following steps:

1. Connects to the relevant Usage database
2. Gets the list of targets per cycle code

Synchronization Module

The Synchronization thread performs the following:

1. Gets the list of the cycles for which it is responsible.
2. Gets the list of Worker threads and their CTRs.
3. Prepares an array per cycle code. This array holds the notification sequence numbers with gaps.

Worker Module

Each Worker thread performs the following steps:

1. Gets the list of the CTRs for which it is responsible.
2. Connects to the relevant Usage databases.
3. Creates outputs for each CTR. For Transaction Broker targets, the thread creates only one output per Usage database.
4. Initializes the outputs.

Recovery Module

At start-up, each Recovery thread gets the list of CTRs for which it is responsible.

Main Loop

Each module in the process has its own main loop, and all modules work in parallel.

Manager Module

The Manager thread loops on each cycle in the cycle's vector and performs the following:

1. Reads the bulk of update IDs that are pending work from the APE1_NOTIFICATIONS_CTRL table. The update IDs are selected according to the day-of-month key and cycle code, and their notification sequence numbers must be larger than the last safe point.
2. For each update ID in the bulk, attaches an increasing internal sequence number that is dedicated to the cycle.
3. Inserts this information into the queue of the Reader thread.

Reader Module

Each Reader thread performs the following steps:

1. Gets the next item from the queue
2. From the APE1_NOTIFICATIONS table, selects all notifications and scheduled events that are related to a specific day-of-month key, cycle code, update ID, and the related targets
3. Inserts this information into the queue of the Synchronization thread

Synchronization Module

The Synchronization thread performs the following:

1. Gets the next item from the queue.
2. Checks whether the notification sequence number has already been inserted into the queue of the Worker thread.
 - If so, drops the item.
 - If not, checks whether there is a gap in the order of the internal sequence numbers for the cycle.
 - If there is a gap, inserts the item into the array. The item remains in the array until the gap is closed.
 - If there is no gap, distributes the notifications to the queue of the relevant Worker thread until the next gap.

Worker Module

Each Worker thread performs the following steps:

1. Gets the next item from the queue. Each item contains a list of notifications or scheduled event transactions.
2. For each notification or scheduled event transaction, performs the following:
 - a. Gets the relevant CTR from the CTR map.
 - If the notification or scheduled event transaction has already been handled, drops the notification or transaction.
 - If the Usage database failed, drops the notification or transaction.
 - b. Writes the notification to the relevant output and checks whether the formatting and writing of the notification or scheduled event transaction to the output was successful:
 - If it was successful, increases the count of successful notifications or transactions for this CTR
 - If it failed with an applicative error, adds the notification or transaction to the Applicative Erred Notification queue for this CTR
 - If it failed with a system error, resets the output and updates the status for the update IDs in the CTR map to ‘RC’ (Recovery)
3. Every configurable period of time:
 - a. Commits all outputs
 - b. Checks the return status of the commit:
 - If it failed with an applicative error, adds all notifications to the Applicative Erred Notification queue of the appropriate CTRs.
 - If it failed with a system error:
 - 1) Resets the outputs
 - 2) Updates the status of the update IDs in the CTR map to ‘RC’ (Recovery)
 - c. Inserts an entry for each update ID in the CTR to the APR1_NOTIFICATIONS_CTRL table, indicating its status (‘CO (Complete) or ‘RC’ (Recovery)) and the number of successful and failed notifications and scheduled event transactions.
 - d. Notifies the Recovery module of the CTR in the ‘RC’ (Recovery) status.
 - e. Inserts the update IDs in each CTR that failed with an applicative error to the APR1_REJECTED_NOTIFICATION table.
 - f. Commits the data to all relevant Usage databases.

Recovery Module

Each Recovery thread performs the following steps:

1. For CTRs that must be recovered, gets notifications and scheduled event transactions from the Worker module.
2. For each recovered CTR, checks whether there are entries in the APR1_NOTIFICATIONS_CTRL with the status of ‘RC’ (Recovery).
3. If there are such entries, attempts to process the notifications or transactions.
4. If possible, performs the same flow of writing the notification or transaction to its output as in the Worker module. Otherwise, continues to the next CTR.
 - a. Updates the entry in the APR1_NOTIFICATIONS_CTRL table with the ‘CO’ (Complete) status and the number of successful and failed notifications or transactions
 - b. Inserts the update IDs in each CTR that failed with an applicative error to the APR1_REJECTED_NOTIFICATION table
 - c. Commits the data to all relevant Usage databases
 - d. Notifies the Worker module regarding the recovered CTR

Process Shutdown

Because the Notification process is a daemon, it only shuts down if it receives a specific manual request to do so. No special actions are performed during shutdown. The Worker module updates the APR1_NOTIFICATIONS_CTRL table with the handled notifications and scheduled event transactions for each CTR.

The subsequent Notification run continues processing each CTR exactly from the place at which the previous run left off.

Environment Variables

The following environment variables affect the process.

Name	Description	Default Value	Method of Changing the Variable
APR_PW_LF_NO_COMMIT_TIME_OUT_DELTA	The timeout from the last commit, in milliseconds, after which the queue sends an expiration message to the Persistence Writer.	1000	N/A
APR_NTF_READER_NUM_OF_THREADS	The number of Reader threads	5	N/A
APR_NTF_WORKER_NUM_OF_INSTANCES	The number of Worker instances	1	N/A

Name	Description	Default Value	Method of Changing the Variable
APR_NTF_SCHEDULER_PARTITION_FACTOR	<p>The number of subpartitions in the GN1_GTS_TRX table. This number, correlated with the SCHEDULER_PARTITION_KEY column of the GN1_GTS_TRX table, determines the number of Notification Worker threads that handle scheduled event transactions in parallel.</p> <p>If the number of subpartitions must be different from the one provided out of the box, the GN1_GTS_TRX table must be re-created with the required number of subpartitions, and this parameter must be defined accordingly.</p>	23	The value is static and cannot be changed at runtime. It is set along with the GN1_GTS_TRX table definition.
GTS_TRX_ADDITIONAL_TIME_SECONDS	Additional time for the expiration date of recurring scheduled events.	120	N/A
NTF_DAY_PARTITIONS_NUM	<p>The number of day-of-month partitions in the following tables:</p> <ul style="list-style-type: none"> ■ APE1_NOTIFICATIONS ■ APE1_NOTIFICATIONS_CTRL ■ APR1_NOTIFICATIONS_CTRL ■ APR1_REJECTED_NOTIFICATION <p>The value of the variable is taken at process start-up from the partition definitions for the Notification tables in the Usage database. This value is also used by the Event Server. The definition of the partitions must be the same in all the Usage databases in the environment.</p>	6	The value is defined only once when building the database, and it cannot be changed.

For more information on the script, see the “Amdocs Batch Job Manager Application Script” section in the “Introduction” chapter.

Parameters

The following parameters must be set for the process to run properly.

Parameter	Description	Default Value
APPLICATION_ID	The process instance name.	N/A
CCF_FILE_NAME	An XML file that contains the relevant configuration.	APR1_NTF.ccf
NTF_CYCLES_TARGETS_LIST	The cycle, a list of cycles, or a combination of cycle and target for which the process is responsible. It is also possible to include all cycles by setting the value of this parameter to ALL. This parameter is configured in the APR1_CONF_SECTION_PARAM table.	N/A

Configuration Parameters

The configuration parameters (properties) of the Notification process are defined in the database.

The following table shows the performance tuning (configuration) parameters of the Notification process. These parameters are defined in the APR1_CONF_SECTION_PARAM table using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*). Some of the parameters are optional and have default values.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
NTF	config	DD_LIB_NAME	mpr_dd	The name of Dictionary library of the Implementation Compiler that contains all general information that is relevant for the general interface between an application and the generated code.
NTF	config	NOTIF_DD_LIB_NAME	mpr_notifications	The name of the Dictionary library of the Implementation Compiler that contains only the notification formatting functions that are relevant for the interface between the Notification process and the generated code.
NTF	TARGET_<target_id>	LIBRARY_NAME	<Customized library name>	The library name of the customized target. This entry appears only for customized output. For more information about creating a customized target, see <i>Turbo Charging Customization Guide</i> .
NTF	TARGET_<target_id>	REGISTRATOR_FUNCTION_NAME	<Registrar function name>	The name of the registrator function of the customized target. This entry appears only for customized output. For more information about creating a customized target, see <i>Turbo Charging Customization Guide</i> .
NTF	SERVER_PARAMETERS	DATALAYERS_BUFFER_SIZE	100	The size of the data layer buffer for reading records from the APE1_NOTIFICATIONS_CTRL table. The default value in the code is 100. To change this value, insert an entry into the APR1_CONF_SECTION_PARAM table.
NTF	SERVER_PARAMETERS	MAX_CURSOR_SIZE	500	The maximum size of the cursor that is opened for reading from the APE1_NOTIFICATIONS_CTRL table. The default value in the code is 500. To change this value, insert an entry into the APR1_CONF_SECTION_PARAM table.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
NTF	SERVER_PARAMETERS	MAX_UPDATE_IDS_TO_KEEP_PER_CYCLE	300	The number of entries (update IDs) to allocate in advance in the SyncManager module for each cycle code. This array is used to re-arrange the update IDs for each cycle code in the correct order. If the SyncManager gets an entry that is out of order, it keeps the entry in the array until the expected update ID arrives.
NTF	SERVER_PARAMETERS	MEM_POOL_SIZE_PER_THREAD	2000	The memory pool size per thread.
NTF	SERVER_PARAMETERS	ROOT_PATH	\${ABP_APP_ROOT}	The root path for writing the files.
NTF	SERVER_PARAMETERS	SELECT_LAG_MSEC	\${APR_PW_LF_NO_COMMIT_TIMEOUT_DELTA}	Defines the delay (in milliseconds) from the current GMT time according to which the Notification process must select records from the APE1_NOTIFICATION_CTRL table. In other words, the Notification processes selects those rows in which the value of the GMT_SYS_CREATION_DATE field is smaller than the current GMT time by this number of milliseconds.
NTF	SERVER_PARAMETERS	SCHEDULER_PARTITION_FACTOR	\${APR_NTF_SCHEDULER_PARTITION_FACTOR}	The number of subpartitions in the GN1_GTS_TRX table. This number, correlated with the SCHEDULER_PARTITION_KEY column of the GN1_GTS_TRX table, determines the number of Notification Worker threads that handle scheduled event transactions in parallel. If the number of subpartitions must be different from the one provided out of the box, the GN1_GTS_TRX table must be re-created with the required number of subpartitions, and this parameter must be defined accordingly.
NTF	SERVER_PARAMETERS	SOURCE_FILES_PATH	/interfaces/source/	The file path before it is stored in Audit & Control.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
NTF	SERVER_PARAMETERS	TARGET_LIST	TRB, TARGET_6, TARGET_21197, TARGET_21199, TARGET_30000	The list of targets that contains configuration parameters at the target level. Targets 6, 21197, and 21199 are file targets. Target 21199 is used for dispatching cases. Target 30000 is reserved for scheduled events, and it cannot be used for other purposes.
NTF	SERVER_PARAMETERS	TIME_TO_COMMIT	3	The time it takes for the Worker thread to commit data to Notifications tables (in seconds).
NTF	SERVER_PARAMETERS	UPDATE_IDS_RECOVERY_THRESHOLD	15	The number of update IDs in the APR1_NOTIFICATIONS_CTRL table for a specific CTR (cycle, target, and routing key) in RC (Recovery) status that are to be handled by the Recovery thread. If the Recovery thread reaches this threshold, it notifies the Worker thread that it must commit to the database and stop inserting entries to the APR1_NOTIFICATIONS_CTRL table for the specific CTR in RC status. The default value in the code is 15. To change this value, insert an entry into the APR1_CONF_SECTION_PARAM table.
NTF	TARGET_21197	DATA_GROUP	NTF1DATA	The prefix in the DATA_GROUP column of the AC1_CONTROL table for the file that the Notification process will register in the table.
NTF	TARGET_21197	ROUTING_CRITERIA	NTF1ROUTING_21197	The value in the ROUTING_CRITERIA column of the AC1_CONTROL table for the file that the Notification process will register in the table.
NTF	TARGET_21197	ROUTING_KEY	1	Defines the sub-partitions of cycle codes and target IDs. The routing key of each notification is calculated according to the following formula: Customer segment % The number of routing keys in the target + 1 If the entry is not defined, the default value is 1.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
NTF	TARGET_21197	WEIGHT	1	<p>The weight of the target. The Notification process allocates the targets to different Worker threads according to the weight of each target.</p> <p>There are three levels of weight:</p> <ul style="list-style-type: none"> ■ 1 – Normal (the target is not busy). ■ 2 – Medium. ■ 3 – High (the target is overloaded). <p>If the weight is not specified, the Notification process assumes that all targets have the same normal weight.</p>
NTF	TARGET_21199	DATA_GROUP	NTF1DATA	<p>The prefix in the DATA_GROUP column of the AC1_CONTROL table for the file that the Notification process will register in the table.</p>
NTF	TARGET_21199	ROUTING_CRITERIA	NTF1ROUTING_21199	<p>The value in the ROUTING_CRITERIA column of the AC1_CONTROL table for the file that the Notification process will register in the table.</p>
NTF	TARGET_21199	ROUTING_KEY	1	<p>Defines the sub-partitions of cycle codes and target IDs. The routing key of each notification is calculated according to the following formula:</p> <p>Customer segment % The number of routing keys in the target + 1</p> <p>If the entry is not defined, the default value is 1.</p>
NTF	TARGET_21199	WEIGHT	1	<p>The weight of the target. The Notification process allocates the targets to different Worker threads according to the weight of each target.</p> <p>There are three levels of weight:</p> <ul style="list-style-type: none"> ■ 1 – Normal (the target is not busy). ■ 2 – Medium. ■ 3 – High (the target is overloaded). <p>If the weight is not specified, the Notification process assumes that all targets have the same normal weight.</p>

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
NTF	TARGET_30000	DATA_GROUP	NTF1DATA	<p>The prefix in the DATA_GROUP column of the AC1_CONTROL table for the file that the Notification process will register in the table.</p> <p>Currently, this parameter is not in use.</p>
NTF	TARGET_30000	ROUTING_CRITERIA	SCHEDULER	<p>The value in the ROUTING_CRITERIA column of the AC1_CONTROL table for the file that the Notification process will register in the table.</p> <p>Currently, this parameter is not in use.</p>
NTF	TARGET_30000	ROUTING_KEY	1	<p>Defines the sub-partitions of cycle codes and target IDs. The routing key of each notification is calculated according to the following formula:</p> <p>Customer segment % The number of routing keys in the target + 1</p> <p>If the entry is not defined, the default value is 1.</p> <p>Currently, this parameter is not in use.</p>
NTF	TARGET_30000	WEIGHT	1	<p>The weight of the target. The Notification process allocates the targets to different Worker threads according to the weight of each target.</p> <p>There are three levels of weight:</p> <ul style="list-style-type: none"> ■ 1 – Normal (the target is not busy). ■ 2 – Medium. ■ 3 – High (the target is overloaded). <p>If the weight is not specified, the Notification process assumes that all targets have the same normal weight.</p> <p>Currently, this parameter is not in use.</p>
NTF	TARGET_6	DATA_GROUP	NTF1DATA	<p>The prefix in the DATA_GROUP column of the AC1_CONTROL table for the file that the Notification process will register in the table.</p>
NTF	TARGET_6	ROUTING_CRITERIA	NTF1ROUTING_6	<p>The value in the ROUTING_CRITERIA column of the AC1_CONTROL table for the file that the Notification process will register in the table.</p>

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
NTF	TARGET_6	ROUTING_KEY	1	<p>Defines the sub-partitions of cycle codes and target IDs. The routing key of each notification is calculated according to the following formula:</p> <p>Customer segment % The number of routing keys in the target + 1</p> <p>If the entry is not defined, the default value is 1.</p>
NTF	TARGET_6	WEIGHT	1	<p>The weight of the target. The Notification process allocates the targets to different Worker threads according to the weight of each target.</p> <p>There are three levels of weight:</p> <ul style="list-style-type: none"> ■ 1 – Normal (the target is not busy). ■ 2 – Medium. ■ 3 – High (the target is overloaded). <p>If the weight is not specified, the Notification process assumes that all targets have the same normal weight.</p>
NTF	TRB	CONNECT_XML	<code> \${ABP_HOME}/core/config/NTF_Connect_Publish.xml</code>	The full path and name of the Connect Publish XML file of the Notification process. This file registers the Notification process on the Transaction Broker.
NTF	TRB	ROUTING_KEY	1	<p>Defines the sub-partitions of cycle codes and target IDs. The routing key of each notification is calculated according to the following formula:</p> <p>Customer segment % The number of routing keys in the target + 1</p> <p>If the entry is not defined, the default value is 1.</p>

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
NTF	TRB	WEIGHT	1	<p>The weight of the target. The Notification process allocates the targets to different Worker threads according to the weight of each target.</p> <p>There are three levels of weight:</p> <ul style="list-style-type: none"> ■ 1 – Normal (the target is not busy). ■ 2 – Medium. ■ 3 – High (the target is overloaded). <p>If the weight is not specified, the Notification process assumes that all targets have the same normal weight.</p>
NTF1510	SERVER_PARAMETERS	NTF_CYCLES_TARGETS_LIST	1	<p>The cycle, a list of cycles, or a combination of cycle and target for which the process is responsible. It is also possible to include all cycles by setting the value of this parameter to ALL. For Transaction Broker targets, set the value to <cycle>_TRB (there is no need to list all the targets).</p> <p>This entry is defined at the process instance level. For example, the NTF1500 process and its back-up, NTF1510, have the same value.</p>

In addition, the process uses Light Tracer configuration parameters, which are defined in the ADJ1_CONF_SECTION_PARAM table. For more information, see [Light Tracer Parameters](#) in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

For each process instance, an entry with the name of the process instance must be created in the APR1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
NTF1510	NTF

Any parameter to be overridden at the process instance level must be added to the APR1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
NTF1510	TRB	WEIGHT	3

For more information on parameters and the properties mechanism, see the “Configuration Parameter Mechanism” appendix.

The Notification process reads additional target configuration parameters from the APR1_DISP_TARGET using the Reference Table Synchronizer. These parameters are:

- *FILE_PATH* – A relative path to which files are to be written.
- *FILE_PREFIX* – The file prefix of the file target. It is also used as the transaction code value for Transaction Broker notifications.
- *FILE_ALIAS* – The file alias for registering the files to Audit & Control.
- *MAX_COUNT* – The maximum number of records in a file. It is also used as the maximum number of notification records to be published to the Transaction Broker.

Configuration Files

The Notification process requires a configuration file (a .ccf file) that is defined via an internal configuration tool.

The environment variables that appear in the .ccf file and can be customized are defined in the op_adj_env_sh script (for more information, see the “Environment Variables” section in this chapter).

Database Connections

During initialization, the Notification process connects to several reference tables. For a complete description of each table, see *Turbo Charging Data Model*. After initialization, it connects primarily to the following tables.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	AC1_CONTROL	Insert
Application	GN1_GTS_TRX	Insert, Update
Application	TRB1_PUB_LOG	Insert
Reference	APR1_DISP_TARGET	Select
Usage	APE1_NOTIFICATIONS	Select
Usage	APE1_NOTIFICATIONS_CTRL	Select
Usage	APR1_NOTIFICATIONS_CTRL	Select, Insert, Update
Usage	APR1_REJECTED_NOTIFICATION	Insert

Admin Commands

The Notification process receives the following admin commands. For information on how to execute the commands, see the “Handling Admin Commands” section in the “High Availability” chapter.

Command	Description	Parameters
SET_LIGHT_TRACER_COMMAND	Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters	<ul style="list-style-type: none"> ▪ Activation parameters: <ul style="list-style-type: none"> • <i>turnTracer “on/off”</i> – Activates or stops the Light Tracer • <i>--turnAssert “on/off”</i> – Activates or stops Assertions • <i>--turnLogOnTrace “on/off”</i> – Activates or stops writing log messages in Light Tracer files • <i>--turnFlowTracer/turnAll “on/off”</i> – Activates or stops tracing a single flow or event that is transferred between machines • <i>--turnAll “on/off”</i> – Activates or stops all of the above. The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter). ▪ <i>--moduleId “ALL”</i> ▪ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • <i>--threadInfo “Y/N”</i> – Specifies whether trace or assertion messages include the thread information. • <i>--timestampInd “Y/N”</i> – Specifies whether trace or assertion messages include the time stamp. • <i>--contextInd “Y/N”</i> – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • <i>--noOfMessages “<number>”</i> – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down. • <i>--period “5/10/20/60/120/480/1440”</i> – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated.

Command	Description	Parameters
		 Note: If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached. <ul style="list-style-type: none"> • <code>--assertAbortInd "Y/N"</code> – Specifies whether the application is to be aborted if the assertion condition fails. ▪ <code>--commandType "changeBufferSize"</code> – Resizes the Light Tracer buffer at run time. • <code>--bufferSize "<number>"</code> – The new size for the Light Tracer buffer.
SET_TRACE_SQL_COMMAND	<p>Activates or stops SQL tracing at runtime. The command name with its parameters must be enclosed in quotation marks.</p> <p><i>Example:</i></p> <pre>ADJ1_Send_Admin_Command_Sh illin1027 <process instance> "SET_TRACE_SQL_COMMAND --traceSqlOn \"yes\" --duration \"1\""</pre>	<code>--traceSqlOn "<yes or no>" --duration "<minutes>"</code> If the IS_DURATION_REQUIRED environment variable is set to ‘Y’, the --duration parameter is mandatory. If the duration specified in the command exceeds the allowed maximum value, SQL tracing is activated for the maximum allowed period of time only (60 minutes).  Note: The SHUT_DOWN_CHECK_FREQ parameter of the Light Tracer in the ADJ1_CONF_SECTION_PARAM table determines the frequency (defined in the number of messages) at which the SQL Trace Manager checks whether the period specified in the --duration parameter has passed. Therefore, SQL tracing is stopped after the specified duration or after this number of messages has been reached, whichever comes later.

Command	Description	Parameters
SWITCH_LIGHT_TRACER_ON ALL	Activates the Light Tracer without the ability to specify the mode or send parameters.	N/A
SWITCH_LIGHT_TRACER_OFF ALL	Stops the Light Tracer	N/A
CPF_GracefulShutdownCommand	Shuts down the Notification process gracefully	N/A

Example:

ADJ1_Send_Admin_Command_Sh 10.232.53.45 NTF 1510 SWITCH_LIGHT_TRACER_ON ALL

Distribution

The Notification process distributes its results as follows:

- Creates files with notifications and registers them to Audit & Control
- Publishes notifications on the Transaction Broker
- Updates the Notifications tables
- Updates the Transaction table of Generic Transaction Scheduler

Screen Messages

There are no special screen messages. Log messages are also sent to the console, but they are best viewed in the log file.

Troubleshooting

The following important or frequent errors can occur during the process run.

Error	Preventing Action
A Notification instance starts running and immediately shuts down.	Search for the ERROR or FATAL string in the Notification log. The first ERROR or FATAL message probably describes the problem.
The Notification log prints that process could not be initialized.	There is probably another Notification instance (with the same instance number) already running on the same machine. Running more than one Notification instance with the same name on the same machine is not allowed. If you need to run more than one Notification instance on the same machine, change the name of one of the instances and the list of cycles and targets.
The Notification process shuts down after it has processed some notifications.	Search for the ERROR or FATAL string in the Notification log. The first ERROR or FATAL message probably describes the problem.
The Notification process inserts too many rows into the APR1_REJECTED_NOTIFICATION table.	Check the applicative error in the Description column of the APR1_REJECTED_NOTIFICATION table.
The Notification process inserts entries with Recovery status into the APR1_NOTIFICATIONS_CTRL table.	Check the log files for the reason that causes the targets to be added to the Recovery flow.
The Notification process does not insert entries into the APR1_NOTIFICATIONS_CTRL table.	Check whether there is a problem with the connection to the Usage database.

Recovery Instructions

When the Notification process starts, its recovery mechanism is automatically activated. Therefore, no special actions are needed after the process is killed or has crashed. After the process shuts down, regardless of whether it shut down gracefully or was stopped immediately, it initializes the safe point of the last handled notification sequence number (update ID) and its day-of-month key per cycle code. The process continues handling the notifications per cycle code from there.

Recovery for File Targets

If the notifications were distributed, and the files were closed, but the commit operation failed:

1. At start-up, the Recovery thread checks the directory for working files.
2. If the directory contains files, the thread deletes them.

Recovery for Transaction Broker Targets

If failure occurred after performing a commit operation on the Transaction Broker but not on the OCI connection, the transactions were published to the Transaction Broker but not marked as completed in the control table. In this case, the transactions are published again.

The following sections describe the steps that are performed by the Notification modules for error handling and recovery.

Recovery for Scheduled Event Transactions

No special recovery is needed. If failure occurred after performing a commit operation on the applicative connection, the transaction is published. If the same transaction is published again during recovery at start-up, the Notification process throws Oracle's unique constraint error.

Manager Module

If the home database is not available and the Notification process receives a command to switch to the alternate database, it starts working against the alternate database. If the process has not received this command, the Manager module performs the following:

1. If a database connection fails, makes three attempts to reconnect.
2. If fails to reconnect, stops reading the bulk of update IDs that are pending work from the APE1_NOTIFICATIONS_CTRL table of the problematic database.

3. When the database connection is recovered:
 - a. Gets an admin command from the Availability Manager that the connection backed to the home database
 - b. Renews the connection
 - c. Continues reading the bulk of update IDs that are pending work from the APE1_NOTIFICATIONS_CTRL table of the home database
4. If no records are found for the day-of-month key, moves on to the next day.

Reader Module

If the home database is not available and the Notification process receives a command to switch to the alternate database, it starts working against the alternate database. If the process has not received this command, the Reader module performs the following:

1. If there are database connection problems, makes three attempts to reconnect.
2. If fails to reconnect, inserts a technical message into the queue of the Synchronization thread. This message notifies the Synchronization thread that it must stop synchronizing the order of notification sequence numbers of the cycle in the problematic database.
3. When the database connection is recovered, stops sending technical messages and handles all notifications and scheduled event transactions.

Synchronization Module

If the home database is not available and the Notification process receives a command to switch to the alternate database, it starts working against the alternate database. If the process has not received this command, the Synchronization module performs the following:

1. If there are database connection problems (indicated by a technical message regarding the problematic database from the Reader threads), deletes the data in the array that holds the notification sequence numbers with gaps for the cycle of the problematic database
2. When the database connection is recovered, stops receiving technical messages regarding the problematic database from the Reader threads, and handles all notifications and scheduled event transactions

Worker Module

If the home database is not available and the Notification process receives a command to switch to the alternate database, it starts working against the alternate database. If the process has not received this command, the Worker module performs the following:

- Database failure:
 - a. If there are database connection problems, makes three attempts to reconnect
 - b. If fails, drops all the notifications and scheduled event transactions from the cycles of the problematic database
 - c. When the database connection is recovered, handles all notifications and scheduled event transactions
- Target failure:
 - a. In the case of a target failure, such as if the file system is full or the Transaction Broker connection has failed, notifies the Recovery module regarding the CTRs of the problematic target
 - b. Gets the notifications and scheduled event transactions from the Recovery module pertaining to the CTRs that were recovered

Recovery Module

If the home database is not available and the Notification process receives a command to switch to the alternate database, it starts working against the alternate database. If the process has not received this command, the Recovery module performs the following:

1. If there are database connection problems, makes three attempts to reconnect
2. If fails, stops processing the notifications and scheduled event transactions from the cycles of the problematic database
3. When the database connection is recovered, handles all notifications and scheduled event transactions

Important Remarks

This section provides additional information on the Notification process.

Defining Multiple Instances of Notification Process

Multiple instances of the Notification process may run simultaneously on the same machine, or on different machines. Each Notification instance must be defined in the routing tables (the set of tables that are used by Availability Manager). The names of these tables begin with GN1_SYS_. There must be a separate row for each instance of the Notification process. The PROCESS_INSTANCE_ID (or sometimes PROCESS_EXEC_ID) field in the routing tables must contain the name of the instance, and it must be the same value as in the APPLICATION_ID parameter in the process start-up command line (for example, NTF1510).

Different instances of the Notification process must not run with the same cycle_target list.

Example:

The NTF1500 process runs with the following cycle_target list: 1, 2_3. This means that that the process is to handle all notifications for all targets in cycle 1 and all notifications for target 3 in cycle 2. NTF1510 cannot run with the cycle_target list of 1_3 (meaning that the NTF1510 process is to handle all notifications for target 3 in cycle 1). The reason is that the NTF1500 process handles all notifications in cycle 1; therefore, it is not necessary to specify that NTF1510 is to handle the notifications for target 3 in the same cycle.

Clean-up Job

There is an external daily clean-up job which is responsible for cleaning the processed notifications from the tables. The configuration of the job includes a parameter that specifies how many days of history may be kept in the Notification tables. The value cannot be more than 25, and the recommended number of days is 3.

The Clean-up job performs the following steps:

1. Connects to relevant Usage database.
2. Calculates the partition of the day-of-month key that is to be truncated. The calculation is relative to the current batch date, according to the number of history days to keep.
3. Checks whether all the notifications and scheduled event transactions in the day-of-month partition to truncate have been processed. To do this, compares the count of notifications and transactions in the APE1_NOTIFICATIONS_CTRL table to the count of successful and erred notifications and transactions in the APR1_NOTIFICATIONS_CTRL table.
 - If all notifications and transactions were processed, truncates this day-of-month partition from the following tables:
 - APE1_NOTIFICATIONS
 - APE1_NOTIFICATIONS_CTRL
 - APR1_NOTIFICATIONS_CTRL
 - APR1_REJECTED_NOTIFICATION
 - If not all the notifications were processed, the partition is not deleted from these tables.

21 ADJ1DISPSRV – Dispatcher

This chapter describes the Dispatcher process.

Description

The Dispatcher process is a multi-threaded component of Turbo Charging that performs the following:

1. Waits for connection requests from the Event Server(s). For every connection that the Persistence Writer maintains with the Usage databases, the Event Server requests to establish a connection with the Dispatcher. This is because there is a one-to-one correspondence between Persistence Writer transactions and Dispatcher sessions.
2. Upon connecting, receives external records from the Persistence Writer Function of the Event Server.
3. Formats external records using an extension function of the Implementation Compiler, and writes the formatted records to their target physical files. If an external record contains a real-time notification, the Dispatcher sends the notification to a Web service.
4. When the session is complete, verifies that all target files are flushed and reported to Audit & Control and then to Event-Level Auditing. Before a UDP message is sent to the ELA Collector, it is written into the APR1_DISP_ELA_RECOVERY table for recovery of event-level auditing data if the Dispatcher fails.
5. If the connection to an Event Server is dropped, or the Event Server fails to connect, the Persistence Writer writes the external records to the database. Later, the Dispatcher in Degraded mode reads these records and processes them as if they were received from the Event Server.

The process supports parallelism by activating multiple threads that concurrently handle a large number of connections with the Event Server(s).

Process Type

Daemon

Run Frequency

Runs continuously

Activation

Command line:	ADJ1_Dispatcher_Daemon_Shell_Sh -n <APPLICATION_ID> -c "-f <CCF_FILE_NAME>" -e <EWD_EXE>  Note: The -e parameter is optional. It is used in environments with Availability Manager.
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Dispatcher_Daemon_Shell_Sh
Executable Name:	gcpf1fwcApp

Example:

```
ADJ1_Dispatcher_Daemon_Shell_Sh -n DISP0777 -c "-f APR_Dispatcher.ccf" -e  
gn1avm_ewd
```

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> DSP \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes

Example:

```
ADJ1_Send_Admin_Command_Sh 10.232.53.45 DISP 0777  
CPF_GracefulShutdownCommand
```

Affected Applications

The Dispatcher process affects all applications (defined in the NXT_PGM_NAME field of the AC_PGM_RULES_CONTROL table) that read files from the Audit & Control table.

In addition, it affects the Web services that receive real-time notifications.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*

ADJ1DSPSRV_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1DSPSRV_DISP0777_20081109_130927_1.log

- *Console log files:*

ADJ1_Dispatcher_Daemon_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1_Dispatcher_Daemon_inner_Sh_DISP0777_20081109_130927.log

- *Operational log files:*

ADJ1DSPSRV_<APPLICATION_ID>_\${ABP_MARKET}_<Date>_<Time>.log

Example:

ADJ1DSPSRV_DISP0777_M3G_20081109_130927.log

- *Event log files:*

ADJ1DSPSRV_EVENT_LOG_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1DISPSRV_EVENT_LOG_DISP0777_20110522_124354_28.log

- *Latency log files:*

ADJ1DSPSRV_LATENCY_LOG_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1DSPSRV_LATENCY_LOG_DSP105_20110524_101125_0.log

- *Real-time notification latency log files:*

RTNLatency_YYYYMMDD_HHMMSS.log

Example:

RTNLatency_20140930_122700.log

- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG

- *Event log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
 -  Note: Event logs are also printed to the Light Tracer output when the Light Tracer is activated. For more information about activating the Light Tracer, see the “Admin Commands” section in this chapter.
- *Latency log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Real-time notification latency log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter)

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands)
- *Operational log files* – The output of operational scripts
- *Event log files* – Comma-separated technical information about each record processed by the Dispatcher:
 - Whether the record was successfully processed or aborted
 - Whether the record was processed online or in the Degraded mode
 - Session ID
 - Event ID
 - Source ID
 - Target ID
 - File name



Note: The file name does not appear in the Dispatcher event log that is part of the Light Tracer log file.

- *Latency log files* – Latency information about each session processed by the Dispatcher:
 - The ID of the session.
 - The mode in which the session was processed:
 - Network
 - Degraded
-  Note: *Currently, the latency log is generated only for sessions processed in the Network mode.*
- The final status of the session:
 - SUCCESS
 - ABORTED_ON_START
 - ABORTED
 - ABORTED_BY_REQUEST
-  Note: *Currently, the latency log is generated only for sessions whose status is SUCCESS.*
- The ID of the Event Server that processed the session.
 - The ID of the Persistence Writer that processed the session.
 - The number of times that the target files of the session were checked for closing according to the configured time-out.
 - The number of records in the session that were processed.
 - Information about the Start Session message, which is a message that the Event Server sends to the Dispatcher upon starting a new session:
 - The time stamp (in nanoseconds, from 1970) when the Dispatcher retrieved the message from the TCP stream and put it into the processing queue.
 - The time for which the message waited in the processing queue.
 - The time it took the Dispatcher to process the message.
 - The time it took the Dispatcher to flush the data into files, close the files, remove unnecessary files, and report to Audit & Control.
 - The time it took to commit the data to the database.
 - The time it took the Dispatcher to prepare for a new session.
 - The time it took to insert data into the APR1_DISPATCHER_CTRL table.
 - The time it took to commit the changes in the APR1_DISPATCHER_CTRL table.

- Information about the Close Session message, which is a message that the Event Server sends to the Dispatcher at the end of the current session:
 - The time stamp (in nanoseconds, from 1970) when the Dispatcher put the message into the processing queue.
 - The time for which the message waited in the processing queue.
 - The time it took the Dispatcher to process the message.
 - The time it took the Dispatcher to flush the buffers to disk, close them, and update the APR1_DISPATCHER_RECOVERY table.
 - The time it took to update the APR1_DISPATCHER_CTRL table.
 - The time it took to commit the changes in the APR1_DISPATCHER_CTRL table.
 - The time it took the Dispatcher to send a response back to the Event Server.
- Information about the Abort Session message, which is a message that the Event Server sends to the Dispatcher if the Persistence Writer fails to commit the current transaction:
 - The time stamp (in nanoseconds, from 1970) when the Dispatcher put the message into the processing queue.
 - The time for which the message waited in the processing queue.
 - The time it took the Dispatcher to process the message.
 - The time it took to update the APR1_DISPATCHER_CTRL table.
 - The time it took to commit the changes in the APR1_DISPATCHER_CTRL table.
- Information about the Session Expiration message, which is an internal message within the Dispatcher that is sent periodically to check whether the target files of the sessions must be closed according to the configured time-out:
 - The time stamp (in nanoseconds, from 1970) when the Dispatcher put the message into the processing queue.
 - The time for which the message waited in the processing queue.
 - The time it took the Dispatcher to process the message.
 - The time it took the Dispatcher to check that the session had expired.
 - The time it took the Dispatcher to flush the data into files, close the files, remove unnecessary files, and report to Audit & Control.
 - The time it took to remove data from the APR1_DISPATCHER_CTRL table.
 - The time it took to commit the changes in the APR1_DISPATCHER_CTRL table.
 - The time it took the Dispatcher to close the files.
 - The time it took to update the APR1_DISPATCHER_CTRL table.

- Information about the Disconnect Session message, which is an internal message within the Dispatcher that is sent when the Event Server is disconnected from the Dispatcher:
 - The time stamp (in nanoseconds, from 1970) when the Dispatcher put the message into the processing queue.
 - The time for which the message waited in the processing queue.
 - The time it took the Dispatcher to process the message.
 - The time it took the Dispatcher to close the files, remove unnecessary files, and report to Audit & Control.
 - The time it took to report the data to Audit & Control
 - The time it took to update the APR1_DISPATCHER_CTRL table.
 - The time it took to remove data from the APR1_DISPATCHER_CTRL table.
 - The time it took to commit the changes in the APR1_DISPATCHER_CTRL table.
 - The time it took the Dispatcher to close the files.
- Information about the first message in the session:
 - The time stamp (in nanoseconds, from 1970) when the Dispatcher put the message into the processing queue.
 - The time for which the message waited in the processing queue.
- Information about the last message in the session:
 - The time stamp (in nanoseconds, from 1970) when the Dispatcher put the message into the processing queue.
 - The time for which the message waited in the processing queue.
- *Real-time notification latency log files* – Latency information about real-time notification processing:
 - The ID of the notification
 - The time stamp (in nanoseconds, from 1970) when the notification was created in the Event Server
 - An indication of whether the notification is from Degraded mode
 - The time stamp (in nanoseconds, from 1970) when the processing of the notification was completed
 - The time stamp (in nanoseconds, from 1970) when the Output Worker retrieved the notification
 - The time stamp (in nanoseconds, from 1970) when the Output Worker finished processing the notification
 - The time stamp (in nanoseconds, from 1970) when the RTN Prepare task retrieved the notification

- The time stamp (in nanoseconds, from 1970) when the RTN Prepare task finished processing the notification
 - The time stamp (in nanoseconds, from 1970) when the Dispatcher started creating a response to the notification
 - The time stamp (in nanoseconds, from 1970) when the Dispatcher finished creating a response to the notification
 - The time stamp (in nanoseconds, from 1970) when the RTN Sender retrieved the notification
 - The time stamp (in nanoseconds, from 1970) when the RTN Sender finished processing the notification
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Output Files

The Dispatcher process creates output files for external records with target IDs whose LOGICAL_FORMAT = ‘X’ (Dispatcher) in the APR1_DISP_TARGET table.



Note: If the target of an external record is a Web service, the external record is handled as a real-time notification. If keeping history is enabled, it is also written to a file. In addition, if a notification was not sent successfully, it is written to an error, expiration, or time-out file (depending on the status of the notification).

Regular Output Files

This section describes the output files created for dispatching records.

Name

FilePrefix_<Cycle ID>_<Target ID>_<Process ID>_<Thread ID>_<Date>_<Time>.dat

The file’s prefix is defined in the FILE_PREFIX column of the specific target ID in the APR1_DISP_TARGET table. The .dat suffix is only appended to the file name when no other suffix is defined for it in the APR1_DISP_TARGET table.

Location and Format

- *The location of the source file before it is stored in Audit & Control* – Defined in the APR1_CONF_SECTION_PARAM table as \$ABP_APP_ROOT/interfaces/source/
- *The location of the destination file after it inserted into Audit & Control* – \$ABP_APP_ROOT/<path from the FILE_PATH column of the specific target ID in the APR1_DISP_TARGET table>
- *Format – USAGE*

Contents

This file contains the formatted external records for a specific target. The format of the external records depends on the target and is defined in the generated code that was created from the Format XML file.

Real-Time Notification History Files

This section describes the history files that are created for real-time notifications.

Name

RTNFILE_<Process Instance>_<Process ID>_YYYYMMDDHHMMSS_<Counter>.history

Example:

RTNFILE_105_24868_20140930112815_1.history

Location and Format

- *Location – \${ABP_APP_ROOT}/work/DSP_<Instance ID>*
- *Format – USAGE*

Contents

This file contains all real-time notifications that were handled. This file is registered to the AC1_CONTROL table only if the RTN_HISTORY_ENABLE configuration parameter is set to ‘true’.

Real-Time Notification Expiration Files

This section describes the files created for expired real-time notifications.

Name

RTNFILE_<Process Instance>_<Process ID>_YYYYMMDDHHMMSS_<Counter>.acexpired

Example:

RTNFILE_105_13078_20140929092501_0.acexpired

Location and Format

- *Location – \${ABP_APP_ROOT}/work/acoutput/DSP_<Instance ID>*
- *Format – USAGE*

Contents

This file contains all real-time notifications that were expired during processing. Records are expired according to the TARGET_NOTIFICATION_TIMEOUT configuration parameter. The file is registered to the AC1_CONTROL table with the ‘HO’ (Hold) file status. For the file to be recycled, the file status must be manually changed to ‘RD’ (Ready).

Real-Time Notification Time-out Files

This section describes the files created for real-time notifications that timed out.

Name

RTNFILE_<Process Instance>_<Process ID>_YYYYMMDDHHMMSS_<Counter>.actimedOut

Example:

RTNFILE_105_13078_20140929092501_0.actimedOut

Location and Format

- *Location* – \${ABP_APP_ROOT}/work/acoutput/DSP_<Instance ID>
- *Format* – USAGE

Contents

This file contains all real-time notifications that were timed out from the Web service. Records are timed out according to the TARGET_CONNECTION_TIMEOUT configuration parameter. The file is registered to the AC1_CONTROL table with the ‘RD’ (Ready) file status.

Real-Time Notification Error Files

This section describes the error files created for real-time notifications.

Name

RTNFILE_<Process Instance>_<Process ID>_YYYYMMDDHHMMSS_<Counter>.acerrored

Example:

RTNFILE_105_13078_20140929092501_0.acerrored

Location and Format

- *Location* – \${ABP_APP_ROOT}/work/acoutput/DSP_<Instance ID>
- *Format* – USAGE

Contents

This file contains all real-time notifications that received an error response from the Web service. The file is registered to the AC1_CONTROL table with the ‘HO’ (Hold) file status. For the file to be recycled, the file status must be manually changed to ‘RD’ (Ready).

Flow

This section describes the flow of the Dispatcher process.

The Dispatcher process includes the following main modules:

- *TCP Server* – Waits for and maintains connections with the Event Server(s). This module passes session management messages and external records to downstream modules for handling. The maximum number of connection threads is hard-coded in the .ccf file.
 - *Session Manager* – Maintains a session for each active connection and processes the external records received for it. The number of Session Manager threads is defined by the APR_DISP_SESS_MNGR_NUM_OF_THREADS environment variable.
 - *Output Worker* – Formats all external records for a single session and does the following:
 - If the target is not a Web service, sends the external records to multiple physical targets belonging to the session. This module also registers closed files in the Audit & Control database.
 - If the target is a Web service, sends the external records to the RTN Prepare task.
- This task runs in the thread context of the relevant Session Manager.
- *Physical File* – Manages the state of a single physical file, with revision and write thresholds. There is Physical File module per target ID per session. This module runs in the thread context of the relevant Session Manager.
 - *RTN Prepare* – Calls the relevant exit point to format real-time notifications for the target Web service.
 - *RTN File Handler* – Writes real-time notifications to a memory-mapped file, which is used during recovery.
 - *RTN Recovery Handler* – In recovery mode, reads the memory-mapped file with real-time notifications and checks whether each notification was handled. If a notification was not handled, the RTN Recovery Handler sends it to the RTN Prepare task.
 - *RTN Recycler* – Reads real-time notifications from the expiration, time-out, and error files in the AC1_CONTROL table and sends these notifications to the RTN Prepare task.
 - *RTN Sender* – Sends the formatted real-time notifications to the target Web service.
 - *Degraded Mode* – Checks whether any sessions and external records were written to the database (for example, because of a communication failure between the Event Server and the Dispatcher). This module gets this data and inserts it into the normal flow of the Dispatcher. The number of Degraded Mode acceptors is defined by the APR_DISP_DEGR_ACC_NUM_OF_INSTS environment variable.
 - *Recovery* – Upon start-up or re-assignment of an Event Server, verifies that all files from the previous sessions have been closed and resolved.
 - *ELA Recovery* – Ensures the high availability of event-level auditing data at the Dispatcher reporting point. This is not part of the development. Not sure why you have added it.

Process Start-up

At start-up, the Dispatcher process performs the following steps:

1. Configures the TCP Server module to enable it to receive connections
2. Loads the list of connections to the Usage databases for the cycles handled by this instance of the Dispatcher.
3. Selects all relevant targets for external records from the APR1_DISP_TARGET table
4. Loads the current version of the formatting libraries generated by the Implementation Compiler
5. Creates an internal data model used by all the modules that are part of the flow
6. Initializes the Degrade Mode module
7. Initiates the Recovery module
8. Initiates the ELA Recovery module
9. If the RTN_ENABLE configuration parameter is set to ‘true’, initializes the memory-mapped files used by the RTN File Handler task
10. If the RTN_ENABLE configuration parameter is set to ‘true’, initializes the RTN Recovery Handler module

Each module in the process has its own start-up flow described in subsequent sections.

TCP Server Module

At start-up, the TCP Server module starts listening on the configured port for connection requests.

Session Manager Module

At start-up, the Session Manager module creates threads that listen on the queue for events.

Degrade Mode Module

At start-up, the Degrade Mode module creates a list of Usage databases for the Event Servers under the responsibility of the Dispatcher.

Recovery Module

At start-up, the Recovery module finds any files that were not completed in the previous run and registers them in the Audit & Control database.

ELA Recovery Module

At start-up, the ELA Recovery module reads the records from the APR1_DISP_ELA_RECOVERY table, creates messages, and sends them to the ELA Collector.

RTN Recovery Handler Module

At start-up, the RTN Recovery Handler module reads real-time notifications from the memory-mapped file and checks whether the notifications have been sent.

Main Loop

Each module in the process has its own main loop, and all modules work in parallel.

TCP Server Module

Each TCP Server thread performs the following steps:

1. Gets the next item from the queue
2. Forwards messages containing external records and service requests to the relevant queues

Session Manager Module

Each Session Manager thread performs the following:

1. Gets the next item from the queue
2. Handles the service requests (mostly to manage an actual session)
3. Synchronously passes the external records to the Output Worker module

Output Worker Module

Each Output Worker thread accepts messages from the Session Manager and calls the relevant API for physical targets. If the target is a Web service, the Output Worker forwards the message to the RTN File Handler module.

Physical File Module

Each Physical File thread maintains the physical file in the file system (including open, write, and close operations) and checks for file thresholds.

RTN File Handler Module

Each RTN File Handler thread writes real-time notifications to a memory –mapped file and forwards the file to the RTN Prepare module.

RTN Recycler Module

Each RTN Recovery thread periodically checks the AC1_CONTROL table for expiration, time-out, and error files, reads the records from the files and sends these records to the RTN Prepare module.

RTN Prepare Module

Each RTN Prepare thread reads the external record from the memory-mapped file and calls the relevant exit point to format the message for the specific Web service according to the target ID.

RTN Sender Module

Each RTN Senders thread reads the formatted real-time notification from the queue and sends it to the target Web service. If a notification is expired, timed-out, or erroneous, it is written to the corresponding file.

Degraded Mode Module

Each Degraded Mode module connects to one of the listed Usage databases and checks whether it contains any sessions that were committed to it. For each such session found, it creates a Dispatcher session, retrieves the relevant external records, and dispatches them to their targets. After handling all degraded sessions for this Usage database, the module closes the Usage database connection and checks the next database in the list.

Process Shutdown

Because the Dispatcher process is a daemon, it only shuts down if it receives a specific manual request to do so. When a shutdown is requested, all current sessions are gracefully closed.

Environment Variables

The Dispatcher process uses the following environment variables that are initialized by the op_adj_env_sh script.

Name	Description	Default Value
APR_DISP_ABORT_SES_ON_DUP	Indicates whether the Dispatcher process must check for duplicate records (with the same event ID and target) and abort the session if a duplicate is found.	Y
APR_DISP_CLNUP_IWD_TASK_HANGUP	The time period for which the Clean-up task threads can run without sending a heartbeat to the Internal Watchdog. If the task threads do not send a heartbeat to the Internal Watchdog within this time period, the task is considered blocked.	300
APR_DISP_CLNUP_IWD_TASK_WEIGHT	The weight of the Clean-up task. If (<i><Number of blocked threads></i> / <i><Total number of threads></i>) * Weight >= 100, the Availability Manager is notified. For example, if a task has four threads, two of which are blocked, the Availability Manager is notified only if the weight is configured as 200 or if all the threads become blocked. The default value of 100 means that only when all the threads are blocked, the Availability Manager is notified.	100
APR_DISP_DEGR_ACC_IWD_TASK_HANGUP	The time period for which the Degrade Mode Acceptor task threads can run without sending a heartbeat to the Internal Watchdog. If the task threads do not send a heartbeat to the Internal Watchdog within this time period, the task is considered blocked.	300

Name	Description	Default Value
APR_DISP_DEGR_ACC_IWD_TASK_WEIGHT	<p>The weight of the Degrade Mode Acceptor task. If $(\langle \text{Number of blocked threads} \rangle / \langle \text{Total number of threads} \rangle) * \text{Weight} \geq 100$, the Availability Manager is notified. For example, if a task has four threads, two of which are blocked, the Availability Manager is notified only if the weight is configured as 200 or if all the threads become blocked.</p> <p>The default value of 100 means that only when all the threads are blocked, the Availability Manager is notified.</p>	100
APR_DISP_DEGR_ACC_NUM_OF_INSTS	<p>The number of Degrade Mode Acceptors created by the .ccf file (a single thread each). Because Dispatcher for End of Cycle handles only one cycle connection to the Usage database, the value for the End-of-Cycle mode must be ‘1’.</p>	1
APR_DISP_DEGR_ACC_THREADS_TIMEOUT	<p>The time-out of Degrade Mode Acceptor threads, in milliseconds.</p>	5000
APR_DSP_FILEHANDLER_IWD_TASK_HANGUP	<p>The time period for which the RTN File Handler task threads can run without sending a heartbeat to the Internal Watchdog. If the task threads do not send a heartbeat to the Internal Watchdog within this time period, the task is considered blocked.</p>	300
APR_DSP_FILEHANDLER_IWD_TASK_WEIGHT	<p>The weight of the RTN File Handler task. If $(\langle \text{Number of blocked threads} \rangle / \langle \text{Total number of threads} \rangle) * \text{Weight} \geq 100$, the Availability Manager is notified. For example, if a task has four threads, two of which are blocked, the Availability Manager is notified only if the weight is configured as 200 or if all the threads become blocked.</p> <p>The default value of 100 means that only when all the threads are blocked, the Availability Manager is notified</p>	100
APR_DISP_LATENCY_LOG_LEVEL	<p>Indicates whether a latency log is generated ('1') or not ('0').</p>	0
APR_DISP_RECVR_ACC_IWD_TASK_HANGUP	<p>The time period for which the Recovery Acceptor task threads can run without sending a heartbeat to the Internal Watchdog. If the task threads do not send a heartbeat to the Internal Watchdog within this time period, the task is considered blocked.</p>	300
APR_DISP_RECVR_ACC_IWD_TASK_WEIGHT	<p>The weight of the Recovery Acceptor task. If $(\langle \text{Number of blocked threads} \rangle / \langle \text{Total number of threads} \rangle) * \text{Weight} \geq 100$, the Availability Manager is notified. For example, if a task has four threads, two of which are blocked, the Availability Manager is notified only if the weight is configured as 200 or if all the threads become blocked.</p> <p>The default value of 100 means that only when all the threads are blocked, the Availability Manager is notified.</p>	100

Name	Description	Default Value
APR_DSP_SENDER_IWD_TASK_WEIGHT	The weight of the RTN Sender task. If (<i><Number of blocked threads> / <Total number of threads></i>) * Weight ≥ 100 , the Availability Manager is notified. For example, if a task has four threads, two of which are blocked, the Availability Manager is notified only if the weight is configured as 200 or if all the threads become blocked. The default value of 100 means that only when all the threads are blocked, the Availability Manager is notified.	100
APR_DSP_SENDER_IWD_TASK_HANGUP	The time period for which the RTN Sender task threads can run without sending a heartbeat to the Internal Watchdog. If the task threads do not send a heartbeat to the Internal Watchdog within this time period, the task is considered blocked.	300
APR_DISP_SESS_Mngr_IWD_TASK_HANGUP	The time period for which the Session Manager task threads can run without sending a heartbeat to the Internal Watchdog. If the task threads do not send a heartbeat to the Internal Watchdog within this time period, the task is considered blocked.	300
APR_DISP_SESS_Mngr_IWD_TASK_WEIGHT	The weight of the Session Manager task. If (<i><Number of blocked threads> / <Total number of threads></i>) * Weight ≥ 100 , the Availability Manager is notified. For example, if a task has four threads, two of which are blocked, the Availability Manager is notified only if the weight is configured as 200 or if all the threads become blocked. The default value of 100 means that only when all the threads are blocked, the Availability Manager is notified.	100
APR_DISP_SESS_Mngr_NUM_OF_THREADS	The number of threads allocated by the Session Manager to handle connections from the Event Server.	1
APR_RTN_RECYCLER_ENABLED	Indicates whether real-time notification recycling is enabled.	N
RTN_ER_FILE_STRUCTURE	The location and name of the file that contains the external record structure for real-time notifications.	\${ABP_BIN}/RTN_ER_structure.txt
RTN_OUTPUT_FILE_DIR	The directory for real-time notification output files	\${ABP_AP_ROOT}/work/
RTN_PREPRATOR_NUM_OF_THREADS	The number of threads for the RTN Prepare task.	2
RTN_SENDER_NUM_OF_THREADS	The number of threads for the RTN Sender task.	4

For more information on the script, see the “Amdocs Batch Job Manager Application Script” section in the “Introduction” chapter.

In addition, the following environment variables affect the process.

Name	Description	Default Value	Method of Changing the Variable
APE1_DISABLE_CONVERT	Disables the Data Upgrader of the Implementation Compiler.	Y	For this variable to be used, it must be added to the op_pm_en_sh script.
DSP_DAY_PARTITIONS_NUM	The number of day-of-month partitions in the following tables: <ul style="list-style-type: none">■ APR1_DISPATCHING_RECORDS■ APR1_DISPATCHER_CTRL The value of the variable is taken at process start-up from the partition definitions for the Dispatcher tables in the Usage database. This value is also used by the Event Server. The definition of the partitions must be the same in all the Usage databases in the environment.	6	The value is defined only once when building the database, and it cannot be changed.
DSP_ENABLE_EVENT_LOG	Enables the event log of the Dispatcher.	N	N/A
ISDISPOCMODE	Indicates whether the Dispatcher is running in the End-of-Cycle mode.	N	N/A

Parameters

The following parameters must be set for the process to run properly.

Parameter	Description	Default Value
APPLICATION_ID	The process instance name.	N/A
CCF_FILE_NAME	An XML file that contains the relevant configuration.	APR_DISPATCHER.ccf

Configuration Parameters

The configuration parameters (properties) of the Dispatcher process are defined in the database.

The following table shows the performance tuning (configuration) parameters of the Dispatcher process. These parameters are defined in the APR1_CONF_SECTION_PARAM table using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*). Some of the parameters are optional and have default values.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	config	AVM_ENABLED	N	Defines whether the Dispatcher is working with the Availability Manager (AVM).
APR	config	RADAR_ENABLED	\${RADAR_ENABLED}	Indicates whether Pre-Balance reporting is enabled. The value of this parameter is an environment variable that must be defined in runtime environments. Its default value is 'N'.
DSP	config	ABORT_SESSION_ON_DUPLICATE	\${APR_DISP_ABORT_SESSION_ON_DUP}	Indicates whether the Dispatcher process must check for duplicate records (with the same event ID and target) and abort the session if a duplicate is found.
DSP	config	DAY_PARTITIONS_NUM	6	The maximum number of partitions into which a day is divided.
DSP	config	ENABLE_EVENT_LOG	\${DSP_ENABLE_EVENT_LOG}	Indicates whether an event log is generated.
DSP	config	EOC	\${ISDISPEOCMODE}	Indicates whether the Dispatcher is running in the End-of-Cycle mode.
DSP	config	LATENCY_LOG_LEVEL	\${APR_DISP_LATENCY_LOG_LEVEL}	Indicates whether a latency log is generated ('1') or not ('0').
DSP	config	LOAD_MAPPER_LIBS	Y	A flag set by default to load the libraries generated by the Implementation Compiler for formatting external records.
DSP	config	MAX_SESSION_NUMBER	256	The maximum number of concurrent sessions that can be handled.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
DSP	config	NOTIF_DD_LIB_NAME	N/A	The name of the Dictionary library of the Implementation Compiler that contains only the formatting functions that are relevant for the interface between the Dispatcher process and the generated code.
DSP	config	RESERVE_CYCLE_SZ	200	The maximum number of concurrent cycle connections expected to be handled.
DSP	config	ROOT_PATH	\${ABP_AP_Root}	The root path for output files in the file system.
DSP	config	RTN_ENABLE	true	Indicates whether sending real-time notifications through the Dispatcher is enabled.
DSP	config	RTN_ER_STRUCTURE	\${RTN_ER_FILE_STRUCTURE}	The location and name of the file that contains the external record structure for real-time notifications.
DSP	config	RTN_FILE_COUNT	2	The number of memory-mapped files that can be created for each Session Manager thread.
DSP	config	RTN_HISTORY_ENABLE	true	Indicates whether keeping history for real-time notifications is enabled.
DSP	config	RTN_LATENCY_LOG_ENABLE	true	Indicates whether the real-time notification log is enabled.
DSP	config	RTN_LATENCY_LOG_FREQ	1	The number of real-time notification messages for which a real-time latency log must be generated.
DSP	config	RTN_NOTIFICATION_FILE_SIZE	3145728	The file size of the memory-mapped file.
DSP	config	RTN_OUTPUT_FILE_DIR	\${RTN_OUTPUT_FILE_DIR}	The path to real-time notification output files.
DSP	config	RTN_RECYC_BULK_SIZE	20	The number of real-time notifications that the RTN Recycler module can send to the RTN Prepare module in one bulk.
DSP	config	RTN_RECYCLER_ENABLED	\${APR_RTN_RECYCLE_R_ENABLED}	Indicates whether real-time notification recycling is enabled.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
DSP	config	SESSION_ACK_REPORT_THRESHOLD	1000	The number of external records after which an acknowledgment message is sent to the Event Server. If this parameter is set to 0, such messages are never sent.
DSP	config	SLEEP_SEC_BEFORE_ABORT	1	The period of time (in seconds) for which the Dispatcher sleeps before aborting the flushed session.
DSP	config	TARGET_DISCONNECT_COUNT	100	If the RTN Sender task becomes disconnected from the Web service this number of times, it tries to reconnect to the Web service after an incrementally increasing sleep period (until a maximum of 4 seconds). This parameter does not exist in the APR1_CONF_SECTION_PARAM table by default and must be added if necessary.
DSP	config	VIRTUAL_OR_SHARED_MEMORY_USED	VIRTUAL	Indicates whether the process uses virtual or shared memory.
DSP	DISPATCHER_MAPPER_PARAMS	MIN_WRITE_SIZE_THRESHOLD	5 * 1024	The minimum expected size (in bytes) of the formatted external record. When this threshold is reached in the parser buffer, a write operation is forced on the file system. This parameter is related to the PARSE_BUFFER_SIZE parameter.
DSP	DISPATCHER_MAPPER_PARAMS	PARSE_BUFFER_SIZE	100 * 1024	The size (in bytes) of the work buffer that is passed to the Implementation Compiler. If fewer or more external records are handled in each message, this value can be reduced.
DSP	ExitPoint.APR1.IDDispatcherRTNCF	CustomFlowObject	DispatcherRTNCF_Impl	The name of the CustomFlow function.
DSP	ExitPoint.APR1.IDDispatcherRTNCF	LibraryName	libgaprddispatcher.so	The name of the customized library.
DSP	LOGGER.output.DIS_EVENTLOG	BufferSize	\${BUFFER_SIZE}	The size of the temporary buffer of the event log.
DSP	LOGGER.output.DIS_EVENTLOG	Enable	Y	Enables the event log output. This parameter must always be set to 'Y'. The ENABLE_EVENT_LOG parameter actually controls the printing of messages to the log.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
DSP	LOGGER.output.DSP_EVENTLOG	Filename	<code> \${ABP_AP_R_ROOT}/log/\${GN1_TASK_NAME}_EVENT_LOG_\${APPLICATION_ID}_%D_%T_%n.log</code>	The name of the event log file.
DSP	LOGGER.output.DSP_EVENTLOG	FlushPeriod	<code> \${FLUSH_PERIOD}</code>	The interval at which the data from the temporary buffer is flushed to a file.
DSP	LOGGER.output.DSP_EVENTLOG	ImmediateFlush	<code> \${IMMEDIATE_FLUSH}</code>	Indicates whether the data is flushed to a file immediately and not periodically.
DSP	LOGGER.output.DSP_EVENTLOG	Layout	<code> Disp_EventLog_Layout</code>	The layout type that the event log uses to format messages.
DSP	LOGGER.output.DSP_EVENTLOG	MaxFileSize	<code> \${MAX_FILE_SIZE}</code>	The maximum size of an event log file.
DSP	LOGGER.output.DSP_EVENTLOG	Mode	<code> \${MODE}</code>	The working mode of the event log: synchronous or asynchronous.
DSP	LOGGER.output.DSP_EVENTLOG	Outputs	<code> DSP_EVENTLOG</code>	The output object assigned to the event log.
DSP	LOGGER.output.DSP_EVENTLOG	Severity	<code> 0</code>	The message severity threshold for the event log.
DSP	LOGGER.output.DSP_LATENCYLOG	BufferSize	<code> \${BUFFER_SIZE}</code>	The size of the temporary buffer of the latency log.
DSP	LOGGER.output.DSP_LATENCYLOG	Enable	<code> Y</code>	Enables the latency log output. This parameter must always be set to ‘Y’. The LATENCY_LOG_LEVEL parameter actually controls the printing of messages to the log.
DSP	LOGGER.output.DSP_LATENCYLOG	Filename	<code> \${ABP_AP_R_LOG}/\${GN1_TASK_NAME}_LATENCY_LOG_\${APPLICATION_ID}_%D_%T_%n.log</code>	The name of the latency log file.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
DSP	LOGGER.output.DSP_LATENCYLOG	FlushPeriod	\${FLUSH_PERIOD}	The interval at which the data from the temporary buffer is flushed to a file.
DSP	LOGGER.output.DSP_LATENCYLOG	ImmediateFlush	\${IMMEDIATE_FLUSH}	Indicates whether the data is flushed to a file immediately and not periodically.
DSP	LOGGER.output.DSP_LATENCYLOG	Layout	Disp_LatencyLog_Layout	The layout type that the latency log uses to format messages.
DSP	LOGGER.output.DSP_LATENCYLOG	MaxFileSize	\${MAX_FILE_SIZE}	The maximum size of a latency log file.
DSP	LOGGER.output.DSP_LATENCYLOG	Mode	\${MODE}	The working mode of the latency log: synchronous or asynchronous.
DSP	LOGGER.output.DSP_LATENCYLOG	Outputs	DSP_LATENCYLOG	The output object assigned to the latency log.
DSP	LOGGER.output.DSP_LATENCYLOG	Severity	0	The message severity threshold for the latency log.
DSP	TARGET_21198	TARGET_ADDR	10.232.199.34:28466	The host and port of the target Web service.
DSP	TARGET_21198	TARGET_NOTIFICATION_TIMEOUT	4000	The timeout after which a real-time notification that has not been processed is dropped.
DSP	TARGET_21198	TARGET_RETRIES	2	The number of times that a real-time notification must be resent before it is written to an error file.
DSP	TARGET_21198	TARGET_CONNECTION-TIMEOUT	1000	The timeout for the connection to the target.
DSP	TARGET_21198	TARGET_URI	/ws/ANMTriggeringService	The URI of the target Web service.
DSP	TARGET_21198	TARGET_WEIGHT	50	The weight of the target, which is a number between 1 and 100 that indicates the load on the target. The Dispatcher allocates the targets to different threads according to the weight of each target.

In addition, the process uses Light Tracer configuration parameters, which are defined in the ADJ1_CONF_SECTION_PARAM table. For more information, see [Light Tracer Parameters](#) in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

For each process instance, an entry with the name of the process instance must be created in the APR1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
DISP0777	DISP

Any parameter to be overridden at the process instance level must be added to the APR1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
DISP0777	config	DAY_PARTITIONS_NUM	4

For more information on parameters and the properties mechanism, see the “Configuration Parameter Mechanism” appendix.

For file targets, the Dispatcher process reads target configuration parameters from the APR1_DISP_TARGET using the Reference Table Synchronizer. These parameters are:

- *FILE_PATH* – The relative path to which files are to be written.
- *FILE_PREFIX* – The file prefix of the file target.
- *FILE_ALIAS* – The file alias for registering the files in Audit & Control.
- *MAX_COUNT* – The maximum number of records in a file.
- *DATA_GROUP* – The data group for registering the files in Audit & Control.
- *ROUTING_CRITERIA* – The routing criteria for registering the files in Audit & Control. Supports environment variables.
- *MAX_TIMEOUT* – The timeout after which a file is automatically flushed to the disk. The default value is 1 hour.

Configuration Files

The Dispatcher process requires a configuration file (a .ccf file) that is defined via an internal configuration tool.

The environment variables that appear in the .ccf file and can be customized are defined in the op_adj_env_sh script (for more information, see the “Environment Variables” section in this chapter).

In addition, real-time notification processing requires a file that contains the external record structure for the target Web service. The location and name of the file are specified in the RTN_ER_STRUCTURE configuration parameter. In this file, the order of the attributes must be the same as in the OM1_Formatting.xml file, and the names of the attributes must match the tag names that must be sent in the request. The exit point reads the structure from this file and creates a request to be sent to the target.

Example:

```
TargetID=21198
IntegerParameterValue,ExternalRecordID
StringParameterValue,NotificationID
StringParameterValue,TimeStamp
StringParameterValue,Entity_Name
IntegerParameterValue,Event_ID
DateTimeParameterValue,Network start time
DateTimeParameterValue,Start time
IntegerParameterValue,Subscriber_ID
IntegerParameterValue,Customer_ID
IntegerParameterValue,Target customer_ID
StringParameterValue,Service filter
IntegerParameterValue,Account
IntegerParameterValue,Billing arrangement
IntegerParameterValue,Pay channel
StringParameterValue,Rerate type
StringParameterValue,Guiding resource_ID
StringParameterValue,Guiding resource_type
StringParameterValue,Event type name
IntegerParameterValue,Offer_ID
IntegerParameterValue,Pricing_item_ID
StringParameterValue,Equipment_ID
StringParameterValue,Calling_country
StringParameterValue,Calling_number
StringParameterValue,Called_country
StringParameterValue,Called_number
IntegerParameterValue,Duration
IntegerParameterValue,Rounded_duration
IntegerParameterValue,Total_volume
IntegerParameterValue,Rounded_total_volume
StringParameterValue,Event_start_period
StringParameterValue,Call_forward_ind
FloatParameterValue,Charge_amount
FloatParameterValue,Charge_including_free_allowance
FloatParameterValue,Discount_amount
```

```
FloatParameterValue,Additional charge
FloatParameterValue,Free charge amount
IntegerParameterValue,Free total volume
IntegerParameterValue,Free duration
StringParameterValue,Call direction
FloatParameterValue,Original currency charge amount
FloatParameterValue,Original converted charge amount
StringParameterValue,Provider ID
StringParameterValue,Call category
StringParameterValue,Cell ID
StringParameterValue,Switch ID
IntegerParameterValue,Cycle code
IntegerParameterValue,Cycle instance
IntegerParameterValue,Cycle year
StringParameterValue,Charge code
StringParameterValue,Special number
StringParameterValue,Zone ID
StringParameterValue,Dialed digits
IntegerParameterValue,Truncated charge amount
IntegerParameterValue,Test charge amount
StringParameterValue,Pricing item name
StringParameterValue,USA zone
IntegerParameterValue,Source ID
```

Database Connections

During initialization, the Dispatcher process connects to several reference tables. For a complete description of each table, see *Turbo Charging Data Model*. After initialization, it connects to the following tables.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	AC1_CONTROL	Select, Insert
Reference	ADJ1_CUST_GROUPS_SEGMENTS	Select
Reference	GN1_SYS_SEG_PROC_CFG	Select
Reference	APR1_DISP_TARGET	Select
Usage	APR1_DISP_ELA_RECOVERY	Select, Insert, Delete
Usage	APR1_DISPATCHING_RECORDS	Select

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Usage	APR1_DISPATCHER_RECOVERY	Select, Insert, Update, Delete
Usage	APR1_DISPATCHER_CTRL	Select, Insert, Update

Admin Commands

The Dispatcher process receives the following admin commands. For information on how to execute the commands, see the “Handling Admin Commands” section in the “High Availability” chapter.

Command	Description	Parameters
SET_LIGHT_TRACER_COMMAND	Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters	<ul style="list-style-type: none"> ■ Activation parameters: <ul style="list-style-type: none"> • <code>--turnTracer "on/off"</code> – Activates or stops the Light Tracer • <code>--turnAssert "on/off"</code> – Activates or stops Assertions • <code>--turnLogOnTrace "on/off"</code> – Activates or stops writing log messages in Light Tracer files • <code>--turnFlowTracer/turnAll "on/off"</code> – Activates or stops tracing a single flow or event that is transferred between machines • <code>--turnAll "on/off"</code> – Activates or stops all of the above. The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter). ■ <code>--moduleID "ALL"</code> ■ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • <code>--threadInfo "Y/N"</code> – Specifies whether trace or assertion messages include the thread information. • <code>--timestampInd "Y/N"</code> – Specifies whether trace or assertion messages include the time stamp. • <code>--contextInd "Y/N"</code> – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • <code>--noOfMessages "<number>"</code> – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down. • <code>--period "5/10/20/60/120/480/1440"</code> – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated.

Command	Description	Parameters
		 Note: If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached. <ul style="list-style-type: none"> • <code>--assertAbortInd "Y/N"</code> – Specifies whether the application is to be aborted if the assertion condition fails. ▪ <code>--commandType "changeBufferSize"</code> – Resizes the Light Tracer buffer at run time. • <code>--bufferSize "<number>"</code> – The new size for the Light Tracer buffer.
SET_TRACE_SQL_COMMAND	<p>Activates or stops SQL tracing at runtime. The command name with its parameters must be enclosed in quotation marks.</p> <p><i>Example:</i> <code>ADJ1_Send_Admin_Command_Sh illin1027 <process instance> "SET_TRACE_SQL_COMMAND -- traceSqlOn "yes" --duration \"1\""</code></p>	<code>--traceSqlOn "<yes or no>" --duration "<minutes>"</code> If the IS_DURATION_REQUIRED environment variable is set to 'Y', the --duration parameter is mandatory. If the duration specified in the command exceeds the allowed maximum value, SQL tracing is activated for the maximum allowed period of time only (60 minutes).  Note: The SHUT_DOWN_CHECK_FREQ parameter of the Light Tracer in the ADJ1_CONF_SECTION_PARAM table determines the frequency (defined in the number of messages) at which the SQL Trace Manager checks whether the period specified in the --duration parameter has passed. Therefore, SQL tracing is stopped after the specified duration or after this number of messages has been reached, whichever comes later.
SWITCH_LIGHT_TRACER_ON ALL	Activates the Light Tracer without the ability to specify the mode or send parameters.	N/A

Command	Description	Parameters
SWITCH_LIGHT_TRACER_OFF ALL	Stops the Light Tracer	N/A
CPF_GracefulShutdownCommand	Shuts down the Dispatcher process gracefully.	N/A

Examples:

- ADJ1_Send_Admin_Command_Sh hpbl820 DISP 0777 SET_LIGHT_TRACER_COMMAND –turnTracer “on” –moduleId “ALL” –threadInfo “y”
- ADJ1_Send_Admin_Command_Sh hpbl820 DISP 0777 SWITCH_LIGHT_TRACER_OFF ALL

Distribution

The Dispatcher process distributes its results as follows:

1. Creates files with external records and registers them in Audit & Control
2. Updates the Dispatcher Control tables

Screen Messages

There are no special screen messages. Log messages are also sent to the console, but they are best viewed in the log file.

Troubleshooting

The following important or frequent errors can occur during the process run.

Error	Preventing Action
A Dispatcher instance starts running and immediately shuts down.	Search for the ERROR or FATAL string in the Dispatcher log. The first ERROR or FATAL message probably describes the problem.
The Dispatcher log prints that process could not be initialized.	There is probably another Dispatcher instance (with the same instance number) already running on the same machine. Running more than one Dispatcher instance with the same name on the same machine is not allowed. If you need to run more than one Dispatcher instance on the same machine, change the name of one of the instances.
The Dispatcher process shuts down after it has processed some external records.	Search for the ERROR or FATAL string in the Dispatcher log. The first ERROR or FATAL message probably describes the problem.

Recovery Instructions

When the Dispatcher process starts, its recovery mechanism is automatically activated. Therefore, no special actions are needed after the process is killed or has crashed. When the process is re-activated after a shutdown, regardless of whether it shut down gracefully or was stopped immediately, it resumes handling any disconnected sessions in Degraded mode.

The automatic recovery mechanism includes recovery of event-level auditing messages from the APR1_DISP_ELA_RECOVERY table. Before sending a UDP message to the ELA Collector, the Dispatcher writes event-level auditing data to this table. If an acknowledgement is received from the ELA Collector, the corresponding entry is removed. Any remaining entries in the table mean that either the Dispatcher did not send a message to the ELA Collector prior to Dispatcher failure, or the message was sent, but the Dispatcher did not receive an acknowledgement. Therefore, these records must be reprocessed to avoid losing event-level auditing counters. In the Dispatcher recovery flow, the entries from the table are reported to the ELA Collector.

If real-time notifications were not sent to the target Web service, the Dispatcher recovers the external records from the memory-mapped file and sends them to their target in parallel with regular processing.

If the external records were distributed to their targets, and the files were closed, but the commit operation failed, the files are rolled back to the end of the previous session.

If the Dispatcher fails to register a session in Audit & Control, the file is marked in the APR1_DISPATCHER_RECOVERY table as aborted ('AB') and is not recovered automatically until the operator fixes the problem.

Important Remarks

This section provides additional information on the Dispatcher process.

Defining Multiple Instances of Dispatcher Process

Multiple instances of the Dispatcher process may run simultaneously on the same machine or on different machines. Each Dispatcher instance must be defined in the routing tables (the set of tables that are used by the Availability Manager). The names of these tables begin with GN1_SYS_. These tables must contain a separate row for each instance of the Dispatcher process. The PROCESS_INSTANCE_ID (or sometimes PROCESS_EXEC_ID) field in the routing tables must contain the name of the instance, and it must be the same value as in the APPLICATION_ID parameter in the process start-up command line (for example, DISP0777).

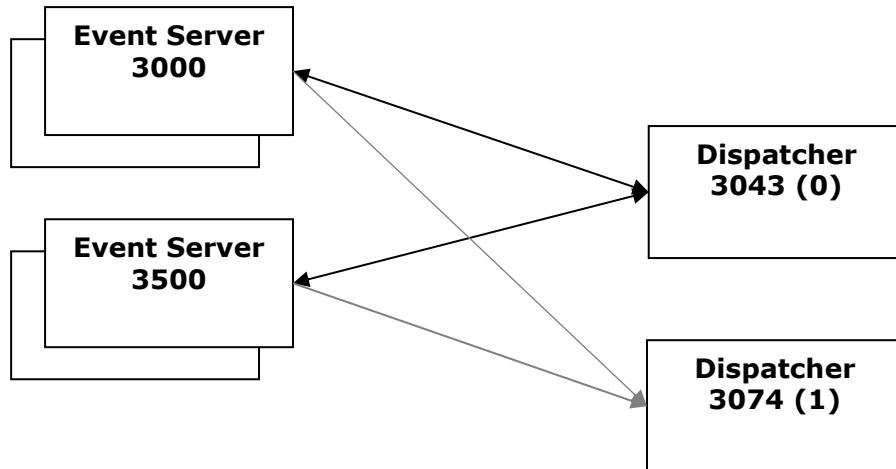
Different instances of the Dispatcher process must not run with the same cycle list (the cycle list is defined by the set of the Event Servers that the Dispatcher serves).

Configuring Relationships between Event Servers and Dispatcher Process

The Dispatcher process serves one or more Event Servers that send their external records for dispatching. This many-to-one relationship is defined in the GN1_SYS_SEG_PROC_CFG table.

Figure 21.1 describes the set-up of two Event Servers and their shadows that connect to a single Dispatcher process and its back-up.

Figure 21.1 Event Servers and Dispatcher



The following table details this set-up.

SEGMENT_GROUP_TYPE	SEGMENT_GROUP_ID	PROCESS_EXEC_ID	PROCESS_ROLE	BACKUP_ORDER
S	3000	3043	0	0
S	3000	3074	0	1
S	3500	3043	0	0
S	3500	3074	0	1

In this table:

- *SEGMENT_GROUP_TYPE* – The segment group type, where ‘S’ stands for Dispatcher
- *SEGMENT_GROUP_ID* – The process ID of the Dispatcher
- *PROCESS_EXEC_ID* – The process ID of the Event Server
- *PROCESS_ROLE* – The process role, where ‘0’ is the role assigned to the Dispatcher (for more information on process roles, see *Availability Manager High Availability Business Parameters Configuration Guide*)
- *BACKUP_ORDER* – The order of the back-ups in the segment group, where ‘0’ stands the primary Dispatcher, and ‘1’ stands for its back-up

Clean-up Job

An external daily clean-up job is responsible for cleaning the processed external records from the tables.

The configuration of the job includes a parameter that specifies how many days of history may be kept in the Dispatcher tables. The value cannot be more than 25, and the recommended number of days is 3.

The Clean-up job performs the following steps:

1. Connects to the relevant Usage database.
2. Calculates the partition of the day-of-month key that is to be truncated. The calculation is relative to the current batch date, according to the number of history days to keep.
3. Checks whether all the external records in the day-of-month partition to truncate have been processed. To do this, it compares the count of external records in the APR1_DISPATCHING_RECORDS table to the count of successful and erred external records in the APR1_DISPATCHER_CTRL table.
 - If all the external records have been processed, truncates this day-of-month partition from the following tables:
 - APR1_DISPATCHING_RECORDS
 - APR1_DISPATCHER_CTRL
 - If not all the external records have been processed, the partition is not deleted from these tables.

22 ADJ1DISPEOC – Dispatcher for End of Cycle

This chapter describes the Dispatcher for End of Cycle job.

Description

The Dispatcher for End of Cycle job initiates the dispatching procedure for cycle data. For more information, see the “[ADJ1DISPSRV – Dispatcher](#)” chapter.

Process Type

Batch job

Run Frequency

Every cycle

Participation in Generic Maps

This section contains information on the generic maps in which the process participates.

Map Name

The process participates in the following operational maps:

- End-of-Cycle (EOC) map
- Undo map

For more information about these maps, see *Amdocs Billing Operator Guide*.

Severity Level

Severity Level 1 – The map can continue to the next job without fixing this job.

Activation

Command Line:	RunJobs ADJ1DISPEOC BYREQ
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Dispatcher_EOC_Job_Sh
Executable Name:	N/A

For more information about the activation scripts, see the “[Scripts](#)” section in the “[Introduction](#)” chapter.

Shutdown

When the job is done, it shuts down on its own. If necessary, it can be shut down from the command line.

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> DISP_EOC \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

ADJ1_Send_Admin_Command_Sh hpbl820 DSP_EOC 113
CPF_GracefulShutdownCommand

Preceding Processes

In the End-of-Cycle map, the following job must run successfully before this process is activated:

- *ADJ1RRPOPREP* – Rerate Population Report

Dependent Processes

In the Undo and End-of-Cycle maps, the following job can run only after the successful completion of this process:

- *ADJ1EXTCLEANE* – Extract Clean-up

Affected Applications

The Dispatcher for End of Cycle process affects all applications (defined in the NXT_PGM_NAME field of the AC_PGM_RULES_CONTROL table) that read files from the Audit & Control table.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*

ADJ1DISPEOC_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1DISPEOC_DSP_EOC113_20081109_130927_1.log

- *Console log files:*

ADJ1_Dispatcher_EOC_Job_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1_Dispatcher_EOC_Job_inner_Sh_DSP_EOC113_20081109_130927.log

- *Operational log files:*

ADJ1DISPEOC_<APPLICATION_ID>_\${ABP_MARKET}_<Date>_<Time>.log

Example:

ADJ1DISPEOC_DSP_EOC113_M3G_20081109_130927.log

- *Event log files:*

ADJ1DSPEOC_EVENT_LOG_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1DISPEOC_EVENT_LOG_DSP_EOC113_20110522_124354_28.log

- *Latency log files:*

ADJ1DSPEOC_LATENCY_LOG_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1DSPEOC_LATENCY_LOG_DSP_EOC113_20110524_101125_0.log

Location

Log files are located in the following directories:

- *Log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG
- *Event log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/



Note: Event logs are also printed to the Light Tracer output when the Light Tracer is activated. For more information about activating the Light Tracer, see the “Admin Commands” section in this chapter.

- *Latency log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information.
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).
- *Operational log files* – The output of operational scripts.
- *Event log files* – Comma-separated technical information about each record processed by the Dispatcher for End of Cycle:
 - Whether the record was successfully processed or aborted
 - Whether the record was processed online or in the Degraded mode
 - Session ID
 - Event ID
 - Source ID
 - Target ID
- *Latency log files* – Latency information about each session processed by the Dispatcher for End of Cycle. For more information about the structure of the log, see the “Log Files” section in the “ADJ1DISPSRV – Dispatcher” chapter.

Output Files

See the “Output Files” section in the “ADJ1DISPSRV – Dispatcher” chapter.

Flow

For information on the flow of the Dispatcher for End of Cycle process, see the “Flow” section in the “ADJ1DISPSRV – Dispatcher” chapter, with the following differences:

- Dispatcher for End of Cycle handles only one cycle connection to the Usage database.
- Dispatcher for End of Cycle shuts down if the Degrade Acceptor does not find any entries to dispatch, and the APR1_OP_MAP_EVENTS table contains a notification with event name RRPEOCFNSH and the current cycle data.

Environment Variables

See the “Environment Variables” section in the “ADJ1DISPSRV – Dispatcher” chapter.

Parameters

The following parameters must be set for the process to run properly.

Parameter	Description	Default Value
APPLICATION_ID	The process instance	DSP_EOC113
CCF_FILE_NAME	An XML file that contains the relevant configuration	APR_Dispatcher.ccf
CYCLE_CODE	The cycle code on which Dispatcher for End of Cycle works	1
CYCLE_INSTANCE	The cycle instance on which Dispatcher for End of Cycle works	1
CYCLE_YEAR	The cycle year on which Dispatcher for End of Cycle works	2009

Configuration Parameters

The configuration parameters (properties) of Dispatcher for End of Cycle are defined in the database.

The Dispatcher for End of Cycle job uses the configuration parameters described in the “Configuration Parameters” section in the “ADJ1DISPSRV – Dispatcher” chapter. In addition, it uses the following parameters.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
DSP_EOC	config	CYCLE_CODE	\${EXTR_CYCLE_CODE}	The cycle code on which the Dispatcher for End of Cycle must work
DSP_EOC	config	CYCLE_INSTANCE	\${EXTR_CYCLE_INST}	The cycle instance on which the Dispatcher for End of Cycle must work
DSP_EOC	config	CYCLE_YEAR	\${EXTR_CYCLE_YEAR}	The cycle year on which the Dispatcher for End of Cycle must work

For each process instance, an entry with the name of the process instance must be created in the APR1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
DSP_EOC113	DSP_EOC

Any parameter to be overridden at the process instance level must be added to the APR1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
DSP_EOC113	config	CYCLE_CODE	\${EXTR_CYCLE_CODE}

For more information on parameters and the properties mechanism, see the “Configuration Parameter Mechanism” appendix.

The Dispatcher for End of Cycle process reads target configuration parameters from the APR1_DISP_TARGET using the Reference Table Synchronizer. These parameters are:

- *FILE_PATH* – The relative path to which files are to be written.
- *FILE_PREFIX* – The file prefix of the file target.
- *FILE_ALIAS* – The file alias for registering the files in Audit & Control.
- *MAX_COUNT* – The maximum number of records in a file.
- *DATA_GROUP* – The data group for registering the files in Audit & Control.
- *ROUTING_CRITERIA* – The routing criteria for registering the files in Audit & Control. Supports environment variables.
- *MAX_TIMEOUT* – The timeout after which a file is automatically flushed to the disk.

Configuration Files

The Dispatcher for End of Cycle process requires a configuration file (a .ccf file) that is defined via an internal configuration tool.

The environment variables that appear in the .ccf file and can be customized are defined in the op_adj_env_sh script (for more information, see the “Environment Variables” section in this chapter).

Database Connections

During initialization, the Dispatcher for End of Cycle process connects to several reference tables. For a complete description of each table, see *Turbo Charging Data Model*. After initialization, it connects to the following tables.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	APR1_OP_MAP_EVENTS	Select
Reference	ADJ1_CUST_GROUPS_SEGMENTS	Select
Reference	GN1_SYS_SEG_PROC_CFG	Select
Reference	APR1_DISP_TARGET	Select
Usage	APR1_DISPATCHING_RECORDS	Select
Usage	APR1_DISPATCHER_RECOVERY	Select, Insert, Update, Delete
Usage	APR1_DISPATCHER_CTRL	Select, Insert, Update

Admin Commands

The Dispatcher for End of Cycle process receives the following admin command. For information on how to execute the commands, see the “Handling Admin Commands” section in the “High Availability” chapter.

Command	Description	Parameters
CPF_GracefulShutdownCommand	Shuts down the Dispatcher process gracefully	N/A

For an example of this command, see the “Shutdown” section in this chapter.

Distribution

See the “Distribution” section in the “ADJ1DISPSRV – Dispatcher” chapter.

Screen Messages

There are no special screen messages. Log messages are also sent to the console, but they are best viewed in the log file.

Troubleshooting

See the “Troubleshooting” section in the “ADJ1DISPSRV – Dispatcher” chapter.

Recovery Instructions

See the “Recovery Instructions” section in the “ADJ1DISPSRV – Dispatcher” chapter.

Important Remarks

Following is a summary of the main differences between the Dispatcher for End of Cycle (ADJ1DISPEOC) and Dispatcher (ADJ1DISPSRV) processes:

- Dispatcher for End of Cycle is a batch job, which shuts down if it has no more tasks and it has received a notification that the Rerate Prepare for End of Cycle process is finished.
- Dispatcher for End of Cycle gets its cycle connection to the Usage database according to the cycle that participates in the End-of-Cycle or Undo map.
- In addition to the connections that the Dispatcher uses, the Dispatcher for End of Cycle job uses a connection to the applicative database.

23 ADJ1ELASRV – ELA Collector

This chapter describes the ELA Collector process.

Description

Event-Level Auditing and Service Level Management provide data about the events processed in the system. The ELA Collector daemon is the data collection component of Event-Level Auditing and Service Level Management. It does the following:

- Receives and collects event-level auditing data from the core and custom reporting points.
- Performs the necessary computations and comparisons needed for auditing.
- Aggregates the data that it receives per token and file (for offline events) – a key that expresses a time frame.
- Persists the collected and aggregated data into Turbo Charging and Audit & Control tables.
- If the database fails, persists the collected data into files. When the database is up again, the ELA Collector loads the data from the files into the database and continues as usual.

Process Type

Deamon

Run Frequency

Runs continuously

Activation

Command line:	ADJ1_ELA_Daemon_Shell_Sh -n <APPLICATION_ID> -c "-f <CCF_FILE_NAME>" -e <EWD_EXE>  Note: The -e parameter is optional. It is used in environments with Availability Manager.
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_ELA_Daemon_Shell_Sh
Executable Name:	gcpf1fwcApp

Example:

ADJ1_ELA_Daemon_Shell_Sh -n ELA104 -c "-f APR1_ELA.ccf" -e gn1avm_ewd

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> ELA \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes

Example:

ADJ1_Send_Admin_Command_Sh hpbl820 ELA 104
CPF_GracefulShutdownCommand

Dependent Processes

If in the configuration of the system, Event-Level Auditing is enabled, the ELA Collector should be activated before the reporting processes. This is not obligatory; however, if the ELA Collector is not activated before these processes, they cannot send event-level auditing data to the ELA Collector. In this case, the reporting processes write errors to the log file.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*

ADJ1ELASRV_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1ELASRV_ELA104_20081109_130927_1.log

- *Console log files:*

ADJ1_ELA_Daemon_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1_ELA_Daemon_inner_Sh_ELA104_20100427_121400.log

- *Operational log files:*

ADJ1ELASRV_<APPLICATION_ID>_\${ABP_MARKET}_<Date>_<Time>.log

Example:

ADJ1ELASRV_ELA104_M3G_20081109_130927.log

- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter)

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information.
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).
- *Operational log files* – The output of operational scripts.
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Flow

When it receives a message from an Event Server, File2E, or Amdocs Service Platform (in an integrated environment), the ELA Collector process executes the following steps:

1. *Audit Gateway module (AuditGW)* – Receives event-level auditing data from the reporting processes (through multicast broadcasting), checks for duplicated messages against the message IDs loaded at start-up from the ADJ1_AUDIT_MESSAGE_DUPKEY table according to the KEEP_HISTORY_TIMEOUT_MIN parameter and against incoming message IDs, and sends a message to the ELA Token Completion module.
2. *ELA Token Completion module* – Stores the message for token completion tracking and sends the message to the Collector module.

3. *Collector module* – Collects event-level auditing data, aggregates message data into tokens, and sends the message to the AuditPW module.
4. *Audit Persistence Writer (AuditPW) module*
 - As part of normal operations, writes the message to the ADJ1_AUDIT_MESSAGES table in the database.
 - If the database fails, writes the data to files. When the connection to the database is re-established, AuditPW transfers the data from the files back to the database.

The following flow is executed periodically when the ELA Collector detects token completion:

1. *Audit Persistence Writer (AuditPW) module* – Periodically commits data to the ADJ1_AUDIT_MESSAGES table and sends a message to the AuditGW module. In addition, it writes the message IDs into the ADJ1_AUDIT_MESSAGE_DUPKEY table to be later used in duplicate checks.
2. *Audit Gateway module (AuditGW)* – Sends an acknowledgement to the reporting processes and cleans message history.
3. *ELA Token Completion module* – Periodically checks for token completion. When token completion is detected, sends a message to the ELA Collector module.
4. *Collector* – Calculates missing events per process ID, removes data associated with the token, and sends a message to the AuditPW module.
5. *Audit Persistence Writer (AuditPW) module*
 - As part of normal operations, writes the token data into the ADJ1_AUDIT_COUNTERS and Audit & Control tables in the database.
 - If the database fails, writes the data to files. The name and location of these files are specified using the BASE_FILE_NAME and FILE_DIRECTORY configuration parameters (see the “Configuration Parameters” section in this chapter). Only the ELA Collector can read these files. When the connection to the database is re-established, AuditPW transfers the data from the files back to the database.

Environment Variables

The following environment variables affect the process.

Name	Description	Default Value	Method of Changing the Variable
APR_ELA_PW_IWD_TASK_HANGUP	The time period for which the Audit PW task threads can run without sending a heartbeat to the Internal Watchdog. If the task threads do not send a heartbeat to the Internal Watchdog within this time period, the task is considered blocked.	300	N/A
APR_ELA_PW_IWD_TASK_WEIGHT	The weight of the Audit PW task. If $(<\text{Number of blocked threads}> / <\text{Total number of threads}>) * \text{Weight} \geq 100$, the Availability Manager is notified. For example, if a task has four threads, two of which are blocked, the Availability Manager is notified only if the weight is configured as 200 or if all the threads become blocked. The default value of 100 means that only when all the threads are blocked, the Availability Manager is notified.	100	N/A
ELA_COLLECTOR_NUM_OF_THREADS	The number of threads for the ELA Collector task	4	Editing the .w.ini.pre.env file
ELA_PW_NUM_OF_THREADS	The number of threads for the Persistence Writer task	1	Editing the .w.ini.pre.env file

Parameters

The following parameters must be set for the process to run properly.

Parameter	Description	Default Value
APPLICATION_ID	The process instance	ELA104
CCF_FILE_NAME	An XML file that contains the relevant configuration	APR1_ELA.ccf

Configuration Parameters

The configuration parameters of the ELA Collector are defined in the APR1_CONF_SECTION_PARAM table and can be modified using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*).

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	ELA	GENERIC_REPORTING_POINTS	N/A	<p>The list of generic, custom reporting points (outside of Turbo Charging) that the ELA Collector must handle. Out of the box, this parameter has no value. The range of reporting point IDs for custom reporting points is 20–99.</p> <p>Further configuration of such reporting points is required. In the APR1_ELA_META_DATA table, a metadata entry must be inserted for each custom reporting point. It must indicate which fields are used in the messages from the reporting point so that the structure of these messages is available to the ELA Collector.</p>
APR	ELA	ENABLE_ELA_RECOVERY	N	Indicates whether ELA Collector recovery at start-up is enabled.
APR	ELA	NUMBER_OF_ELA_COLLECTORS	1	The number of ELA Collector daemons configured in the system. The multicast addresses of ELA Collectors must be sequential, and they must be reserved in advance.
APR	ELA	RESEND_TIMEOUT_EXTENSION_SEC	10	Extra safety time that is added on top of the specified timeout.
APR	ELA	TIME_TO_COMMIT_SEC	5	The time interval at which data is to be committed to the database.
APR	ELA	TOKEN_DURATION_MIN	5	The duration of a token (in which event auditing data is aggregated) in minutes.
APR	ELA_SERVER	AC_OUTPUT_ENABLE	N	Enables or disables writing the output into the Audit & Control tables.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	ELA_SERVER	BASE_FILE_NAME	ELA_DATA	The base string used as part of the file name when the ELA Collector works in file mode (after a database failure).
APR	ELA_SERVER	DL_BUFFER_SIZE	100	The size of the buffer (in the number of records) used in the data layer.
APR	ELA_SERVER	FILE_DIRECTORY	\${ABP_APP_ROOT}/work/ELA/	The location of the files that are created when the ELA Collector works in file mode (after a database failure).
APR	ELA_SERVER	KEEP_HISTORY_TIMEOUT_MIN	30	The period of time for which history is kept (for checking message duplication).
APR	ELA_SERVER	MONEY_PRECISION	0	The precision of the money columns in the AC1_CONTROL table.
APR	ELA_SERVER	SEND_ACK_TIMEOUT_SEC	10	The time for which the ELA Collector stores an acknowledgement before sending it together with other acknowledgements in one message.
APR	ELA_SERVER	TIME_PRECISION	0	The precision of the time columns in the AC1_CONTROL table.
APR	ELA_SERVER	TOKEN_EXPIRATION_WINDOW_MIN	5	The maximum time (in minutes) for which the ELA Collector must wait for token competition until the token expires. This is also the frequency at which token start time is generated for external token IDs.
APR	ELA_SERVER	TOKEN_HISTORY_EXPIRATION_INTERVAL_SEC	1800	The expiration time (in seconds) of a token in token history.
APR	ELA_SERVER	TOKEN_HISTORY_PURGE_PERIOD_SEC	1800	The interval (in seconds) at which the token history is checked for expired tokens.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APR	ELA_SERVER	VOLUME_PRECISION	0	The precision of the volume columns in the AC1_CONTROL table.
ELA SID (for example, ELA104)	ELA_SERVER	ELA_COLLECTOR_INDEX	0	The index of the ELA Collector. Every ELA Collector in the system must be assigned an index from 0 to NUMBER_OF_ELA_COLLECTORS - 1. The indices must be consecutive. The index of the master ELA Collector and the back-up ELA Collector must have the same index.

In addition, the process uses Light Tracer configuration parameters, which are defined in the ADJ1_CONF_SECTION_PARAM table. For more information, see [Light Tracer Parameters](#) in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

For each process instance, an entry with the name of the process instance must be created in the APR1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
ELA104	ELA

Any parameter to be overridden at the process instance level must be added to the APR1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
ELA104	ELA_SERVER	TOKEN_DURATION_MIN	6

For more information on parameters and the properties mechanism, see the “Configuration Parameter Mechanism” appendix.

Configuration Files

The ELA Collector requires a configuration file (a .ccf file) that is defined via an internal configuration tool.

Database Connections

The process connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	AC1_AUDIT_BALANCE	Insert, Update
Application	AC1_CONTROL	Insert, Update, Select
Application	ADJ1_AUDIT_COUNTERS	Insert, Select
Application	ADJ1_AUDIT_MESSAGE_DUPKEY	Insert, Select, Truncate
Application	ADJ1_AUDIT_MESSAGES	Insert, Select

Admin Commands

The ELA Collector supports the following admin commands. For information on how to execute the commands, see the “Handling Admin Commands” section in the “High Availability” chapter.

Command	Description	Parameters
SET_LIGHT_TRACER_COMMAND	Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters	<ul style="list-style-type: none"> ▪ Activation parameters: <ul style="list-style-type: none"> • <i>turnTracer “on/off”</i> – Activates or stops the Light Tracer • <i>--turnAssert “on/off”</i> – Activates or stops Assertions • <i>--turnLogOnTrace “on/off”</i> – Activates or stops writing log messages in Light Tracer files • <i>--turnFlowTracer/turnAll “on/off”</i> – Activates or stops tracing a single flow or event that is transferred between machines • <i>--turnAll “on/off”</i> – Activates or stops all of the above. ▪ The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter). ▪ <i>--moduleID “ALL”</i> ▪ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • <i>--threadInfo “Y/N”</i> – Specifies whether trace or assertion messages include the thread information. • <i>--timestampInd “Y/N”</i> – Specifies whether trace or assertion messages include the time stamp. • <i>--contextInd “Y/N”</i> – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • <i>--noOfMessages “<number>”</i> – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down. • <i>--period “5/10/20/60/120/480/1440”</i> – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated. <p> Note: If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached.</p>

Command	Description	Parameters
		<ul style="list-style-type: none"> • <code>--assertAbortInd "Y/N"</code> – Specifies whether the application is to be aborted if the assertion condition fails. ▪ <code>--commandType "changeBufferSize"</code> – Resizes the Light Tracer buffer at run time. • <code>--bufferSize "<number>"</code> – The new size for the Light Tracer buffer.
SET_TRACE_SQL_COMMAND	<p>Activates or stops SQL tracing at runtime. The command name with its parameters must be enclosed in quotation marks.</p> <p><i>Example:</i></p> <pre>ADJ1_Send_Admin_Command _Sh illin1027 <process instance> "SET_TRACE_SQL_COMMAND --traceSqlOn \"yes\" --duration \"1\""</pre>	<code>--traceSqlOn "<yes or no>" --duration "<minutes>"</code> If the IS_DURATION_REQUIRED environment variable is set to 'Y', the --duration parameter is mandatory. If the duration specified in the command exceeds the allowed maximum value, SQL tracing is activated for the maximum allowed period of time only (60 minutes).  <p>Note: The SHUT_DOWN_CHECK_FREQ parameter of the Light Tracer in the ADJ1_CONF_SECTION_PARAM table determines the frequency (defined in the number of messages) at which the SQL Trace Manager checks whether the period specified in the --duration parameter has passed. Therefore, SQL tracing is stopped after the specified duration or after this number of messages has been reached, whichever comes later.</p>
SWITCH_LIGHT_TRACER_ON ALL	Activates the Light Tracer without the ability to specify the mode or send parameters	N/A
SWITCH_LIGHT_TRACER_OFF ALL	Stops the Light Tracer	N/A
CPF_GracefulShutdownCommand	Shuts down the ELA Collector gracefully	N/A

Examples:

- `ADJ1_Send_Admin_Command_Sh hpbl820 ELA 104 SET_LIGHT_TRACER_COMMAND --turnTracer "on" --moduleId "ALL" --threadInfo "y"`
- `ADJ1_Send_Admin_Command_Sh hpbl820 ELA 104 SWITCH_LIGHT_TRACER_OFF ALL`

Recovery Instructions

Rerun the process. Recovery is performed from the data in the ADJ1_AUDIT_MESSAGES table.

24 ADJ1RORC – Read-Only Rater Client

This chapter describes the Read-Only Rater Client process.

Description

The Read-Only Rater Client process simulates the rating of production events without persisting the results into the Usage database. It extracts events of several input types according to a request table and sends each event for guiding to the corresponding Guiding to Customer Module and then for rating to a dedicated read-only Event Server. The accumulators for each rated event are maintained in memory but not written into the database.



Note: When the process is down, the memory must be deleted.

Each request has a batch ID. All requests within a batch are sent and rated without deleting the shared memory from one request to the next.

Process Type

Daemon

Run Frequency

Runs continuously

Activation

Command Line:	ADJ1_RORC_Daemon_Shell_Sh -n <APPLICATION_ID> -c "-f <CCF_FILE_NAME>" -e <EWD_EXE>  Note: The -e parameter is optional. It is used in environments with Availability Manager.
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_RORC_Daemon_Shell_Sh
Executable Name:	gcpf1fwcApp

Example:

```
ADJ1_RORC_Daemon_Shell_Sh -n RORC1900 -c "-f APR_RORC.ccf" -e  
gn1avm_ewd
```

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

To trigger read-only rating for a request, do the following:

1. Insert the request into the APR1_ROR_REQ table:

- Set the INPUT_TYPE field as follows:
 - R* – Rated event
 - J* – Rejected event
 - F* – File



Note: When the entire subscriber is required, set the INPUT_TYPE field to 'R', but leave the EVENT_ID field empty.

Example:

```
Insert into APR1_ROR_REQ
(BATCH_ID, SEQ_IN_BATCH, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, INPUT_TYPE, PROTOCOL_ID,
CYCLE_CODE, CYCLE_INSTANCE, CUSTOMER_SEGMENT, CUSTOMER_ID, SUBSCRIBER_ID,
EVENT_ID, START_TIME, FILE_NAME, FILE_PATH, STATUS, FAILURE_DESC)
Values
(0, 4, TO_DATE('02/10/2010 18:21:48', 'MM/DD/YYYY HH24:MI:SS'), NULL, NULL,
NULL, NULL, 0, 'R', 13,100, 8, 812, 2350, 1016,1.29516064085E17,
TO_DATE('10/11/2008 15:26:35', 'MM/DD/YYYY HH24:MI:SS'), NULL, NULL, NULL, NULL);
```

- For rated and rejected events, set the following fields to the values in the corresponding fields in the APE1_RATED_EVENT or APE1_REJECTED_EVENT table, respectively:

- CYCLE_INSTANCE
- CYCLE_CODE
- CUSTOMER_SEGMENT
- EVENT_ID
- CUSTOMER_ID
- SUBSCRIBER_ID
- START_TIME

Example:

```
update APR1_ROR_REQ
set CYCLE_INSTANCE =(select CYCLE_INSTANCE from APE1_RATED_EVENT where CYCLE_CODE = 1)
where BATCH_ID = 2;
```

- Leave the STATUS field empty.

2. Run the read-only Event Server.

Examples:

- For a file request

ADJ1_ES_Daemon_Shell_Sh -n ES100 -c " -f complete_ES_TCP_20.ccf"

- For a rated or rejected event request

ADJ1_ES_Daemon_Shell_Sh -n ROR_SERVER156 -c " -f
complete_ES_TCP_20.ccf"

3. Run the Read-Only Rater Client.

Example:

ADJ1_RORC_Daemon_Shell_Sh -n RORC380 -c "-f APR_RORC.ccf"

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> RORC \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

ADJ1_Send_Admin_Command_Sh hpbl820 RORC 1900
CPF_GracefulShutdownCommand

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*

ADJ1RORC_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1RORC_RORC1900_20081109_130927_1.log

- *Console log files:*

ADJ1_RORC_Daemon_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1_RORC_Daemon_inner_Sh_RORC1900_20081109_130927.log

- *Operational log files:*

ADJ1RORC_<APPLICATION_ID>_\${ABP_MARKET}<Date>_<Time>.log

Example:

ADJ1RORC_RORC1900_M3G_20081109_130927.log

- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_LOG
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter)

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information.
- *Operational log files* – The output of operational scripts.
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Input Files

The process takes input files according to the APR1_ROR_REQ table. The table specifies the names and location of these files.

Flow

This section describes the process flow of the Read-Only Rater Client process.

Process Start-up

At start-up, the Read-Only Rater Client process loads the libraries generated by the Implementation Compiler.

Main Loop

Each thread in the process has its own main loop, and all threads work in parallel.

This section describes the main tasks that each thread type performs as part of the main loop.

Batch Work Producer

The Batch Work Producer is a single thread that performs the following:

1. Fetches the next batch of requests from the APR1_ROR_REQ table. A request can be in the form of a file, a specific event, or a subscriber ID.
2. Forwards each request in the batch to the Batch Work Unit Processor.
3. Having forwarded all the requests in the current batch to the Batch Work Unit Processor, waits until every request gets a response from the server. See “[Batch Work Unit Processor](#).”
4. For each result that is received for a request, updates the corresponding record in the database.
5. When all the requests in the queue have been handled or timed out, sends admin requests to the Event Processing Modules that participated in read-only rating to clean their rater groups.
6. When all the batches in the APR1_ROR_REQ have been handled, initiates the shutdown of the process.

Batch Work Unit Processor

The Batch Work Unit Processor comprises multiple threads, where each thread performs the following:

1. Retrieves a request from its queue.
2. Extracts each request (a work unit) into a group of events.
3. Forwards the event to the Event Server performing the Guiding to Customer function.
For more information on how responses from the Guiding to Customer Module are handled, see “[Guiding to Customer \(FR\) Response Handler](#).”

Guiding to Customer (FR) Response Handler

Responses from the Guiding to Customer Module are forwarded to the Guiding to Customer Response Handler, which is also known as FR Response Handler.

The FR Response Handler comprises multiple threads, where each thread performs the following:

1. Updates the status of each response from the Guiding to Customer Module in the inner container.
2. For each successfully guided event, checks whether the dedicated Event Processing Module holds a shared memory group for this cycle:
 - If so, sends the event to this Event Server to the Event Processing (RB) Router (see “Event Processing Router (Router RB)”), which forwards the event to the dedicated Event Processing Module.
 - If not, performs the following:
 - i. Sends a message to the dedicated Event Processing Module to add a group.
 - ii. After receiving a response that the group has been added successfully, sends the event to this Event Server.

Event Processing (RB) Response Handler

Responses from the Event Processing Module are forwarded to the Event Processing Response Handler, which is also known as the RB Response Handler.

The RB Response Handler comprises multiple threads, where each thread updates the status of each response from the Event Processing Module in the inner container.

Guiding to Customer Router (Router FR)

The Guiding to Customer Router thread, which is also known as Router FR, sends messages to the selected Event Server that performing the Guiding to Customer function.

Event Processing Router (Router RB)

The Event Processing Router thread, which is also known as Router RB, sends messages to the dedicated Event Server that functions as the Event Processing Module.

Batch Work Unit Watcher

The Batch Work Unit Watcher comprises multiple threads, where each thread performs the following:

1. Goes over each current work unit and signs it with the appropriate status:
 - If a work unit contains all the responses for all events, signs it with a success status
 - If a work unit does not contain all the responses and its expiration time has passed:
 - i. Checks the percentage of successful events
 - ii. Signs the work unit with the status of failure or success with errors.
2. Closes the work units and sends them back to the Batch Work Producer.

Process Shutdown

The shutdown of the Read-Only Rater Client depends on the requests in the APR1_ROR_REQ table. When the table does not contain any unhandled requests, the daemon can be configured via the parameters in the APR1_CONF_SECTION_PARAM table to either shut down or continue and wait for new requests.

For more information, see the “[Configuration Parameters](#)” section in this chapter.

Environment Variables

The Read-Only Rater Client uses the following environment variables.

Variable Name	Description	Valid Values/ Default Value
APR_RORC_WAITTIME_MILLISECOND_FR	The maximum amount of time (in milliseconds) to wait for an event to be processed by the Guiding to Customer Module.	100000
APR_RORC_WAITTIME_MILLISECOND_RB	The maximum amount of time (in milliseconds) to wait for an event to be processed by the Event Processing Module.	200000

Parameters

The following parameters must be set for the Read-Only Rater Client process to run properly.

Parameter	Description	Default Value
APPLICATION_ID	The process instance	RORC
CCF_FILE_NAME	An XML file that contains the relevant configuration	APR_RORC.ccf

Configuration Parameters

The configuration parameters of the Read-Only Rater Client are defined in the APR1_CONF_SECTION_PARAM table and can be modified using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*).

PARAM_C_LASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
RORC	config	AVM_ENABLED	N	Indicates whether Availability Manager is used when running the process.
RORC	config	BATCH_WAIT_TIME_SEC	10	The maximum amount of time (in seconds) for which the Batch Work Producer must wait until a batch of requests is done before timing out

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
RORC	config	DD_LIB_NAME	Mpr_dd	The name of the library containing the data dictionary of the Implementation Compiler
RORC	config	FORCE_GUIDING	No	Indicates whether a guided event must be reguided.
RORC	config	MEM_ALLOC	10	The maximum number of events in a work unit
RORC	config	PERCENTAGE_EVENTS_FOR_UNIT_SUCCESS	100	The minimum percentage of events with responses in a work unit that qualifies the unit for a status of success
RORC	config	RB_CONNECT_PERCENTAGE_BEFORE_START_WORK	100	The percentage of Event Servers functioning as Event Processing Modules of all configured Event Servers functioning as Event Processing Modules that must be connected to the Read-Only Rater Client before it can start to work on the batches
RORC	config	RB_CONNECT_PERCENTAGE_BEFORE_START_WORK	0	The percentage of Event Servers functioning as Guiding to Customer Modules of all configured Event Servers functioning as Guiding to Customer Modules that must be connected to the Read-Only Rater Client before it can start to work on the batches
RORC	config	SELECT_BUFFER_SIZE	1000	The number of records to be fetched in each loop from either the Rated Events or Rejected Events table
RORC	config	SHUTDOWN_ON_END_WORK	Y	Indicates whether the Read-Only Rater Client must shut down when the APR1_ROR_REQ table no longer contains unhandled requests or continue to wait for new requests
RORC	config	WAITTIME_MILLISECOND_FR	\${APR_RORC_WAITTIME_MILLISECOND_RB}	The maximum amount of time (in milliseconds) to wait for an event to be processed by the Guiding to Customer Module
RORC	config	WAITTIME_MILLISECOND_RB	\${APR_RORC_WAITTIME_MILLISECOND_RB}	The maximum amount of time (in milliseconds) to wait for an event to be processed by the Event Processing Module

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
RORC	config	WU_ALLOCATION_BUFFER	100	The initial number of work units that are allocated to the Read-Only Rater Client
RORC	config	WU_MESSAGES_THRESHOLD	10	The maximum number of events that can have open transactions
RORC	ELA	ENABLE	N	Indicates whether Event-Level Auditing is enabled for the Read-Only Rater Client
RORC	OMC_SDK	OM_ENABLED	N	Indicates whether the process generates operational measurement data

In addition, the process uses Light Tracer configuration parameters, which are defined in the ADJ1_CONF_SECTION_PARAM table. For more information, see [Light Tracer Parameters](#) in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

For each process instance, an entry with the name of the process instance must be created in the APR1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
RORC1900	RORC

Any parameter to be overridden at the process instance level must be added to the APR1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
RORC1900	config	FORCE_GUIDING	Y

For more information on parameters and the properties mechanism, see the “Configuration Parameter Mechanism” appendix.

Configuration Files

The Read-Only Rater Client requires a configuration file (.ccf) that is defined via an internal configuration tool.

The following variables can be customized in the .ccf file:

- In <task_properties> CR Incoming <property name="Task Name" value="CR Incoming" />, guiding to customer can be defined as local or remote (<property name="FR is local" value="false" />). The default value is remote (false).

- The number of threads of each inner module of the Read-Only Rater Client can be changed in the <property name="Threads" value="1" /> line for the following Task Names:
 - BatchWorkUnitWatcher
 - BatchWorkUnitProcessor
 - RouterRB
 - ResponseFR
 - CR Incoming
 - CR Outgoing
 - RouterFR
 - ResponseRB

The Batch Work Producer is designed to work as a single thread only. Therefore, its Threads property must not be changed.

Database Connections

During initialization, the process connects to many reference tables. For a complete description of each table, see *Turbo Charging Data Model*.

After initialization, the Read-Only Rater Client connects primarily to the following tables.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	APR1_ROR_REQ	Select, Update
Usage	APE1_RATED_EVENTS	Select
Usage	APE1_REJECTED_EVENTS	Select

Admin Commands

The Read-Only Rater Client supports the following admin commands. For information on how to execute the commands, see the “Handling Admin Commands” section in the “High Availability” chapter.

Command	Description	Parameters
SET_LIGHT_TRACER_COMMAND	Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters	<ul style="list-style-type: none"> ■ Activation parameters: <ul style="list-style-type: none"> • <i>turnTracer “on/off”</i> – Activates or stops the Light Tracer • <i>--turnAssert “on/off”</i> – Activates or stops Assertions • <i>--turnLogOnTrace “on/off”</i> – Activates or stops writing log messages in Light Tracer files • <i>--turnFlowTracer/turnAll “on/off”</i> – Activates or stops tracing a single flow or event that is transferred between machines • <i>--turnAll “on/off”</i> – Activates or stops all of the above. The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter). ■ <i>--moduleID “ALL”</i> ■ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • <i>--threadInfo “Y/N”</i> – Specifies whether trace or assertion messages include the thread information. • <i>--timestampInd “Y/N”</i> – Specifies whether trace or assertion messages include the time stamp. • <i>--contextInd “Y/N”</i> – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • <i>--noOfMessages “<number>”</i> – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down. • <i>--period “5/10/20/60/120/480/1440”</i> – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated.

Command	Description	Parameters
		 <p>Note: If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached.</p> <ul style="list-style-type: none"> • <code>--assertAbortInd "Y/N"</code> – Specifies whether the application is to be aborted if the assertion condition fails. ▪ <code>--commandType "changeBufferSize"</code> – Resizes the Light Tracer buffer at run time. <ul style="list-style-type: none"> • <code>--bufferSize "<number>"</code> – The new size for the Light Tracer buffer.
SWITCH_LIGHT_TRACER_ON ALL	Activates the Light Tracer without the ability to specify the mode or send parameters.	N/A
SWITCH_LIGHT_TRACER_OFF ALL	Stops the Light Tracer	N/A
CPF_GracefulShutdownCommand	Shuts down the Read-Only Rater Client gracefully	N/A

Examples:

- `ADJ1_Send_Admin_Command_Sh hpbl820 RORC 1900 SET_LIGHT_TRACER_COMMAND --turnTracer "on" --moduleId "ALL" --threadInfo "y"`
- `ADJ1_Send_Admin_Command_Sh hpbl820 RORC 1900 SWITCH_LIGHT_TRACER_OFF ALL`

Recovery Instructions

If a Read-Only Rater Client crushes, the next Read-Only Rater Client that is raised starts from the first batch that was not completed (meaning that it contains work units without a status) and handles it from the beginning.

Important Remarks

The Read-Only Rater Client sends events for guiding to any Event Server that functions as a Guiding to Customer Module. However, for processing and rating, it sends the events to a dedicated Read-Only Rater server, which is the only Event Processing Module in the system that rates events in read-only mode.

25 ADJ1UQPROXY – Usage Query Proxy

This chapter describes the Usage Query Proxy process.

Description

The Usage Query Proxy connects to the front end and the Usage Query Server. This process retains the synchronicity of the calling OLTP process, transforms it to an asynchronous call, and passes it to the Usage Query Server. After processing, it receives a response with the requested data and transfers it to the client system.

Process Type

Daemon

Run Frequency

Runs continuously

Activation

Make sure that the ADJUQ building block is deployed in the environment. To activate the process, start the Application server.

Shutdown

To shut down the process, stop the Application server.

Preceding Processes

The following process must run successfully before this process is activated:

- *Implementation Compiler* – The Implementation Compiler creates an SQL file called *attributes-id-map.sql*, which contains the data of the ADJ1_ATTRIBUTES_ID_MAP table. The table contains the mapping between the generated attribute IDs and the attribute names. The Usage Query Proxy receives IDs from the Usage Query Server and returns attribute names to the client.

Affected Applications

The Usage Query Proxy affects the following application:

- *ADJ1UQSRV* – Usage Query Server

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

ADJ1UQPROXY.<Date>_<Time>.log

Example:

ADJ1UQPROXY.20120213_1121.log

Location

The files are located in \$(APP_DOMAIN_HOME)/\$(APP_SERVER_NAME)/logs.

Contents

The process log file contains the standard output and errors from all the threads. In addition, it contains latency information.

Flow

This section describes the flow of the Usage Query Proxy process.

Initialization

At start-up, if the Availability Manager map was changed while the Usage Query Proxy was down, the Usage Query Proxy gets an updated map.

If a cycle has been moved to an alternate database:

- If the local replication mode is disabled or if the request is an update request, the Usage Query Proxy does not send the request to the Usage Query Server that is responsible for that cycle.
- If the local replication mode is enabled and if the request is a read request, the Usage Query Proxy sends the request to the Usage Query Server that is responsible for that cycle, and the Usage Query Server starts working against the replication database.
The replication database is local to the site.

If Geo Redundancy is enabled, the Usage Query Proxy is active both at the active site and at the standby site.

- At the active site, the Usage Query Proxy processes update requests and sends read requests to the Usage Query Server.
- At the standby site, it does not process update requests and sends only read requests to the Usage Query Server.

Usage Query Request Flow

When the user (for example, a CSR) requests usage information for a customer, the request is sent to the application server, which activates the relevant usage query bean. The bean requests an IUQConnection from the connection factory, and sends the request

through the connection to the Usage Query Connector. Then, usage query requests follow this flow in the Usage Query Connector:

1. The request reaches the Request Manager, which performs the following:
 - a. Checks whether the cycle has been moved to the alternate database. If so:
 - If the request is an update request, does not process it.
 - If the request is a read request, processes it only if the local replication mode is enabled.
 - b. Stamps the request with the request ID
 - c. Adds it to the Ready for Work queue

The request sleeps in the Request Manager until a response is returned, or timeout occurs.
2. The sleeping Request Worker, which waits on the work queue, awakes, takes the request, and processes it as follows:
 - Parses the request (creates a byte buffer according to the protocol).
 - Asks the Connection Manager for a connection to the Usage Query Server. The Connection Manager performs the routing function:
 - If there is more than one server according to the routing function, the connection to the least loaded server is returned.
 - If all connections to that server are busy, the request waits in that server's connection queue until the connection is available.
 - Writes the request on the connection.
 - Returns the connection to the Connection Manager.
 - Takes a new request waiting in the queue.

Usage Query Response Flow

Usage query responses follow this flow:

1. The Response Manager and the Response Handlers are initiated on start-up. Each connection to the Usage Query Server has a dedicated Response Handler that waits for responses.
2. The Response Handler reads the response message from the Usage Query Server and puts it in the Response Manager's Ready for Work queue.
3. The Response Manager creates the task for the Response Worker and adds it to the queue for the Response Worker threads.
4. The sleeping Response Worker, which waits on the queue, awakes, takes the response, and parses it (converts it from a byte buffer into Java objects according to the protocol).
5. The parsed response is returned to the Request Manager.
6. The Request Manager identifies the message ID and notifies the relevant request bean.

7. The Request Manager returns the response to the relevant request bean.

Environment Variables

The Usage Query Proxy uses the following environment variables.

Name	Description	Location	Default Value
log4jFND	The minimum severity of Foundation (FND) messages that are printed to the log. Valid values: <ul style="list-style-type: none">■ ERROR■ WARNING■ DEBUG■ INFO	\${HOME}/ .w.ini.pre.env	ERROR
log4jJF	The minimum severity of Java Foundation (JF) messages that are printed to the log. Valid values: <ul style="list-style-type: none">■ ERROR■ WARNING■ DEBUG■ INFO	\${HOME}/ .w.ini.pre.env	ERROR
log4jRTA	The minimum severity of RTA messages that are printed to the log. Valid values: <ul style="list-style-type: none">■ ERROR■ WARNING■ DEBUG■ INFO	\${HOME}/ .w.ini.pre.env	ERROR
log4jUQ	The minimum severity of Usage Query Proxy (UQ) messages that are printed to the log. Valid values: <ul style="list-style-type: none">■ ERROR■ WARNING■ DEBUG■ INFO	\${HOME}/ .w.ini.pre.env	ERROR
IS_JTES_ENABLED	Indicates whether the J2ES API is enabled.	ADJ1_PROC_PARAMS	Y



Note: The operational data must be related to the OP1J2EESERVER task (in the GN1_TASK_CONNECT table).

Parameters

The Usage Query Proxy requires the *procInstanceName* parameter, which is defined in Amdocs Monitoring & Control.

Configuration Parameters

The configuration parameters (properties) of the Usage Query Proxy are defined in the database.

The following table shows the performance tuning (configuration) parameters of the Usage Query Proxy. These parameters are defined in the ADJ1_CONF_SECTION_PARAM table using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*).

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
FND	fnd.connection.test	query	SELECT 1 FROM DUAL	The query that checks the connection.
FND	fnd.db	fetchSize	50	The size of the bulk that can be fetched.
J2ES	diameter.connection.fail.connect	max_retry	5	The maximum number of attempts to connect to the target Event Server using the Diameter protocol.
J2ES	diameter.connection.fail.connect	reconnect_cycle_sleep_time_seconds	60	The sleep time before trying to connect to the target Event Server if the connection has not been established after max_retry times.
J2ES	diameter.connection.fail.connect	sleep_time_seconds	3	The time to wait between reconnection attempts after a connection failure.
J2ES	diameter.connection.protocol	acc_application_id	11580	The value of the Acct-Application-Id AVP used in the Diameter protocol.
J2ES	diameter.connection.protocol	auth_application_id	4	The value of the Auth-Application-Id AVP used in the Diameter protocol.
J2ES	diameter.connection.protocol	destination_realm	amdocs.com	The destination realm used in the Diameter protocol.
J2ES	diameter.connection.protocol	original_realm	amdocs.com	The original realm used in the Diameter protocol.
J2ES	diameter.connection.protocol	product_name	J2ES	The product name used in the Diameter protocol.
J2ES	diameter.connection.response	timeout_milliseconds	3000	The timeout to wait for a connection to be established.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
J2ES	metric	active	TRUE	Indicates whether metric reporting is enabled.
J2ES	metric	latency_bands_milliseconds	0,50,100,200,300	The latency bands for the entire flow.
J2ES	metric	network_latency_bands_ms	0,50,100,150,200	The latency bands for the network part of the flow.
J2ES	metric	time_refresh_seconds	5	The time in seconds between the updates of metric values.
J2ES	metric	time_window_seconds	300	The time window in which metric average values are calculated.
J2ES	target	cruise_control_max_trx_per_period	500	The maximum number of transactions that can be sent to the target instance per time period.
J2ES	target	cruise_control_period_ms	1000	The time period in milliseconds during which transactions can be sent to the target Event Server.
J2ES	target	max_connections	3	The maximum number of connections to the target Event Server.
J2ES	target	max_open trx_per_target	10000	The maximum number of open transactions (transactions for which a response has not been received) per Event Server.
J2ES	target	max_resends	3	The maximum number of attempts to resend the event to the target Event Server after a timeout.
J2ES	target	trx_sync_timeout_milliseconds	3000	The time for which the thread that sent a sync transaction to the Event Server must wait for a response before timing out.
J2ES	target.manager	max_transactions_threads	5	The maximum number of threads per target Event Server.
J2ES	target.manager	throttling_bulk_size	200	The size of a bulk for moving transactions from the throttling queue to the target queue.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
J2ES	target.manager	throttling_max_threads	2	The maximum number of threads to be fetched from the throttling queue if the target Event Server enters throttling mode.
J2ES	target.overloaded	sleep_time_milliseconds	1000	The time to wait after the maximum number of open transactions has been reached.
J2ES	target.throttling	max_resends	3	The maximum number of attempts to send an event to a target Event Server if the Event Server returned a busy response.
J2ES	target.throttling	sleep_time_milliseconds	10000	The time to wait between the attempts to resend an event to a target Event Server if the Event Server returned a busy response.
UQ_PROXY	uq.character.encoding	class	ISO-8859-1	The class for encoding strings.
UQ_PROXY	uq.connection	maxConnectAttempts	-1	The maximum number of attempts that the server makes to reconnect if the connection was dropped (-1 means retry until connection is established).
UQ_PROXY	uq.connection	maxRenovatorAttempts	2	The maximum number of attempts to connect to the Usage Query server, after which all subsequent requests to that server are immediately returned with an error.
UQ_PROXY	uq.jtes	enabled	\${IS_JTES_ENABLED}	Indicates whether the J2ES API is enabled.
UQ_PROXY	uq.networkconnection.timeout	interval	10	The interval (in seconds) at which the Usage Query Proxy attempts to obtain a valid network connection.
UQ_PROXY	uq.pagination.page.size	threshold	50000	The maximum page size allowed in the system.
UQ_PROXY	uq.pending.requests	threshold	100	The maximum number of requests waiting for a response from the Usage Query Server.
UQ_PROXY	uq.renovator.sleepTime	init	1000	The initial sleep time of the Connection Renovator.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
UQ_PROXY	uq.renovator.sleepTime	max	30000	The maximum sleep time of the Connection Renovator.
UQ_PROXY	uq.request.manager.suspend	timeout	300	The amount of time (in seconds) for which the Request Manager can wait for a write lock if it is not held by another thread during this period.
UQ_PROXY	uq.timeout.response	threshold	30000	The maximum time in milliseconds to wait for a response from the Usage Query Server. The maximum sleep time of an application server is 60000 milliseconds.
UQ_PROXY	uq.vdb	localReplicationMode	N	Indicates whether the local replication mode is enabled. In this mode, read requests are sent to the Usage Query Server even if the cycle has moved to the alternate database.
UQ_PROXY	uq.worker.thread.pool.core.size	response	2	The number of Response Worker threads that can run in parallel.
UQ_PROXY	uq.worker.thread.pool.core.size	request	2	The number of Request Worker threads that can run in parallel.

For each process instance, an entry with the name of the process instance must be created in the ADJ1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
UQ_PROXY123	UQ_PROXY

Any parameter to be overridden at the process instance level must be added to the ADJ1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
UQ_PROXY123	uq.server	port	3456

For more information on parameters and the properties mechanism, see the “Configuration Parameter Mechanism” appendix.

Database Connections

The process connects to tables in the following Turbo Charging areas:

- ADJ Ref area
- ADJ APPL area
- ADJ Config area

Admin Commands

The Usage Query Proxy supports the following admin commands. Because the process resides on the application server, the operator can send these commands only via Amdocs Monitoring & Control and not via the command line.

Command	Description
REFRESH_ALL_GENCODE_AND_IMPL_TABLES_COMMAND	Refreshes all generated implementation libraries and implementation reference tables
REFRESH_IMPL_TABLES_COMMAND	Refreshes all implementation reference tables

In addition, the Usage Query Proxy can get the following command via Amdocs Monitoring & Control or when the Cycle Maintenance process is run.

Command	Description
REFRESH_CYCLE_STATE	Refreshes the ADJ1_CYCLE_STATE table

Recovery Instructions

Restart the Application server.

26 ADJ1UQSRV – Usage Query Server

This chapter describes the Usage Query Server process.

Description

The Usage Query Server process receives requests for data of rated events or accumulators, extracts this data from the Usage database, and returns it as an answer to each request. The requests are received from and returned to the Usage Query Proxy, which is a client of the Usage Query Server.

Process Type

Daemon

Run Frequency

Runs continuously

Activation

Command Line:	ADJ1_UQ_SERVER_Daemon_Shell_Sh -n <APPLICATION_ID> -c "-f <CCF_FILE_NAME>" -e <EWD_EXE>  Note: <i>The -e parameter is optional. It is used in environments with Availability Manager.</i>
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_UQ_SERVER_Daemon_Shell_Sh
Executable Name:	gcpf1fwcApp

Example:

```
ADJ1_UQ_SERVER_Daemon_Shell_Sh -n UQ_SERVER400 -c "-f
Apri1_UQ_Server.ccf" -e gn1avm_ewd
```

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> UQ_SERVER \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

```
ADJ1_Send_Admin_Command_Sh hpbl820 UQ_SERVER 400
CPF_GracefulShutdownCommand
```

Affected Applications

The process affects the following applications:

- *Usage Query Proxy* – The Usage Query Proxy is a client of the Usage Query Server. Therefore, the Usage Query Proxy cannot send requests to the Usage Query Server if the latter is not running.
- *The clients of the Usage Query Proxy* – These clients do not receive answers to requests if the Usage Query Server process is not running.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*
ADJ1UQSRV_<APPLICATION_ID>_%D_%T_%n.log
Example:
ADJ1UQSRV_UQ_SERVER400_20081109_130927_1.log
- *Console log files:*
ADJ1_UQ_SERVER_Daemon_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log
Example:
ADJ1_UQ_SERVER_Daemon_inner_Sh_UQ_SERVER400_20081109_130927.log
- *Operational log files:*
ADJ1UQSRV_<APPLICATION_ID>_\${ABP_MARKET}_<Date>_<Time>.log
Example:
ADJ1UQSRV_UQ_SERVER400_M3G_20081109_130927.log
- *Environment variable log files:*
ADJ1_UQ_SERVER_Daemon_Shell_Sh_<APPLICATION_ID>_<Date>_<Time>.log
Example:
ADJ1_UQ_SERVER_Daemon_Shell_Sh_UQ_SERVER400_20081109_130927.log

- *Latency log files:*

UQ_LATENCY_REPORT_<Date>_<Time>_<ID>.log

Example:

UQ_LATENCY_REPORT_20100914_110837_0.log

- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG
- *Environment variable log files* – \$ABP_LOG
- *Latency log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter)

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information.
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).
- *Operational log files* – The output of operational scripts.
- *Environment variable log files* – Environment variables used by the process.
- *Latency log files* – Information on request latency. Each latency log file contains a header with a list of parameters and the prints of the latency report. Every record of the report starts on a new line. The values of the parameters in each record are separated by a semicolon (‘;’). They appear in the following order:
 - *Work thread ID* – An identifier of the thread.
 - *Unique transaction ID* – An identifier that is attached to the message received from the Usage Query Proxy.

- *Network thread ID* – The network thread ID and the internal stream ID, separated by a colon. When a client connects to the Usage Query Server, it is handled by a separate thread. The internal thread ID that is printed to the latency log can be used to retrieve the client network address that is printed to the console log in the following format: [2012-12-10 09:45:41.0000] INFO - Stream id 0 from client 10.234.9.62:54178 , accepted.
- *Request type* – The type of the request:
 - *EL* – Event list
 - *ED* – Event details
 - *PL* – Accumulator list
 - *PD* – Accumulator details
- *Response status* – The status of request handling.
- *Page size* – The value of the input Page Size parameter received with the request.
- *Page number* – The value of the input Page Number parameter received with the request.
- *Request time stamp* – The date and time when the Usage Query Server started handling the request, with millisecond resolution.
- *Processing finish time stamp* – The date and time when the Usage Query Server finished handling the request, with millisecond resolution.
- *Request backlog* – The delta between the time when the request entered the queue and the time when it left the queue, with millisecond resolution.
- *Open cursor* – The delta between the time when the process started opening the cursor and the time when it finished, with millisecond resolution.
- *Process duration* – The delta between the time when the process started the first fetch operation and the time when it finished the fetch operation, with millisecond resolution.
- *Number of received record fetches* – The total number of physical records received from the database according to the request (i_physicalRecordCount).
- *Number of sent record fetches* – The total number of records sent as a response to the request (m_actualRecordCount).
- *Total transaction duration* – The total duration of request handling, with millisecond resolution.
- *Transaction processing duration without in-queue* – The duration of request handling, not including the time spent pending in a queue.
- *First fetch time* – The time it took to perform the first fetch operation.
- *Cycle code* – The cycle code.
- *Cycle instance* – The cycle instance.

- *Customer ID* – The ID of the customer.
- *Customer segment* – The customer segment.
- *Subscriber ID* – The ID of the subscriber.
- *Version ID* – The ID of the version.

Example:

Work Thread Id, Unique Transaction Id, Network Thread Id, Response Type, Response Status, Page size, Page number, Request Timestamp(datetime), Processing finish timestamp(datetime), Request Backlog(ms), Open Cursor(ms), Process Duration(ms), Received record fetch(count), Send record fetch(count), Total Transaction Duration(ms), Transaction Processing Duration without in-queue(ms), First fetch time(ms), Cycle code, Cycle instance, Customer ID, Customer segment, Subscriber ID, Version ID

1117808960 ;2; 1306626368:7; EL; OK; 10; 1; 17042014 143933 165; 17042014 143933 167; 0; 2; 0; 0; 0; 2; 2; 0; 1; 1 ;0; 0; 0; 11;

- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Flow

The Usage Query Server process (ADJ1UQSRV) executes the following steps:

1. The process is initialized.
2. The working threads are initialized.
3. Each working thread performs the following, for each request that it handles:
 - a. Accepts the request from the client, parses it, and stores the request parameters in local variables.
 - b. Opens a cursor in the Usage database for extracting the requested data of events or accumulators.
 - c. Fetches the records from the database and formats each record according to the requested output format. The formatted output records are put into an output message (or into multiple messages if the buffer of one message cannot contain all the records).
 - d. Sends the output messages as a response to the client.
 - e. When no relevant data is found in the database for the query, returns a response with NO DATA status.
 - f. When an error occurs while processing the request, returns a response with ERROR status.

When Geo Redundancy is enabled, the Usage Query Server is active both at the active site and at the standby site. At the standby site, the Usage Query Proxy does not send update events to the Usage Query Server, so the Usage Query Server normally performs read operations only (although it is capable of performing update operations as well).

Environment Variables

All the environment variables that affect the process are defined automatically by the Usage Query Server activation script.

The Usage Query Server uses the following environment variables that are initialized by the op_adj_env_sh script.

Variable Name	Description	Valid Values/ Default Value
ADI_UQ_EXPIRATION_THRESHOLD	An indication of the duration of the Usage Query Proxy time-out per request	30000
APR_UQ_SERVER_NUM_OF_THREADS	The number of Usage Query threads	1

For more information on the script, see the “Amdocs Batch Job Manager Application Script” section in the “Introduction” chapter.

In addition, the following environment variables enable *testing* the process while using simple networking instead of regular networking, for receiving requests and returning responses.



Attention: Simple networking is only to be used for TESTING purposes when regular networking cannot be used for some reason.

Name	Description	Default Value	Method of Changing the Variable
UQ_DUMMY_SERVER_LOG_NAME	If the UQ_DUMMY_SERVER_IND environment variable is set to ‘Y’, this is the name of an additional log file.	N/A	export UQ_DUMMY_SERVER_PORT=<log file name>
UQ_DUMMY_SERVER_IND	Indicates whether to use simple networking.	N	export UQ_DUMMY_SERVER_IND=Y
UQ_DUMMY_SERVER_PORT	If the UQ_DUMMY_SERVER_IND environment variable is set to ‘Y’, this is the number of the port to which the Usage Query Server listens for requests from the client and to which it returns responses.	N/A	export UQ_DUMMY_SERVER_PORT=<port number>

Parameters

The following parameters must be set for the Usage Query Server to run properly.

Parameter	Description	Default Value
APPLICATION_ID	The process instance	N/A
CCF_FILE_NAME	An XML file that contains the relevant configuration	Apr1_UQ_Server.ccf

Configuration Parameters

The configuration parameters (properties) of the Usage Query Server are defined in the database.

The following table shows the performance tuning (configuration) parameters of the Usage Query Server. These parameters are defined in the APR1_CONF_SECTION_PARAM table using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*).

PARAM_CLASS	SECTION_NAME	PARAM_NAME	Default Value	Description
APE	config	CYCLE_LOAD_FROM_COVERAGE	90	The number of days before the system date for which the cycle is loaded. The value of this parameter must be set according to the date ranges expected for usage queries across multiple cycle instances.
APE	config	CYCLE_LOAD_TO_COVERAGE	90	The number of days after the system date for which the cycle is loaded. The value of this parameter must be set according to the date ranges expected for usage queries across multiple cycle instances.
APE	config	QUERY_TRACING_DIR	\${ABP_APP_LOG}	The location of trace XML files that show query results.
APE	config	QUERY_TRACING_MODE	FALSE	Indicates whether trace XML files that show query results must be printed.
UQ_SERVER	config	ACCUM_HINT_CLAUSE	/*+ full (APE1_ACCUMULATORS) */	The hint for the SQL query used for retrieving the accumulators.
UQ_SERVER	config	EVENT_HINT_CLAUSE	/*+ full (APE1_RATED_EVENT) */	The hint for the SQL query used for retrieving the rated events.
UQ_SERVER	config	PROXY_RESPONSE_EXPIRATION_THRESHOLD	\${ADJ_UQ_EXPIRATION_THRESHOLD}	An indication of the duration of the Usage Query Proxy time-out per request.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	Default Value	Description
UQ_SERVER	config	UQ_DB_CONN_ERROR_THRESHOLD	5	The maximum number of error messages related to the database connection per cycle and per working thread. When this number is exceeded for a cycle on a working thread, the thread receives an indication to switch to the replication database. All the cycles that use the connection on the thread also switch to the replication database.
UQ_SERVER	config	UQ_INITIATE_SECONDARY_DB_CONN_IND	false	Indicates whether the Usage Query Server must open a connection to the replication (secondary) database in the initialization phase or only when the Usage Query Server switches to work with the replication database. Valid values: <ul style="list-style-type: none"> ▪ <i>false</i> – The connection to the replication database is established only when necessary. ▪ <i>true</i> – The connection to the replication database is established in the initialization phase.
UQ_SERVER	config	UQ_LATENCY_REPORT_ENABLED	false	Indicates whether latency reports are enabled by default.
UQ_SERVER	config	UQ_MAX_FETCH_BULK_SIZE	100	The maximum number of records to be fetched in one request. This parameter defines the buffer size.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	Default Value	Description
UQ_SERVER	config	UQ_MIN_NEEDED_BYTES_FOR_ONE_OUTPUT_RECORD	10 * 1024 bytes	The minimum assumed number of bytes that is required for one output record of the Usage Query Server. An output record may contain an external record, an event, or a dimension of an accumulator. This value is used by the Usage Query Server to check whether the space that is left in the current output message buffer is sufficient for writing another record into it. If the space that is left in the current output message buffer is less than this number of bytes, the current output message is sent with the output records that it already contains, and then another message is sent with additional output record or records.
UQ_SERVER	config	UQ_NUM_OF_DB_RECONNECT_ATTEMPTS	3	The maximum number of consecutive attempts to reconnect to a disconnected database for each request.
UQ_SERVER	config	UQ_QUERY_EVENTS_FROM_SECONDARY	true	Indicates whether the Usage Query Server will extract events from the replication (secondary) database. Valid values: <ul style="list-style-type: none"> ■ <i>true</i> – All extract requests are directed to the replication database. ■ <i>false</i> – When the Usage Query Server is connected to the replication database, requests for event extracts result in an error.
UQ_SERVER	config	UQ_RESPONSE_MAX_BUFFER_SIZE_KBYTE	100 KB	The size of the buffer for <i>one</i> output message of the Usage Query Server (a response to a query may consist of several output messages).
UQ_SERVER	config	UQ_RETURN_RECORD_COUNT_IND	N	Indicates whether to return the count of ALL output records when the first page is requested. If this indicator is set to Y, the performance of the Usage Query Server decreases significantly.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	Default Value	Description
UQ_SERVER	config	UQ_SRV_SUBSCRIBER_LAZY_PARTIAL_DATA_LEVEL	Possible values: ■ ALL_DATA (default) ■ SUBS_OFFERS_AND_GLOBAL_PARAMS_DATA ■ SUBS_AND_OFFERS_DATA ■ ONLY_SUBS_DATA	Enables the retrieval of only a part of the subscriber's data (in mapping case functions) from the database. Retrieving only the required part of the subscriber's data can increase the performance of the Usage Query Server.
UQ_SERVER	config	UQ_STATEMENT_CACHE_SIZE	10	The maximum number of already prepared event query statements or accumulator query statements that can be held in the cache of the process. If at some point, the process must use more statements, the cache is cleared and refilled with new statements.
UQ_SERVER	config	UQ_TIME_BETWEEN_RECONNECT_ATTEMPT	1	The time (in seconds) for which the Usage Query Server must wait before it tries to reconnect the database after an error.
UQ_SERVER	config	VIRTUAL_OR_SHARED_MEMORY_USED	VIRTUAL	Indicates whether the process uses virtual or shared memory. For the Usage Query Server, the value must always be VIRTUAL.
UQ_SERVER	LOGGER.APR.UQ_LATENCY	Enable	Y	Indicates whether the latency report is to be written to a latency log file of the Usage Query Server.
UQ_SERVER	LOGGER.APR.UQ_LATENCY	Outputs	UQGNRL	The list of output objects assigned to the latency log file.
UQ_SERVER	LOGGER.APR.UQ_LATENCY	Severity	0	The severity level of the latency log file.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	Default Value	Description
UQ_SERVER	LOGGER.output.UQGNRL	BufferSize	\${BUFFER_SIZE}	The buffer size for the latency log file.
UQ_SERVER	LOGGER.output.UQGNRL	Filename	\${ABP_APRLOG}/UQ_LATENCY_REPORT_%D_%T_%n.log	The path and the name of the latency log file.
UQ_SERVER	LOGGER.output.UQGNRL	FlushPeriod	\${FLUSH_PERIOD}	The flush period for the latency log file, which determines the frequency with which the Logger flushes the data from the temporary buffer to the file.
UQ_SERVER	LOGGER.output.UQGNRL	ImmediateFlush	Y	Indicates whether records must be immediately flushed to a file without using buffer optimizations.
UQ_SERVER	LOGGER.output.UQGNRL	Layout	Latency_Layout	Specifies the layout type to be used by the output class to format the latency log file.
UQ_SERVER	LOGGER.output.UQGNRL	MaxFileSize	\${MAX_FILE_SIZE}	The maximum size of the latency log file.
UQ_SERVER	LOGGER.output.UQGNRL	Mode	\${MODE}	The mode of the latency log file. Possible modes are: <ul style="list-style-type: none">■ <i>Synch</i> – The output runs in the same thread as the caller application.■ <i>Asynch</i> – The output runs in a separate thread.

In addition, the process uses Light Tracer configuration parameters, which are defined in the ADJ1_CONF_SECTION_PARAM table. For more information, see [Light Tracer Parameters](#) in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

For each process instance, an entry with the name of the process instance must be created in the APR1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
UQ_SERVER400	UQ_SERVER

Any parameter to be overridden at the process instance level must be added to the APR1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
UQ_SERVER400	config	UQ_SRV_SUBSCRIBER_LAZY_PARTIAL_DATA_LEVEL	ONLY_SUBS_DATA

For more information on parameters and the properties mechanism, see the “Configuration Parameter Mechanism” appendix.

Configuration Files

The Usage Query Server requires a configuration file that is defined via an internal configuration tool.

The environment variables that appear in the .ccf file and can be customized are defined in the op_adj_env_sh script (for more information, see the “Environment Variables” section in this chapter).

Database Connections

The process connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	ADJ1_USAGE_DB_STATE	Select
Configuration	ADJ1_UQ_DB_CONN_CFG	Select
Usage	<ul style="list-style-type: none">■ APE1_RATED_EVENT■ APE1_ACCUMULATORS■ APE1_EVENT_TO_EXT_REC■ APE1_ACCUM_TO_EXT_REC	Select

By default, the Usage Query Server connects to all Usage database instances and handles all cycles. The ADJ1_UQ_DB_CONN_CFG table maintains the relations between the Usage Query Server and the database instance, and enables the operator to control the number of database connections. For the purposes of high availability of the Usage Query mechanism, at least two Usage Query Servers must handle each database instance.

Example:

UQ_APPLICATION_ID	USAGE_DB_CONNECT_CODE
UQ_SERVER401	ADJUSG1
UQ_SERVER400	ADJUSG1
UQ_SERVER402	ADJUSG1

The Usage Query Server maintains two physical database connection destinations for any logical Usage database that it works against: one is master, and the other is replication, which serves as a back-up. The replication database is local to the site. The data is replicated from the master to the replication database using third-party software. The master and replication databases are defined in the ADJ1_USAGE_DB_STATE table.



Note: The terms “master database” and “replication database” refer to complete databases. In other words, if each of the databases is a RAC-based cluster, the Usage Query Server switches to the replication database after a failure of both instances that the master RAC manages or of the storage on which the RAC database instances rely for database files.

By default, the Usage Query Server connects to the master Usage database in the initialization phase. If the master database fails, the Usage Query Server switches to work against the replication database. The decision to switch to the replication database is made for each cycle and working thread in each Usage Query Server independently, according to the configuration parameters. Threads do not share any information about the connection, and each Usage Query Server maintains a counter of attempts to connect to the master database for each cycle. Switching to the replication database is initiated as soon as the maximum number of attempts is exceeded for one cycle on a thread. When the switch occurs, all the cycles on the thread that share the connection are switched to the replication database.

Switching back to the master database is controlled only by the operator, who sends an admin command to the process. For more information, see the “Configuration Parameters” and “Admin Commands” sections in this chapter.

Whenever an operator wants to switch the work of the Usage Query Server from the master database to the replication database or vice versa, the operator must run the **ADJ1SETDBACT – Set Database Activities** job. This job updates the ADJ1_USAGE_DB_STATE table and sends the relevant admin command to the Usage Query Server.

If Geo Redundancy is enabled, and if the Usage Query Server at the active site is not functional for some reason, the request can be routed to the standby site, even without a switchover. In this case, responses to usage queries are received from the replicated database at the standby site and not from the local replication database. However, because of a replication delay, the data in the Usage database at the standby site might be not as fresh as at the active site.

Admin Commands

The Usage Query Server supports the following admin commands. For information on how to execute the commands, see the “Handling Admin Commands” section in the “High Availability” chapter.

Command	Description	Parameters
REFRESH_CYCLE_COMMAND	Refreshes cycle data from the ADJ1_CYCLE_STATE table	N/A
REFRESH_IMPL_TABLES_COMMAND	Refreshes all implementation reference tables	N/A
REFRESH_ALL_GENCODE_AND_IMPL_TABLES_COMMAND	Refreshes all generated implementation libraries and implementation reference tables	N/A
REMOVE_MEMORY_COMMAND	Removes accumulators related to a specific cycle instance and cycle code from shared memory	--cycleCode "<Cycle code>" --cycleInstance "<Cycle instance>" --cycleYear "<Cycle year>"
UQ_ADMIN_DB_ACTIVITY	Depending on the Activity Type parameter, performs various activities on the given connect code.	--requiredActivity <Activity type> --dbConnectionString <Connect code> where <Activity type> can be one of the following: <ul style="list-style-type: none"> ■ 0 – Print information about the current database to the log file. ■ 1 – Reconnect to the master database. ■ 2 – Switch to the replication database.

Command	Description	Parameters
UQ_ADMIN_ENABLE_TRACE_CUSTOMER_ONCE	<p>When the XML tracing in the Usage Query Server is disabled (for example, in production), temporarily enables XML tracing for a single query, showing the response to the first query for the customer whose ID is specified in the command. The trace XML file is created in addition to the regular response to the client. It is filled with data only if the response contains external records.</p> <p>After producing the trace XML file for one query, the Usage Query Server disables the tracing mode. Then, the command can be sent again for the same customer or a different customer.</p>	--customerId <Customer ID>
UQ_LATENCY_REPORT_ENABLED	Activates and deactivates latency logs	-- enableLatencyLogInd “<Y/N>”
SET_LIGHT_TRACER_COMMAND	Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters	<ul style="list-style-type: none"> ▪ Activation parameters: <ul style="list-style-type: none"> • <i>turnTracer “on/off”</i> – Activates or stops the Light Tracer • <i>--turnAssert “on/off”</i> – Activates or stops Assertions • <i>--turnLogOnTrace “on/off”</i> – Activates or stops writing log messages in Light Tracer files • <i>--turnFlowTracer/turnAll “on/off”</i> – Activates or stops tracing a single flow or event that is transferred between machines • <i>--turnAll “on/off”</i> – Activates or stops all of the above. The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter). ▪ <i>--moduleID “ALL”</i> ▪ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • <i>--threadInfo “Y/N”</i> – Specifies whether trace or assertion messages include the thread information.

Command	Description	Parameters
		<ul style="list-style-type: none"> • <code>--timestampInd "Y/N"</code> – Specifies whether trace or assertion messages include the time stamp. • <code>--contextInd "Y/N"</code> – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • <code>--noOfMessages "<number>"</code> – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down. • <code>--period "5/10/20/60/120/480/1440"</code> – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated. <p> Note: If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached.</p> <ul style="list-style-type: none"> • <code>--assertAbortInd "Y/N"</code> – Specifies whether the application is to be aborted if the assertion condition fails. ▪ <code>--commandType "changeBufferSize"</code> – Resizes the Light Tracer buffer at run time. <ul style="list-style-type: none"> • <code>--bufferSize "<number>"</code> – The new size for the Light Tracer buffer.

Command	Description	Parameters
SET_TRACE_SQL_COMMAND	<p>Activates or stops SQL tracing at runtime.</p> <p>The command name with its parameters must be enclosed in quotation marks.</p> <p><i>Example:</i></p> <pre>ADJ1_Send_Admin_Command_Sh illin1027 <process instance> "SET_TRACE_SQL_COMMAND -- traceSqlOn \"yes\" --duration \"1\""</pre>	<p>--traceSqlOn "<yes or no>" --duration "<minutes>"</p> <p>If the IS_DURATION_REQUIRED environment variable is set to 'Y', the --duration parameter is mandatory. If the duration specified in the command exceeds the allowed maximum value, SQL tracing is activated for the maximum allowed period of time only (60 minutes).</p>  <p><i>Note: The SHUT_DOWN_CHECK_FREQ parameter of the Light Tracer in the ADJ1_CONF_SECTION_PARAM table determines the frequency (defined in the number of messages) at which the SQL Trace Manager checks whether the period specified in the --duration parameter has passed. Therefore, SQL tracing is stopped after the specified duration or after this number of messages has been reached, whichever comes later.</i></p>
SWITCH_LIGHT_TRACER_ON ALL	Activates the Light Tracer without the ability to specify the mode or send parameters.	N/A
SWITCH_LIGHT_TRACER_OFF ALL	Stops the Light Tracer	N/A
CPF_GracefulShutdownCommand	Shuts down the Usage Query Server gracefully	N/A

Examples:

- ADJ1_Send_Admin_Command_Sh hpbl820 UQ_SERVER 400 REFRESH_IMPL_TABLES_COMMAND
- ADJ1_Send_Admin_Command_Sh hpbl820 UQ_SERVER 400 REMOVE_MEMORY_COMMAND --cycleCode "1" --cycleInstance "9" --cycleYear "2008"
- ADJ1_Send_Admin_Command_Sh ilhp092 UQ_SERVER 400 UQ_LATENCY_REPORT_ENABLED --enableLatencyLogInd "Y"

Recovery Instructions

Rerun the process.

27 ADJ1SETDBACT – Set Database Activities

This chapter describes the Set Database Activities job.

Description

The Set Database Activity job sends the UQ_ADMIN_DB_ACTIVITY admin command to the Usage Query Server and handles the switches of the active Usage database from the master database to the replication database and vice versa.

Process Type

Batch job

Run Frequency

By request, whenever the operator wants the Usage Query Server to switch between the master and the replication databases

Activation

Command Line:	RunJobs ADJ1SETDBACT BYREQ
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_DB_ACTIVITY_Job_Sh
Executable Name:	N/A

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Affected Applications

The process affects the following application:

- *Usage Query Server* – Sets the information that determines the database instance to which the Usage Query Server must connect.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Log files are created and handled in the default way.

Name

- Log files:

ADJ1SETDBACT_<ACTIVITY_TYPE>_<Date>_<Time>.log

Example:

ADJ1SETDBACT_0_20120103_174003.log

- Console log files:

ADJ1SETDBACT_<ACTIVITY_TYPE>_<Date>_<Time>.log

Example:

ADJ1SETDBACT_0_20120103_174003.log

- Operational log files:

ADJ1SETDBACT_<Job Record Name>_<ABP_MARKET>_<Date>_<Time>.log

Example:

ADJ1SETDBACT_ByReq_M3G_20120103_174002.log

Location

Log files are located in the following directories:

- *Log files* – \$ABP_APP_ROOT/log/
- *Console log files* – \$ABP_APP_ROOT/log/
- *Operational log files* – \$ABP_LOG

Contents

The log files contain the following:

- *Log files* – The process error information
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands)
- *Operational log files* – The output of operational scripts

Flow

The Set Database Activities job executes the following steps:

1. Sets the connections to the ADJAPPL database
2. Updates the ADJ1_USAGE_DB_STATE table according to the ACTIVITY_TYPE value
3. Sends an admin command to all Usage Query Servers via the Availability Manager
4. Exits

Parameters

The following parameters must be set for the process to run properly.

Parameter	Description	Default Value	Method of Changing the Parameter Manually
ACTIVITY_TYPE	One of the following activity types: <ul style="list-style-type: none">■ 0 – Report to the log.■ 1 – Switch back to the master database.■ 2 – Switch to the replication database.	N/A	Operational job parameter. Can be updated through Amdocs Monitoring & Control.
CONNECT_CODE	The connect code to the database. The master (primary) connection and not the current connection must be used both for switching to the replication database and for switching back.	N/A	Operational job parameter. Can be updated through Amdocs Monitoring & Control.

Database Connections

The process connects to the following table during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	ADJ1_USAGE_DB_STATE	Update

28 ADJ1CEEXT – Cycle Events Extract

This chapter describes Cycle Events Extract process.

Description

The Cycle Events Extract job extracts data of rated events from the Usage database according to the cycle code, cycle instance, and cycle year. It formats the data according to the format name and writes it to output files. These files are registered in Audit & Control.

Process Type

Batch job

Run Frequency

By request

Activation



Caution: When the home database is up after a failure, the operator must not activate Usage Extracts until the Copy Cycle Usage process, which copies the data from the alternate database to the home database, is complete.

Command Line:	RunJobs ADJ1CEEXT BYREQ
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_CycleEventExtract_Job_Sh
Executable Name:	gcpf1fwcApp

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> US_EX_CY_EVENT \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

ADJ1_Send_Admin_Command_Sh hpbl820 US_EX_CY_EVENT 205
CPF_GracefulShutdownCommand

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*

ADJ1CEEXT_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1CEEXT_US_EX_CY_EVENT205_20081109_130927_1.log

- *Console log files:*

ADJ1_CycleEventExtract_Job_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1_CycleEventExtract_Job_inner_Sh_US_EX_CY_EVENT205_20081109_130927.log

- *Operational log files:*

ADJ1CEEXT_<Job Record Name>_\${ABP_MARKET}<Date>_<Time>.log

Example:

ADJ1CEEXT_BYREQ_M3G_20081109_130927.log

- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter)

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information.
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file, and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).

In addition, the console log contains the number of records written to output files. These records are reported per database partition when the process is finished.



Note: The number of records written to the file matches the number of external records and does not match the number of extracted events.

Extract statistics are printed as follows:

```
EXTR STATISTICS: Total output record created [%d], partition [%s]
```

where:

- *%d* – The number of records written to files
- *%s* – The name of the database partition
- *Operational log files* – The output of operational scripts.
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Output Files

This section describes the output files.

Name

Output files use the following naming conventions:

- *Regular files*
EventExtracts<*CustomSuffix*><*DateTime*>_<*PID*>_<*ThreadID*>_<*FileNo*>.dat
- *Error files*
 - errEventExtracts<*CustomSuffix*><*DateTime*>_<*PID*>_<*ThreadID*>_<*FileNo*>.dat
 - custerrEventExtracts<*CustomSuffix*><*DateTime*>_<*PID*>_<*ThreadID*>_<*FileNo*>.dat

Location and Format

- *Location* – \$ABP_APP_ROOT/interfaces/output/us_extract.
- *Format* – Semicolon-delimited file. The contents are defined by configuration.

Contents

A regular output file contains event fields specified by configuration, for all extracted records. If an error occurred during processing:

- The regular output file is empty.
- The errEventExtracts output file contains the value of -1.
- If the COLLECT_CYCLE_ERRORS configuration parameter is set to 'Y', the custerrEventExtracts file contains the IDs of all customers for which an error occurred during data extraction.

Flow

In the Cycle mode, the Cycle Events Extract process (ADJ1CEEXT) performs the following steps:

1. The Process Manager reads the input parameters and divides the work among the working threads.
2. Each working thread performs the following:
 - a. Accepts the cycle code, instance, and Oracle partition to work on as input.
 - b. Opens the output file.
 - c. Opens a cursor on the Rated Events table using the Extract Engine (an internal component). The cursor extracts the entire contents of the partition:
 - For rated events, a full partition scan is performed.
 - If an additional WHERE clause was specified, it is added to the cursor statement.
 - d. For each record found in the partition:
 - i. If the extract requires the execution of a mapping case, the working thread activates the relevant mapping case using the code generated by the Implementation Compiler from the implementation made in the Rating Logic Configurator.
 - ii. Formats the record according to the formatting definitions. Records are formatted regardless of whether the mapping case was activated.If multiple formats are required, the record is formatted according to each format provided as input. In this case, the output is divided among the output files according to the configured output file names.

If an error occurs during the activation of the mapping case or formatting:

- 1) Does one of the following:
 - ◆ If the COLLECT_CYCLE_ERRORS parameter is set to 'Y', proceeds with a dummy run on the partition to collect all errors and writes the respective customer IDs to the custerrEventExtracts file.
 - ◆ If the COLLECT_CYCLE_ERRORS parameter is set to 'N', stops extracting the data.
 - 2) Marks the partition as failed and exits with failure.
- iii. Writes the record into the output file according to the specified format. If an error occurred during the activation of the mapping case or formatting, the regular output file is truncated, and the errEventExtracts file contains the value of -1.
- e. When the output file is closed, writes it to Audit & Control.
3. Updates a control table (APR1_EXTR_MONITOR) for each partition that has finished successfully. The table is used for the following:
 - In recovery, the table enables the operator to execute the Usage Extract process only for those partitions that have not been processed successfully.
 - The table enables the operator to verify that the entire process finished successfully, and files can be merged so that there remains just one file per group.

Environment Variables

The Cycle Events Extract uses the following environment variable that is initialized by the op_adj_env_sh script.

Variable Name	Description	Valid Values/ Default Value
APR_CYCLE_XTR_NUM_OF_THREADS	The number of threads that the Extract uses	3

For more information on the script, see the “Amdocs Batch Job Manager Application Script” section in the “Introduction” chapter.

Parameters

The following parameters must be set for the process to run properly.

Parameter	Description	Default Value
APPLICATION_ID	The ID of the job that enables it to get its parameters.	US_EX_CY_EVENT205
CCF_FILE_NAME	An XML file that contains the relevant configuration.	APR1_Extracts.ccf
EXTR_CYCLE_CODE	The cycle code for which data is to be extracted.	1
EXTR_CYCLE_FROM_DATE	Defines the lower limit of date range (format: year month day, for example, 20081015; time can also be added, for example, 20081015 091558). In the Cycle Events Extract, this limit refers to the START_TIME of the events and not to the SYS_UPDATE_DATE in the APE1_RATED_EVENT table. The CUST_ADDITIONAL_WHERE_CLAUSE parameter can be used to define the limit of the date range according to the SYS_UPDATE_DATE.	N/A
EXTR_CYCLE_INSTANCE	The cycle instance for which data is to be extracted.	1
EXTR_CYCLE_TO_DATE	Defines the upper limit the of date range (format: year month day, for example, 20081015; time can also be added, for example, 20081015 091558). In the Cycle Events Extract, this limit refers to the START_TIME of the events and not to the SYS_UPDATE_DATE in the APE1_RATED_EVENT table. The CUST_ADDITIONAL_WHERE_CLAUSE parameter can be used to define the limit of the date range according to the SYS_UPDATE_DATE.	N/A
EXTR_CYCLE_YEAR	The cycle year for which data is to be extracted.	N/A
EXTR_FORMAT_NAME	The name of the format that is used by the engine to format extracted data. If required, multiple output formats can be specified. In this case, they must be separated by the pipe (' ') character.	Billing Event Format
EXTR_INSTANCE_ID	The instance ID of the job.	1
EXTR_INSTANCES_NUM	The number of instances that run in parallel.	1
EXTR_OUTPUT_DATA_GROUP	The data group of the output file.	Event.1.1.2009 (sample value)

Parameter	Description	Default Value
EXTR_OUTPUT_FILE_ALIAS	The alias of the output file. If multiple output formats are required, a list of aliases for the formats requested in the EXTR_FORMAT_NAME parameter must be specified. The file aliases in the list must be separated by the pipe (' ') character.	CycleEv
EXTR_OUTPUT_FILE_NAME	The name of the output file. If multiple output formats are required, a list of output file names for the formats requested in the EXTR_FORMAT_NAME parameter must be specified. The file names in the list must be separated by the pipe (' ') character.	EventExtracts.dat
EXTR_OUTPUT_FILE_UNION	The union of the output file.	DUMMY
EXTR_OUTPUT_ROUTING_CRITERIA	The routing criteria for the output file.	RPR1_TO_BL1
MAP_MODE	The context (map) in which the process is currently running.	NONE
MONITOR_RECOVERY_FLAG	A flag that indicates whether the Recovery mode is enabled.	N

Configuration Parameters

In Turbo Charging, the configuration parameters (properties) of the Cycle Events Extract process are defined in the database.

The following table shows the performance tuning (configuration) parameters of the Cycle Events Extract process. These parameters are defined in the APR1_CONF_SECTION_PARAM table using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*).



Note: Other parameters must not be changed.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APE	config	QUERY_TRACING_DIR	\${ABP_APRL_LOG}	The location of trace XML files that show query results.
APE	config	QUERY_TRACING_MODE	FALSE	Indicates whether trace XML files that show query results must be printed.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
US_EX_CY_EVENT	EXTRACT	HINT_CLAUSE	/*+ LEADING (EE) use_hash(EE) FULL (E)*/  Note: <i>This parameter is optional.</i>	The SQL hint used by the Extract Engine (an internal component).
US_EX_CY_EVENT	EXTRACT	MONITOR_RECOVERY_FLAG	\${MONITOR_RECOVERY_FLAG}	A flag that indicates whether the Recovery mode is enabled.
US_EX_CY_EVENT	EXTRACT_ENGINE	CUST_ADDITIONAL_WHERE_CLAUSE	  Note: <i>This parameter is optional and appears if there is an additional WHERE clause.</i>	An additional WHERE clause that can be added to the SELECT statement during customization. The clause must follow proper SQL syntax.
US_EX_CY_EVENT	EXTRACT_ENGINE	RATED_EVENT_FLAG	Y	Indicates whether information is extracted for events. If this flag is not set, information is extracted for accumulators.
US_EXTRACT	config	COLLECT_CYCLE_ERRORS	N	If an error occurs, indicates whether the process must stop extracting and exit with failure ('N'), or proceed to collect all errors and write the relevant customer IDs to a file ('Y').
US_EXTRACT	config	IS_RUN_MODE_JOB	\${ADJ1_IS_JOB}	Indicates whether the extract is running as a job: ■ Y – Yes ■ N – No
US_EXTRACT	config	UQ_MAX_FETCH_BULK_SIZE	100	The maximum number of records to be fetched in one request. This parameter defines the buffer size.
US_EXTRACT	EXTRACT	INSTANCE_ID	\${EXTR_INSTANCE_ID}	The instance ID of the Cycle Event Extract process. This is number of the instance that is running of the total number of instances.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
US_EXTRACT	EXTRACT	INSTANCES_NUM	\${EXTR_INSTANCES_NUM}	The number of Cycle Event Extract instances.
US_EXTRACT	EXTRACT	MAP_TYPE	\${MAP_MODE}	The type of the map in which the extract is running.
US_EXTRACT	EXTRACT	MAX_FETCHED_RECORD	100	<p>The maximum number of records that the Extract Engine (an internal component) can fetch at once.</p> <p>This parameter enables flexible configuration of the bulk size. To set a bulk size that is different from the size specified in the UQ_MAX_FETCH_BULK_SIZE parameter, set this parameter to a lower value than the one defined for the UQ_MAX_FETCH_BULK_SIZE parameter.</p>
US_EXTRACT	EXTRACT	REQUEST_ID	\${REQUEST_ID}	The request ID of the current run. In the Cycle mode, the request ID is a concatenation of the cycle code, instance, and year.
US_EXTRACT	FILE_OUTPUT	FILE_DIRECTORY	\${ABP_AP_Root}/interfaces/output/us_extract	The default directory where the output file is created.
US_EXTRACT	FILE_OUTPUT	FILE_NAME	UsageExtractData.dat	The naming convention for the output file.
US_EXTRACT	FILE_OUTPUT	MAX_FILE_SIZE	0	The maximum size of an output file. The value of 0 means unlimited size.
US_EXTRACT	FILE_OUTPUT	MAX_RECORD_NUMBER	10000	The maximum number of records that can be written to a single file.

In addition, the process uses Light Tracer configuration parameters, which are defined in the ADJ1_CONF_SECTION_PARAM table. For more information, see **Light Tracer Parameters** in the “**Configuration Parameters**” section in the “**ADJ1EVENTSRV – Event Server**” chapter.

For each process instance, an entry with the name of the process instance must be created in the APR1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
US_EX_CY_EVENT205	US_EX_CY_EVENT

Any parameter to be overridden at the process instance level must be added to the APR1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
US_EX_CY_EVENT205	FILE_OUTPUT	MAX_RECORD_NUMBER	5000

For more information on parameters and the properties mechanism, see the “**Configuration Parameter Mechanism**” appendix.

Configuration Files

The Usage Extract process requires a configuration file (.ccf) that is defined via an internal configuration tool.

The environment variables that appear in the .ccf file and can be customized are defined in the op_adj_env_sh script (for more information, see the “**Environment Variables**” section in this chapter).

Database Connections

The process connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	■ AC1_CONTROL ■ APR_EXTR_MONITOR	Insert Insert, Delete, Update
Customer	■ APE1_SUBSCR_DATA ■ APE1_SUBSCR_OFFERS ■ APE1_SUBSCR_PARAMS	Select
Usage	APE1_RATED_EVENT	Select

Admin Commands

The Cycle Event Extract supports the following admin commands. For information on how to execute the commands, see the “Handling Admin Commands” section in the “High Availability” chapter.

Command	Description	Parameters
SET_LIGHT_TRACER_COMMAND	Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters	<ul style="list-style-type: none"> ▪ Activation parameters: <ul style="list-style-type: none"> • <i>turnTracer “on/off”</i> – Activates or stops the Light Tracer • <i>--turnAssert “on/off”</i> – Activates or stops Assertions • <i>--turnLogOnTrace “on/off”</i> – Activates or stops writing log messages in Light Tracer files • <i>--turnFlowTracer/turnAll “on/off”</i> – Activates or stops tracing a single flow or event that is transferred between machines • <i>--turnAll “on/off”</i> – Activates or stops all of the above. The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter). ▪ <i>--moduleID “ALL”</i> ▪ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • <i>--threadInfo “Y/N”</i> – Specifies whether trace or assertion messages include the thread information. • <i>--timestampInd “Y/N”</i> – Specifies whether trace or assertion messages include the time stamp. • <i>--contextInd “Y/N”</i> – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • <i>--noOfMessages “<number>”</i> – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down. • <i>--period “5/10/20/60/120/480/1440”</i> – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated.

Command	Description	Parameters
		 Note: If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached. <ul style="list-style-type: none"> • <code>--assertAbortInd "Y/N"</code> – Specifies whether the application is to be aborted if the assertion condition fails. ▪ <code>--commandType "changeBufferSize"</code> – Resizes the Light Tracer buffer at run time. • <code>--bufferSize "<number>"</code> – The new size for the Light Tracer buffer.
SET_TRACE_SQL_COMMAND	<p>Activates or stops SQL tracing at runtime. The command name with its parameters must be enclosed in quotation marks. <i>Example:</i> <code>ADJ1_Send_Admin_Command_Sh illin1027 <process instance></code> <code>"SET_TRACE_SQL_COMMAND --traceSqlOn "yes!" --duration "1""</code></p>	<code>--traceSqlOn "<yes or no>" --duration "<minutes>"</code> If the IS_DURATION_REQUIRED environment variable is set to 'Y', the --duration parameter is mandatory. If the duration specified in the command exceeds the allowed maximum value, SQL tracing is activated for the maximum allowed period of time only (60 minutes).  Note: The SHUT_DOWN_CHECK_FREQ parameter of the Light Tracer in the ADJ1_CONF_SECTION_PARAM table determines the frequency (defined in the number of messages) at which the SQL Trace Manager checks whether the period specified in the --duration parameter has passed. Therefore, SQL tracing is stopped after the specified duration or after this number of messages has been reached, whichever comes later.
SWITCH_LIGHT_TRACER_ON ALL	Activates the Light Tracer without the ability to specify the mode or send parameters.	N/A

Command	Description	Parameters
SWITCH_LIGHT_TRACER_OFF ALL	Stops the Light Tracer	N/A
CPF_GracefulShutdownCommand	Shuts down the Usage Extract gracefully	N/A

Examples:

- ADJ1_Send_Admin_Command_Sh hpbl820 US_EX_CY_EVENT 205 SET_LIGHT_TRACER_COMMAND –turnTracer “on” –moduleId “ALL” –threadInfo “y”
- ADJ1_Send_Admin_Command_Sh hpbl820 US_EX_CY_EVENT 205 SWITCH_LIGHT_TRACER_OFF ALL

Recovery Instructions

When the Cycle Event Extract process starts, and the MONITOR_RECOVERY_FLAG parameter is set to ‘Y’ (working in Recovery mode), it searches for records in the APR1_EXTR_MONITOR table with its process name, cycle code, and cycle instance to check whether it needs to work in Recovery mode.

If it finds such records, it gets all the Oracle partitions that are still in the Open (O) status and extracts the event data as usual.

After process or system failure, perform the following:

1. Change the value of the MONITOR_RECOVERY_FLAG parameter to ‘Y’.
2. Rerun the job.

29 ADJ1CPEXT – Cycle Accumulators Extract

This chapter describes Cycle Accumulators Extract process.

Description

The Cycle Accumulators Extract job extracts data of accumulators from the Usage database according to the cycle code, cycle instance, and cycle year. It formats the data according to the format name and writes it to output files. These files are registered in Audit & Control.

Process Type

Batch job

Run Frequency

By request

Activation



Caution: When the home database is up after a failure, the operator must not activate Usage Extracts until the Copy Cycle Usage process, which copies the data from the alternate database to the home database, is complete.

Command Line:	RunJobs ADJ1CPEXT BYREQ
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_CyclePIExtract_Job_Sh
Executable Name:	gcpf1fwcApp

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> US_EX_CY_PI \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

ADJ1_Send_Admin_Command_Sh hpbl820 US_EX_CY_PI 210
CPF_GracefulShutdownCommand

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*

ADJ1CPEXT_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1CPEXT_US_EX_CY_PI210_20081109_130927_1.log

- *Console log files:*

ADJ1_CyclePIExtract_Job_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1_CyclePIExtract_Job_inner_Sh_US_EX_CY_PI210_20081109_130927.log

- *Operational log files:*

ADJ1CPEXT_<Job Record Name>_\${ABP_MARKET}_<Date>_<Time>.log

Example:

ADJ1CPEXT_BYREQ_M3G_20081109_130927.log

- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter)

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information.
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file, and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).

In addition, the console log contains the number of records written to output files. These records are reported per database partition when the process is finished.



Note: The number of records written to the file matches the number of external records and does not match the number of extracted accumulators.

Extract statistics are printed as follows:

EXTR STATISTICS: Total output record created [%d], partition [%s]

where:

- *%d* – The number of records written to files
- *%s* – The name of the database partition
- *Operational log files* – The output of operational scripts.
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Output Files

This section describes the output files.

Name

Output files use the following naming conventions:

- *Regular files*
PIExtracts<*CustomSuffix*><*DateTime*>_<*PID*>_<*ThreadID*>_<*FileNo*>.dat
- *Error files*
 - errPIExtracts<*CustomSuffix*><*DateTime*>_<*PID*>_<*ThreadID*>_<*FileNo*>.dat
 - custerrPIExtracts<*CustomSuffix*><*DateTime*>_<*PID*>_<*ThreadID*>_<*FileNo*>.dat

Location and Format

- *Location* – \$ABP_APP_ROOT/interfaces/output/us_extract.
- *Format* – Semicolon-delimited file. The contents are defined by configuration.

Contents

A regular output file contains event fields specified by configuration, for all extracted records. If an error occurred during processing:

- The regular output file is empty.
- The errPIExtracts output file contains the value of -1.
- If the COLLECT_CYCLE_ERRORS configuration parameter is set to 'Y', the custerrrPIExtracts file contains the IDs of all customers for which an error occurred during data extraction.

Flow

In the Cycle mode, the Cycle Accumulators Extract process (ADJ1CPEXT) performs the following steps:

1. The Process Manager reads the input parameters and divides the work among the working threads.
 2. Each working thread performs the following:
 - a. Accepts the cycle code, instance, and Oracle partition to work on as input.
 - b. Opens the output file.
 - c. Opens a cursor on the Accumulators table using the Extract Engine (an internal component). The cursor extracts the entire contents of the partition:
 - For accumulators, the entire partition is extracted, but it is sorted by the accumulator key. The reason for sorting is to extract the global accumulator with its dimensioned accumulator. The global accumulator is extracted first, followed by all its dimensions, one by one.
 - If an additional WHERE clause was specified, it is added to the cursor statement.
 - d. For each record found in the partition:
 - i. If the extract requires the execution of a mapping case, the working thread activates the relevant mapping case using the code generated by the Implementation Compiler from the implementation made in the Rating Logic Configurator.
 - ii. Formats the record according to the formatting definitions. Records are formatted regardless of whether the mapping case was activated.
- If multiple formats are required, the record is formatted according to each format provided as input. In this case, the output is divided among the output files according to the configured output file names.

If an error occurs during the activation of the mapping case or formatting:

- 1) Does one of the following:
 - ◆ If the COLLECT_CYCLE_ERRORS parameter is set to ‘Y’, proceeds with a dummy run on the partition to collect all errors and writes the respective customer IDs to the custerrPIExtracts file.
 - ◆ If the COLLECT_CYCLE_ERRORS parameter is set to ‘N’, stops extracting the data.
- 2) Marks the partition as failed and exits with failure.
- iii. Writes the record into the output file according to the specified format. If an error occurred during the activation of the mapping case or formatting, the regular output file is truncated, and the errPIExtracts file contains the value of -1.
 - e. When the output file is closed, writes it to Audit & Control.
3. Updates a control table (APR1_EXTR_MONITOR) for each partition that has finished successfully. The table is used for the following:
 - In recovery, the table enables the operator to execute the Usage Extract process only for those partitions that have not been processed successfully.
 - The table enables the operator to verify that the entire process finished successfully, and files can be merged so that there remains just one file per group.

Environment Variables

The Cycle Accumulators Extract uses the following environment variable that is initialized by the op_adj_env_sh script.

Variable Name	Description	Valid Values/ Default Value
APR_CYCLE_XTR_NUM_OF_THREADS	The number of threads that the Extract uses	3

For more information on the script, see the “Amdocs Batch Job Manager Application Script” section in the “Introduction” chapter.

Parameters

The following parameters must be set for the process to run properly.

Parameter	Description	Default Value
APPLICATION_ID	The ID of the job that enables it to get its parameters.	US_EX_CY_PI210
CCF_FILE_NAME	An XML file that contains the relevant configuration.	APR1_Extracts.ccf
EXTR_CYCLE_CODE	The cycle code for which data is to be extracted.	1
EXTR_CYCLE_FROM_DATE	Defines the lower limit of date range (format: year month day, for example, 20081015; time can also be added, for example, 20081015 091558).	N/A
EXTR_CYCLE_INSTANCE	The cycle instance for which data is to be extracted.	1
EXTR_CYCLE_TO_DATE	Defines the upper limit the of date range (format: year month day, for example, 20081015; time can also be added, for example, 20081015 091558).	N/A
EXTR_CYCLE_YEAR	The cycle year for which data is to be extracted.	N/A
EXTR_FORMAT_NAME	The name of the format that is used by the engine to format extracted data. If required, multiple output formats can be specified. In this case, they must be separated by the pipe (' ') character.	UC Charge Format
EXTR_INSTANCE_ID	The instance ID of the job.	1
EXTR_INSTANCES_NUM	The number of instances that run in parallel.	1
EXTR_OUTPUT_DATA_GROUP	The data group of the output file.	PI.1.1.2009 (sample value)
EXTR_OUTPUT_FILE_ALIAS	The alias of the output file. If multiple output formats are required, a list of aliases for the formats requested in the EXTR_FORMAT_NAME parameter must be specified. The file aliases in the list must be separated by the pipe (' ') character.	CyclePI
EXTR_OUTPUT_FILE_NAME	The name of the output file. If multiple output formats are required, a list of output file names for the formats requested in the EXTR_FORMAT_NAME parameter must be specified. The file names in the list must be separated by the pipe (' ') character.	PIExtracts.dat
EXTR_OUTPUT_FILE_UNION	The union of the output file.	DUMMY

Parameter	Description	Default Value
EXTR_OUTPUT_ROUTING_CRITERIA	The routing criteria for the output file.	RPR1_TO_BL1
MAP_MODE	The context (map) in which the process is currently running.	NONE
MONITOR_RECOVERY_FLAG	A flag that indicates whether the Recovery mode is enabled.	N

Configuration Parameters

In Turbo Charging, the configuration parameters (properties) of the Cycle Accumulators Extract process are defined in the database.

The following table shows the performance tuning (configuration) parameters of the Cycle Accumulators Extract process. These parameters are defined in the APR1_CONF_SECTION_PARAM table using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*).



Note: Other parameters must not be changed.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APE	config	QUERY_TRACING_DIR	`\${ABP_APP_LOG}`	The location of trace XML files that show query results.
APE	config	QUERY_TRACING_MODE	FALSE	Indicates whether trace XML files that show query results must be printed.
US_EX_CY_PI	EXTRACT	MONITOR_RECOVERY_FLAG	`\${MONITOR_RECOVERY_FLAG}`	A flag that indicates whether the Recovery mode is enabled.
US_EX_CY_PI	EXTRACT	HINT_CLAUSE	/*+ LEADING (AA) use_hash(AA) FULL (A)*/	The SQL hint used by the Extract Engine (an internal component).
US_EX_CY_PI	EXTRACT_ENGINE	CUST_ADDITIONAL_WHERE_CLAUSE	Note: This parameter is optional.	An additional WHERE clause that can be added to the SELECT statement during customization. The clause must follow proper SQL syntax.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
US_EX_CY_PI	EXTRACT_ENGINE	RATED_EVENT_FLAG	N	Indicates whether information is extracted for events. If this flag is not set, information is extracted for accumulators.
US_EXTRACT	config	COLLECT_CYCLE_ERRORS	N	If an error occurs, indicates whether the process must stop extracting and exit with failure ('N'), or proceed to collect all errors and write the relevant customer IDs to a file ('Y').
US_EXTRACT	config	IS_RUN_MODE_JOB	<code> \${ADJ1_IS_JOB}</code>	Indicates whether the extract is running as a job: <ul style="list-style-type: none">■ <i>Y</i> – Yes■ <i>N</i> – No
US_EXTRACT	config	UQ_MAX_FETCH_BULK_SIZE	100	The maximum number of records to be fetched in one request. This parameter defines the buffer size.
US_EXTRACT	EXTRACT	INSTANCE_ID	<code> \${EXTR_INSTANCE_ID}</code>	The instance ID of the Cycle Accumulators Extract process. This is number of the instance that is running of the total number of instances.
US_EXTRACT	EXTRACT	INSTANCES_NUM	<code> \${EXTR_INSTANCES_NUM}</code>	The number of Cycle Accumulators Extract instances.
US_EXTRACT	EXTRACT	MAP_TYPE	<code> \${MAP_MODE}</code>	The type of the map in which the extract is running.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
US_EXTRACT	EXTRACT	MAX_FETCHED_RECORD	100	The maximum number of records that the Extract Engine (an internal component) can fetch at once. This parameter enables flexible configuration of the bulk size. To set a bulk size that is different from the size specified in the UQ_MAX_FETCH_BULK_SIZE parameter, set this parameter to a lower value than the one defined for the UQ_MAX_FETCH_BULK_SIZE parameter.
US_EXTRACT	EXTRACT	REQUEST_ID	\${REQUEST_ID}	The request ID of the current run. In the Cycle mode, the request ID is a concatenation of the cycle code, instance, and year.
US_EXTRACT	FILE_OUTPUT	FILE_DIRECTORY	\${ABP_APP_ROOT}/interfaces/output/us_extract	The default directory where the output file is created.
US_EXTRACT	FILE_OUTPUT	FILE_NAME	UsageExtractData.dat	The naming convention for the output file.
US_EXTRACT	FILE_OUTPUT	MAX_FILE_SIZE	0	The maximum size of an output file. The value of 0 means unlimited size.
US_EXTRACT	FILE_OUTPUT	MAX_RECORD_NUMBER	10000	The maximum number of records that can be written to a single file.

In addition, the process uses Light Tracer configuration parameters, which are defined in the ADJ1_CONF_SECTION_PARAM table. For more information, see [Light Tracer Parameters](#) in the “[Configuration Parameters](#)” section in the “[ADJ1EVENTSRV – Event Server](#)” chapter.

For each process instance, an entry with the name of the process instance must be created in the APR1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
US_EX_CY_PI210	US_EX_CY_PI

Any parameter to be overridden at the process instance level must be added to the APR1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
US_EX_CY_PI210	FILE_OUTPUT	MAX_RECORD_NUMBER	5000

For more information on parameters and the properties mechanism, see the “[Configuration Parameter Mechanism](#)” appendix.

Configuration Files

The Usage Extract process requires a configuration file (.ccf) that is defined via an internal configuration tool.

The environment variables that appear in the .ccf file and can be customized are defined in the op_adj_env_sh script (for more information, see the “[Environment Variables](#)” section in this chapter).

Database Connections

The process connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	<ul style="list-style-type: none"> ■ AC1_CONTROL ■ APR_EXTR_MONITOR 	Insert Insert, Delete, Update
Customer	<ul style="list-style-type: none"> ■ APE1_SUBSCR_DATA ■ APE1_SUBSCR_OFFERS ■ APE1_SUBSCR_PARAMS 	Select
Usage	APE1_ACCUMULATORS	Select

Admin Commands

The Cycle Accumulators Extract supports the following admin commands. For information on how to execute the commands, see the “Handling Admin Commands” section in the “High Availability” chapter.

Command	Description	Parameters
SET_LIGHT_TRACER_COMMAND	Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters	<ul style="list-style-type: none"> ▪ Activation parameters: <ul style="list-style-type: none"> • <i>turnTracer “on/off”</i> – Activates or stops the Light Tracer • <i>--turnAssert “on/off”</i> – Activates or stops Assertions • <i>--turnLogOnTrace “on/off”</i> – Activates or stops writing log messages in Light Tracer files • <i>--turnFlowTracer/turnAll “on/off”</i> – Activates or stops tracing a single flow or event that is transferred between machines • <i>--turnAll “on/off”</i> – Activates or stops all of the above. The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter). ▪ <i>--moduleID “ALL”</i> ▪ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • <i>--threadInfo “Y/N”</i> – Specifies whether trace or assertion messages include the thread information. • <i>--timestampInd “Y/N”</i> – Specifies whether trace or assertion messages include the time stamp. • <i>--contextInd “Y/N”</i> – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • <i>--noOfMessages “<number>”</i> – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down. • <i>--period “5/10/20/60/120/480/1440”</i> – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated.

Command	Description	Parameters
		 Note: If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached. <ul style="list-style-type: none"> • --assertAbortInd "Y/N" – Specifies whether the application is to be aborted if the assertion condition fails. ▪ --commandType "changeBufferSize" – Resizes the Light Tracer buffer at run time. • --bufferSize "<number>" – The new size for the Light Tracer buffer.
SET_TRACE_SQL_COMMAND	<p>Activates or stops SQL tracing at runtime. The command name with its parameters must be enclosed in quotation marks.</p> <p><i>Example:</i></p> <pre>ADJ1_Send_Admin_Command_Sh illin1027 <process instance> "SET_TRACE_SQL_COMMAND --traceSqlOn \"yes\" --duration \"1\""</pre>	<p>--traceSqlOn "<yes or no>" --duration "<minutes>"</p> <p>If the IS_DURATION_REQUIRED environment variable is set to 'Y', the --duration parameter is mandatory. If the duration specified in the command exceeds the allowed maximum value, SQL tracing is activated for the maximum allowed period of time only (60 minutes).</p>  Note: The SHUT_DOWN_CHECK_FREQ parameter of the Light Tracer in the ADJ1_CONF_SECTION_PARAM table determines the frequency (defined in the number of messages) at which the SQL Trace Manager checks whether the period specified in the --duration parameter has passed. Therefore, SQL tracing is stopped after the specified duration or after this number of messages has been reached, whichever comes later.

Command	Description	Parameters
SWITCH_LIGHT_TRACER_ON ALL	Activates the Light Tracer without the ability to specify the mode or send parameters.	N/A
SWITCH_LIGHT_TRACER_OFF ALL	Stops the Light Tracer	N/A
CPF_GracefulShutdownCommand	Shuts down the Usage Extract gracefully	N/A

Examples:

- `ADJ1_Send_Admin_Command_Sh hpbl820 US_EX_CY_PI 210 SET_LIGHT_TRACER_COMMAND –turnTracer “on” –moduleId “ALL” –threadInfo “y”`
- `ADJ1_Send_Admin_Command_Sh hpbl820 US_EX_CY_PI 210 SWITCH_LIGHT_TRACER_OFF ALL`

Recovery Instructions

When the Cycle Accumulators Extract process starts and the MONITOR_RECOVERY_FLAG parameter is set to ‘Y’ (working in Recovery mode), it searches for records in the APR1_EXTR_MONITOR table with its process name, cycle code, and cycle instance to check whether it needs to work in Recovery mode.

If it finds such records, it gets all the Oracle partitions that are still in the Open (O) status and extracts the event data as usual.

After process or system failure, perform the following:

- Change the value of the MONITOR_RECOVERY_FLAG parameter to ‘Y’.
- Rerun the job.

30 ADJ1CAEXT – Cycle Allowances Extract

This chapter describes Cycle Allowances Extract process.

Description

The Cycle Allowances Extract job extracts data of accumulators from the Usage database according to the cycle code, cycle instance, and cycle year. It formats the data according to the format name and writes it to output files. These files are registered in Audit & Control.

Process Type

Batch job

Run Frequency

By request

Activation



Caution: When the home database is up after a failure, the operator must not activate Usage Extracts until the Copy Cycle Usage process, which copies the data from the alternate database to the home database, is complete.

Command Line:	RunJobs ADJ1CAEXT BYREQ
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_CycleAllowancesExtract_Job_Sh
Executable Name:	gcpf1fwcApp

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> US_EX_CY_ALLOWANCE \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

ADJ1_Send_Admin_Command_Sh hpbl820 US_EX_CY_ALLOWANCE 225
CPF_GracefulShutdownCommand

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*

ADJ1CAEXT_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1CAEXT_US_EX_CY_ALLOWANCE225_20081109_130927_1.log

- *Console log files:*

ADJ1_CycleAllowancesExtract_Job_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1_CycleAllowancesExtract_Job_inner_Sh_US_EX_CY_ALLOWANCE225_20081109_130927.log

- *Operational log files:*

ADJ1CAEXT_<Job Record Name>_\${ABP_MARKET}<Date>_<Time>.log

Example:

ADJ1CAEXT_BYREQ_M3G_20081109_130927.log

- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter)

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information.
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file, and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).

In addition, the console log contains the number of records written to output files. These records are reported per database partition when the process is finished.



Note: The number of records written to the file matches the number of external records and does not match the number of extracted allowances.

Extract statistics are printed as follows:

```
EXTR STATISTICS: Total output record created [%d], partition [%s]
```

where:

- *%d* – The number of records written to files
- *%s* – The name of the database partition
- *Operational log files* – The output of operational scripts.
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Output Files

This section describes the output files.

Name

Output files use the following naming conventions:

- *Regular files*
AIExtracts<CustomSuffix><DateTime>_<PID>_<ThreadID>_<FileNo>.dat
- *Error files*
 - errAIExtracts<CustomSuffix><DateTime>_<PID>_<ThreadID>_<FileNo>.dat
 - custerrAIExtracts<CustomSuffix><DateTime>_<PID>_<ThreadID>_<FileNo>.dat

Location and Format

- *Location* – \$ABP_APP_ROOT/interfaces/output/us_extract.
- *Format* – Semicolon-delimited file. The contents are defined by configuration.

Contents

A regular output file contains event fields specified by configuration, for all extracted records. If an error occurred during processing:

- The regular output file is empty.
- The errAIExtracts output file contains the value of -1.
- If the COLLECT_CYCLE_ERRORS configuration parameter is set to 'Y', the custerrAIExtracts file contains the IDs of all customers for which an error occurred during data extraction.

Flow

In the Cycle mode, the Cycle Allowances Extract process (ADJ1CAEXT) performs the following steps:

1. The Process Manager reads the input parameters and divides the work among the working threads.
2. Each working thread performs the following:
 - a. Accepts the cycle code, instance, and Oracle partition to work on as input.
 - b. Opens the output file.
 - c. Opens a cursor on the Accumulators table using the Extract Engine (an internal component). The cursor extracts the entire contents of the partition:
 - d. For accumulators, the entire partition is extracted, but it is sorted by the accumulator key. The reason for sorting is to extract the global accumulator with its dimensioned accumulator. The global accumulator is extracted first, followed by all its dimensions, one by one.
 - e. If an additional WHERE clause was specified, it is added to the cursor statement.
 - f. For each record found in the partition:
 - i. If the extract requires the execution of a mapping case, the working thread activates the relevant mapping case using the code generated by the Implementation Compiler from the implementation made in the Rating Logic Configurator.
 - ii. Formats the record according to the formatting definitions. Records are formatted regardless of whether the mapping case was activated.If multiple formats are required, the record is formatted according to each format provided as input. In this case, the output is divided among the output files according to the configured output file names.

If an error occurs during the activation of the mapping case or formatting:

- 1) Does one of the following:
 - ◆ If the COLLECT_CYCLE_ERRORS parameter is set to ‘Y’, proceeds with a dummy run on the partition to collect all errors and writes the respective customer IDs to the custerrAIExtracts file.
 - ◆ If the COLLECT_CYCLE_ERRORS parameter is set to ‘N’, stops extracting the data.
 - 2) Marks the partition as failed, and exits with failure.
- iii. Writes the record into the output file according to the specified format. If an error occurred during the activation of the mapping case or formatting, the regular output file is truncated, and the errAIExtracts file contains the value of -1.
- g. When the output file is closed, writes it to Audit & Control.
3. Updates a control table (APR1_EXTR_MONITOR) for each partition that has finished successfully. The table is used for the following:
 - In recovery, the table enables the operator to execute the Usage Extract process only for those partitions that have not been processed successfully.
 - The table enables the operator to verify that the entire process finished successfully, and files can be merged so that there remains just one file per group.

Environment Variables

The Cycle Allowances Extract uses the following environment variable that is initialized by the op_adj_env_sh script.

Variable Name	Description	Valid Values/ Default Value
APR_CYCLE_XTR_NUM_OF_THREADS	The number of threads that the Extract uses	3

For more information on the script, see the “Amdocs Batch Job Manager Application Script” section in the “Introduction” chapter.

Parameters

The following parameters must be set for the process to run properly.

Parameter	Description	Default Value
APPLICATION_ID	The ID of the job that enables it to get its parameters.	US_EX_CY_ALLOWANCE225
CCF_FILE_NAME	An XML file that contains the relevant configuration.	APR1_Extracts.ccf
EXTR_CYCLE_CODE	The cycle code for which data is to be extracted.	1
EXTR_CYCLE_FROM_DATE	Defines the lower limit of date range (format: year month day, for example, 20081015; time can also be added, for example, 20081015 091558).	N/A
EXTR_CYCLE_INSTANCE	The cycle instance for which data is to be extracted.	1
EXTR_CYCLE_TO_DATE	Defines the upper limit the of date range (format: year month day, for example, 20081015; time can also be added, for example, 20081015 091558).	N/A
EXTR_CYCLE_YEAR	The cycle year for which data is to be extracted.	N/A
EXTR_FORMAT_NAME	The name of the format that is used by the engine to format extracted data. If required, multiple output formats can be specified. In this case, they must be separated by the pipe (' ') character.	UC Charge Format
EXTR_INSTANCE_ID	The instance ID of the job.	1
EXTR_INSTANCES_NUM	The number of instances that run in parallel.	1
EXTR_OUTPUT_DATA_GROUP	The data group of the output file.	PI1.1.2009 (sample value)
EXTR_OUTPUT_FILE_ALIAS	The alias of the output file. If multiple output formats are required, a list of aliases for the formats requested in the EXTR_FORMAT_NAME parameter must be specified. The file aliases in the list must be separated by the pipe (' ') character.	CycleAl
EXTR_OUTPUT_FILE_NAME	The file name of the output file. If multiple output formats are required, a list of output file names for the formats requested in the EXTR_FORMAT_NAME parameter must be specified. The file names in the list must be separated by the pipe (' ') character.	ALEExtracts.dat
EXTR_OUTPUT_FILE_UNION	The union of the output file.	DUMMY

Parameter	Description	Default Value
EXTR_OUTPUT_ROUTING_CRITERIA	The routing criteria for the output file.	RPR1_TO_BL1
MAP_MODE	The context (map) in which the process is currently running.	NONE
MONITOR_RECOVERY_FLAG	A flag that indicates whether the Recovery mode is enabled.	N

Configuration Parameters

In Turbo Charging, the configuration parameters (properties) of the Cycle Allowances Extract process are defined in the database.

The following table shows the performance tuning (configuration) parameters of the Cycle Allowances Extract process. These parameters are defined in the APR1_CONF_SECTION_PARAM table using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*).



Note: Other parameters must not be changed.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APE	config	QUERY_TRACING_DIR	\${ABP_APRL_LOG}	The location of trace XML files that show query results.
APE	config	QUERY_TRACING_MODE	FALSE	Indicates whether trace XML files that show query results must be printed.
US_EX_CY_ALLOWANCE	EXTRACT	MONITOR_RECOVERY_FLAG	\${MONITOR_RECOVERY_FLAG}	A flag that indicates whether the Recovery mode is enabled.
US_EX_CY_ALLOWANCE	EXTRACT	HINT_CLAUSE	/*+ LEADING (AA) use_hash(AA) FULL (A)*/	The SQL hint used by the Extract Engine (an internal component).
US_EX_CY_ALLOWANCE	EXTRACT_ENGINE	CUST_ADDITIONAL_WHERE_CLAUSE	 <i>Note: This parameter is optional.</i>	An additional WHERE clause that can be added to the SELECT statement during customization. The clause must follow proper SQL syntax.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
US_EX_CY_ALLOWANCE	EXTRACT_ENGINE	RATED_EVENT_FLAG	N	Indicates whether information is extracted for events. If this flag is not set, information is extracted for accumulators.
US_EXTRACT	config	COLLECT_CYCLE_ERRORS	N	If an error occurs, indicates whether the process must stop extracting and exit with failure ('N'), or proceed to collect all errors and write the relevant customer IDs to a file ('Y').
US_EXTRACT	config	IS_RUN_MODE_JOB	\${ADJ1_IS_JOB}	Indicates whether the extract is running as a job: <ul style="list-style-type: none"> ■ Y – Yes ■ N – No
US_EXTRACT	config	UQ_MAX_FETCH_BUFFER_SIZE	100	The maximum number of records to be fetched in one request. This parameter defines the buffer size.
US_EXTRACT	EXTRACT	INSTANCE_ID	\${EXTR_INSTANCE_ID}	The instance ID of the Cycle Allowance Extract process. This is number of the instance that is running of the total number of instances.
US_EXTRACT	EXTRACT	INSTANCES_NUM	\${EXTR_INSTANCES_NUM}	The number of Cycle Allowances Extract instances.
US_EXTRACT	EXTRACT	MAP_TYPE	\${MAP_MODE}	The type of the map in which the extract is running.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
US_EXTRACT	EXTRACT	MAX_FETCHED_RECORD	100	The maximum number of records that the Extract Engine (an internal component) can fetch at once. This parameter enables flexible configuration of the bulk size. To set a bulk size that is different from the size specified in the UQ_MAX_FETCH_BULK_SIZE parameter, set this parameter to a lower value than the one defined for the UQ_MAX_FETCH_BULK_SIZE parameter.
US_EXTRACT	EXTRACT	REQUEST_ID	\${REQUEST_ID}	The request ID of the current run. In the Cycle mode, the request ID is a concatenation of the cycle code, instance, and year.
US_EXTRACT	FILE_OUTPUT	FILE_DIRECTORY	\${ABP_APP_ROOT}/interfaces/output/us_extract	The default directory where the output file is created.
US_EXTRACT	FILE_OUTPUT	FILE_NAME	UsageExtractData.dat	The naming convention for the output file.
US_EXTRACT	FILE_OUTPUT	MAX_FILE_SIZE	0	The maximum size of an output file. The value of 0 means unlimited size.
US_EXTRACT	FILE_OUTPUT	MAX_RECORD_NUMBER	10000	The maximum number of records that can be written to a single file.

In addition, the process uses Light Tracer configuration parameters, which are defined in the ADJ1_CONF_SECTION_PARAM table. For more information, see [Light Tracer Parameters](#) in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

For each process instance, an entry with the name of the process instance must be created in the APR1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
US_EX_CY_ALLOWANCE225	US_EX_CY_ALLOWANCE

Any parameter to be overridden at the process instance level must be added to the APR1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
US_EX_CY_ALLOWANCE225	FILE_OUTPUT	MAX_RECORD_NUMBER	5000

For more information on parameters and the properties mechanism, see the “Configuration Parameter Mechanism” appendix.

Configuration Files

The Usage Extract process requires a configuration file (.ccf) that is defined via an internal configuration tool.

The environment variables that appear in the .ccf file and can be customized are defined in the op_adj_env_sh script (for more information, see the “Environment Variables” section in this chapter).

Database Connections

The process connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	<ul style="list-style-type: none"> ■ AC1_CONTROL ■ APR_EXTR_MONITOR 	Insert Insert, Delete, Update
Customer	<ul style="list-style-type: none"> ■ APE1_SUBSCR_DATA ■ APE1_SUBSCR_OFFERS ■ APE1_SUBSCR_PARAMS 	Select
Usage	APE1_ACCUMULATORS	Select

Admin Commands

The Cycle Allowances Extract supports the following admin commands. For information on how to execute the commands, see the “Handling Admin Commands” section in the “High Availability” chapter.

Command	Description	Parameters
SET_LIGHT_TRACER_COMMAND	Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters	<ul style="list-style-type: none"> ▪ Activation parameters: <ul style="list-style-type: none"> • <i>turnTracer “on/off”</i> – Activates or stops the Light Tracer • <i>--turnAssert “on/off”</i> – Activates or stops Assertions • <i>--turnLogOnTrace “on/off”</i> – Activates or stops writing log messages in Light Tracer files • <i>--turnFlowTracer/turnAll “on/off”</i> – Activates or stops tracing a single flow or event that is transferred between machines • <i>--turnAll “on/off”</i> – Activates or stops all of the above. ▪ The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter). ▪ <i>--moduleID “ALL”</i> ▪ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • <i>--threadInfo “Y/N”</i> – Specifies whether trace or assertion messages include the thread information. • <i>--timestampInd “Y/N”</i> – Specifies whether trace or assertion messages include the time stamp. • <i>--contextInd “Y/N”</i> – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • <i>--noOfMessages “<number>”</i> – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down. • <i>--period “5/10/20/60/120/480/1440”</i> – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated. <p> Note: If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached.</p>

Command	Description	Parameters
		<ul style="list-style-type: none"> • <code>--assertAbortInd "Y/N"</code> – Specifies whether the application is to be aborted if the assertion condition fails. ▪ <code>--commandType "changeBufferSize"</code> – Resizes the Light Tracer buffer at run time. <ul style="list-style-type: none"> • <code>--bufferSize "<number>"</code> – The new size for the Light Tracer buffer.
SET_TRACE_SQL_COMMAND	<p>Activates or stops SQL tracing at runtime. The command name with its parameters must be enclosed in quotation marks.</p> <p><i>Example:</i> <code>ADJ1_Send_Admin_Command_Sh illin1027 <process instance></code> <code>"SET_TRACE_SQL_COMMAND --traceSqlOn \"yes\" --duration \"1\""</code></p>	<code>--traceSqlOn "<yes or no>" --duration "<minutes>"</code> <p>If the IS_DURATION_REQUIRED environment variable is set to 'Y', the --duration parameter is mandatory. If the duration specified in the command exceeds the allowed maximum value, SQL tracing is activated for the maximum allowed period of time only (60 minutes).</p>  <p>Note: The SHUT_DOWN_CHECK_FREQ parameter of the Light Tracer in the ADJ1_CONF_SECTION_PARAM table determines the frequency (defined in the number of messages) at which the SQL Trace Manager checks whether the period specified in the --duration parameter has passed. Therefore, SQL tracing is stopped after the specified duration or after this number of messages has been reached, whichever comes later.</p>
SWITCH_LIGHT_TRACER_ON ALL	Activates the Light Tracer without the ability to specify the mode or send parameters.	N/A
SWITCH_LIGHT_TRACER_OFF ALL	Stops the Light Tracer	N/A
CPF_GracefulShutdownCommand	Shuts down the Usage Extract gracefully.	N/A

Examples:

- `ADJ1_Send_Admin_Command_Sh hpbl820 US_EX_CY_ALLOWANCE 225 SET_LIGHT_TRACER_COMMAND – turnTracer "on" –moduleId "ALL" –threadInfo "y"`
- `ADJ1_Send_Admin_Command_Sh hpbl820 US_EX_CY_ALLOWANCE 225 SWITCH_LIGHT_TRACER_OFF ALL`

Recovery Instructions

When the Cycle Allowances Extract process starts and the MONITOR_RECOVERY_FLAG parameter is set to ‘Y’ (working in Recovery mode), it searches for records in the APR1_EXTR_MONITOR table with its process name, cycle code, and cycle instance to check whether it needs to work in Recovery mode.

If it finds such records, it gets all the Oracle partitions that are still in the Open (O) status and extracts the event data as usual.

After process or system failure, perform the following:

1. Change the value of the MONITOR_RECOVERY_FLAG parameter to ‘Y’.
2. Rerun the job.

31 ADJ1OCEEXT – Optimized Customer Group Event Extract

This chapter describes the Optimized Customer Group Events Extract process.

Description

The Optimized Customer Group Events Extract process extracts event data for specific customers in an optimized way. The process accepts input files with a list of customers and may perform a full partition scan or an indexed access, depending on the size of the request.

Process Type

Batch job

Run Frequency

By request

Participation in Generic Maps

This section contains information on the generic maps in which the process participates.

Map Name

The Optimized Customer Group Events Extract participates in the following maps:

- Rerun
- QA

For more information about these maps, see *Amdocs Billing Operator Guide*.

Severity Level

Severity Level 2 – The map must hold until the job is fixed.

Activation



Caution: When the home database is up after a failure, the operator must not activate Usage Extracts until the Copy Cycle Usage process, which copies the data from the alternate database to the home database, is complete.

Command Line:	RunJobs ADJ1OCEEXT BYREQ
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_OptimizedCustomerEventExtract_Job_Sh
Executable Name:	gcpf1fwcApp

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> US_EX_OPT_CUST_EVENT \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

ADJ1_Send_Admin_Command_Sh hpbl820 US_EX_OPT_CUST_EVENT 230
CPF_GracefulShutdownCommand

Preceding Processes

The following job must run successfully before the Optimized Customer Group Events Extract process is activated:

- *ADJ1EVGRPLSTN* – Event Group Listener

Dependent Processes

The following processes can be run only after the successful completion of the Optimized Customer Group Events Extract:

- *ADJ1MAS* – Merge and Sort
- *ADJ1EXTCLEANE* – Extract Clean-up

For more information about the Merge and Sort process, see *Amdocs Billing Utilities Run Book*.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*

ADJ1OCEEXT_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1OCEEXT_US_EX_OPT_CUST_EVENT230_20081109_130927_1.log

- *Console log files:*

ADJ1_OptimizedCustomerEventExtract_Job_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1_OptimizedCustomerEventExtract_Job_inner_Sh_US_EX_OPT_CUST_EVENT230_20081109_130927.log

- *Operational log files:*

ADJ1OCEEXT_<Job Record Name>_\${ABP_MARKET}_<Date>_<Time>.log

Example:

ADJ1OCEEXT_BYREQ_M3G_20081109_130927.log

- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter)

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information.
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file, and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).

In addition, the console log contains the number of records written to output files. These records are reported per database partition when the process is finished.



Note: The number of records written to the file matches the number of external records and does not match the number of extracted events.

Extract statistics are printed as follows:

```
EXTR STATISTICS: Total output record created [%d], partition  
[%s]
```

where:

- *%d* – The number of records written to files
- *%s* – The name of the database partition
- *Operational log files* – The output of operational scripts.
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Input Files

This section describes the input files.

Name

The name of input file is taken from Audit & Control database.

Location and Format

This process receives the file from Audit & Control database.

Contents

This file contains information about which customer/subscriber data is to be extracted and in what format this data is to be stored.

Output Files

This section describes the output files.

Name

Output files use the following naming conventions:

- *Regular files*
UsageExtractData<*CustomSuffix*><*DateTime*>_<*PID*>_<*ThreadID*>_<*FileNo*>.dat
- *Error files*
errUsageExtractData<*CustomSuffix*><*DateTime*>_<*PID*>_<*ThreadID*>_<*FileNo*>.dat

Location and Format

- *Location* – \$ABP_APP_ROOT/interfaces/output/us_extract.
- *Format* – Semicolon-delimited file. The contents are defined by configuration.

Contents

A regular output file contains event fields specified by configuration, for all extracted records. If an error occurred during processing:

- The regular output file is empty.
- The errUsageExtractData file contains the value of -1.

Flow

The Optimized Customer Group Events Extract process (ADJ1OCEEXT) performs the following steps:

1. The Process Manager reads all the input Customer Group files and loads them into a temporary table.
 - The process can only start running after the requesting system has published all the groups. This is verified by the order of process execution. First, Amdocs Invoicing runs the process that generates the Customer Group files, and then it executes the Optimized Customer Group Events Extract process.
 - If more than one Optimized Customer Group Events Extract process is executed, each process loads all the customer groups. This is achieved by a feature of Audit & Control, which may be configured to replicate entries for each file to targets in the Audit & Control table.
 - When loading the files into a temporary table, the Process Manager calculates the size of the population that is selected from each rated event partition. Based on this calculation, it determines for each partition whether a full partition scan or an index access must be performed.
2. The Process Manager divides the work among the rated event partitions. Each partition is assigned to one thread.
 - If the number of threads is smaller than the number of partitions, the threads that finish their work first are assigned with additional partitions. This is done until all the partitions have been processed.
 - If more than one process is executed on different partitions, the Process Manager handles only the partitions that it is responsible for.
3. Each working thread performs the following:
 - a. Handles one rated event partition.
 - b. Opens the output file per customer group and rated event partition (it is possible not to open the files until they are required).

- c. Loads the contents of the input file (group ID, customer ID, billing arrangement ID, and subscriber ID, if defined) into a temporary Oracle table. This table is used to retrieve only the requested population from the relevant table through a join.

Before opening the cursor, if the billing arrangement ID is provided in the input file from Amdocs Invoicing, the Extract Engine checks whether the billing arrangement is supported according to the format (defined in *f_formatName*). If at least one external record does not support the billing arrangement, the process generates an error, and the flow stops.
 - d. Opens a cursor that retrieves the required information. When a full table scan is performed, the cursor is opened using a hint of a full table scan. Otherwise, an index access is specified, and a join is performed to select the required population from the current partition.
 - e. For each record:
 - i. If the extract requires the execution of a mapping case, activates the relevant mapping case using the code generated by the Implementation Compiler from the implementation made in the Rating Logic Configurator.
 - ii. Formats the record according to the formatting definitions. Records are formatted regardless of whether the mapping case was activated. If an error occurs during the activation of the mapping case or formatting, the job stops extracting the data, marks the partition as failed, and exits immediately with failure.
 - iii. If the billing arrangement ID is provided in the input file, the Extract Engine checks whether the billing arrangement ID in the record is in the list of the billing arrangements that need to be extracted for the customer. If it is not in the list, the record is dropped.
 - f. Checks the record for the group to which the record belongs and writes it to the relevant file.
 - g. Closes the files at the end of the cursor (which means the end of the partition) and writes them to Audit & Control.
4. Updates a control table (APR1_EXTR_MONITOR) for each partition that has finished successfully. The table is used for the following:
 - In recovery, the table enables the operator to execute the Optimized Customer Group Event Extract process only for those partitions that have not been processed successfully.
 - The table enables the operator to verify that the entire process finished successfully, and files can be merged into one file per group.

After the processes have finished successfully, the results of all the processes are merged into one file per input customer group.

Environment Variables

The Optimized Customer Group Event Extract uses the following environment variable that is initialized by the op_adj_env_sh script.

Variable Name	Description	Valid Values/ Default Value
APR_OPT_EVT_XTR_NUM_OF_THREADS	The number of threads that the Extract uses	3

For more information on the script, see the “Amdocs Batch Job Manager Application Script” section in the “Introduction” chapter.

Parameters

The following parameters must be set for the process to run properly. When the job runs as part of a map, the Turbo Charging Cycle Bill Run script populates these parameters automatically. For more information on this script, see the “ADJ1_AP4_CycleBillRun_Sh – Turbo Charging Cycle Bill Run” chapter.

Parameter	Description	Default Value
APPLICATION_ID	The ID of the job that enables it to get its parameters.	US_EX_OPT_CUST_EVENT230
CCF_FILE_NAME	An XML file that contains the relevant configuration.	APR1_CustEventExtracts Opt.ccf
EXTR_CYCLE_CODE	The cycle code for which data is to be extracted.	1
EXTR_CYCLE_INSTANCE	The cycle instance for which data is to be extracted.	1
EXTR_CYCLE_YEAR	The cycle year for which data is to be extracted.	N/A
EXTR_INPUT_MAP_KEY	The input map key. This key is used as a filter. Only the record with this value appears in the DATA_GROUP field of the AC1_CONTROL table.	EXTRACT
EXTR_INSTANCE_ID	The instance ID of the job.	1
EXTR_INSTANCES_NUM	The number of instances that run in parallel.	1

Parameter	Description	Default Value
MAP_MODE	<p>The context (map) in which the process is currently running. Valid values are:</p> <ul style="list-style-type: none"> ■ NONE (if the job is run manually and not as part of a map) ■ EOC ■ RERUN ■ QA ■ UNDO 	N/A
RUN_MODE	<p>The run mode of the map that is currently running. The value of this parameter is provided by Amdocs Invoicing. Valid values are:</p> <ul style="list-style-type: none"> ■ FULL_RUN ■ RUN+AR ■ EXTRACTS ■ FULL_RERUN ■ RERUN+AR ■ FULL_QA ■ FULL_QAM 	N/A

Configuration Parameters

In Turbo Charging, the configuration parameters (properties) of the Optimized Customer Group Event Extract Post Rerate process are defined in the database.

The following table shows the performance tuning (configuration) parameters of the Optimized Customer Group Event Extract Post Rerate process. These parameters are defined in the APR1_CONF_SECTION_PARAM table using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*).



Note: Other parameters must not be changed.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APE	config	QUERY_TRACING_DIR	\${ABP_APRL_LOG}	The location of trace XML files that show query results.
APE	config	QUERY_TRACING_MODE	FALSE	Indicates whether trace XML files that show query results must be printed.
US_EX_OPT_CUST_EVENT	EV_FILE_OUTPUT	FILE_ALIAS	RTEVT	The default value of the Audit & Control file alias for the output file.
US_EX_OPT_CUST_EVENT	EV_FILE_OUTPUT	ROUTING_CRITERIA	TC1_TO_MAS	The default value of the Audit & Control routing criteria for the output file.
US_EX_OPT_CUST_EVENT	EXTRACT	ADDITIONAL_HINT_CLAUSE	/* */	The SQL hint used by the Extract Engine (an internal component) when the number of subscribers is less than the number defined by the MAX_SUBS_NUMBER parameter.
US_EX_OPT_CUST_EVENT	EXTRACT	HINT_CLAUSE	/*+ full (E) full(ec) use_hash (ec, E) leading(ec) */	The SQL hint used by the Extract Engine (an internal component).
US_EX_OPT_CUST_EVENT	EXTRACT_ENGINE	MAX_SUBS_NUMBER	1000	The maximum number of subscribers that can be extracted per Oracle partition without a full table scan. The value of 0 means unlimited number.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
US_EX_OPT_CUST_EVENT	EXTRACT_ENGINE	RATED_EVENT_FLAG	Y	Indicates whether information is extracted for events. If this flag is not set, information is extracted for accumulators.
US_EX_OPT_CUST_EVENT	FILE_OUTPUT	DATA_GROUP	NA	The default value of the Audit & Control data group for the output file.
US_EXTRACT	config	IS_RUN_MODE_JOB	\${ADJ1_IS_JOB}	Indicates whether the current extract is running as a job: <ul style="list-style-type: none"> ■ Y – Yes ■ N – No
US_EXTRACT	config	UQ_MAX_FETCH_BULK_SIZE	100	The maximum number of records to be fetched in one request. This parameter defines the buffer size. For performance reasons, it is recommended that the operator increase the value of this parameter to at least 1000 to reduce the number of fetches.
US_EXTRACT	EXTRACT	DL_TEMP_TABLE_BUFF_SIZE	1000	The size of the record bulk inserted into a temporary table with customer data.
US_EXTRACT	EXTRACT	GET_NETWORK_FILES_INDICATOR_TIMEOUT	180 sec	The period of time for which the Extract waits for a reply from the Audit & Control Manager about files available in the network.
US_EXTRACT	EXTRACT	GET_NETWORK_FILES_MAX_TIMES	5	The number of times that the Extract tries to retrieve files from the network if the response from the Audit & Control Manager indicates that there are such files available. If no files are received after this number of attempts, the Extract stops looking for files over the network.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
US_EXTRACT	EXTRACT	GETTING_FILES_OVER_NETWORK_MODE	Y	<p>Determines the behavior of the Extract in a distributed environment if Amdocs Invoicing and the Extract are on different machines:</p> <ul style="list-style-type: none"> ■ <i>N</i> – When there are no more files, the Extract shuts down. ■ <i>Y</i> – Before shutting down, the Extract checks whether relevant files are available in the network. If they do, it waits for a period of time determined by the TIME_TO_WAIT_AFTER_NETWORK_FILES_IND parameter and then tries to get these files before shutting down.
US_EXTRACT	EXTRACT	INITIALIZATION_SLEEP	600	The number of seconds that the process sleeps before starting.
US_EXTRACT	EXTRACT	INSTANCE_ID	1	The instance ID of the Optimized Customer Group Event Extract process. This is number of the instance that is running of the total number of instances.
US_EXTRACT	EXTRACT	INSTANCES_NUM	1	The number of Optimized Customer Group Event Extract instances.
US_EXTRACT	EXTRACT	MAP_TYPE	\${MAP_MODE}	The type of the map in which the extract is running.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
US_EXTRACT	EXTRACT	MAX_FETCHED_RECORD	100	<p>The maximum number of records that the Extract Engine (an internal component) can fetch at once. For performance reasons, it is recommended that the operator increase the value of this parameter to at least 1000 to reduce the number of fetches.</p> <p>This parameter enables flexible configuration of the bulk size. To set a bulk size that is different from the size specified in the UQ_MAX_FETCH_BULK_SIZE parameter, set this parameter to a lower value than the one defined for the UQ_MAX_FETCH_BULK_SIZE parameter.</p>
US_EXTRACT	EXTRACT	MAX_IDLE_TIME	600	The maximum number of seconds that the process can be idle.
US_EXTRACT	EXTRACT	REQUEST_ID	\${REQUEST_ID}	The request ID of the current run.
US_EXTRACT	EXTRACT	TIME_TO_WAIT_AFTER_NET_WORK_FILES_IND	5 sec	The period of time for which the Extract waits between getting a response from Audit & Control Manager that there are files available in the network and trying to retrieve those files. This parameter is used in a distributed environment if the GETTING_FILES_OVER_NETWORK_MODE parameter is ‘Y’.
US_EXTRACT	FILE_OUTPUT	FILE_DIRECTORY	\${ABP_APP_ROOT}/interfaces/output/us_extract	The default directory where the output file is created.
US_EXTRACT	FILE_OUTPUT	FILE_NAME	UsageExtractData.dat	The naming convention for the output file.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
US_EXTRACT	FILE_OUTPUT	MAX_FILE_SIZE	0	<p>The maximum size of an output file. The value of 0 means unlimited size.</p>  <p><i>Note: The Optimized Customer Group Event Extract does not take this parameter into consideration even if the parameter exists in the database. If an error message about this parameter appears in the log, it can be ignored.</i></p>
US_EXTRACT	FILE_OUTPUT	MAX_RECORD_NUMBER	100	<p>The maximum number of records that can be written to a single file.</p>  <p><i>Note: The Optimized Customer Group Event Extract does not take this parameter into consideration even if the parameter exists in the database. If an error message about this parameter appears in the log, it can be ignored.</i></p>

In addition, the process uses Light Tracer configuration parameters, which are defined in the ADJ1_CONF_SECTION_PARAM table. For more information, see [Light Tracer Parameters](#) in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

For each process instance, an entry with the name of the process instance must be created in the APR1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
US_EX_OPT_CUST_EVENT230	US_EX_OPT_CUST_EVENT

Any parameter to be overridden at the process instance level must be added to the APR1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
US_EX_OPT_CUST_EVENT230	FILE_OUTPUT	MAX_RECORD_NUMBER	150

For more information on parameters and the properties mechanism, see the “Configuration Parameter Mechanism” appendix.

Configuration Files

The Usage Extract process requires a configuration file (.ccf) that is defined via an internal configuration tool.

The environment variables that appear in the .ccf file and can be customized are defined in the op_adj_env_sh script (for more information, see the “Environment Variables” section in this chapter).

Database Connections

The process connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	<ul style="list-style-type: none"> ■ APR1_CUSTOMER_EX ■ AC1_CONTROL ■ APR_EXTR_MONITOR 	<ul style="list-style-type: none"> ■ Insert ■ Insert ■ Insert, Delete, Update
Customer	<ul style="list-style-type: none"> ■ APE1_SUBSCR_DATA ■ APE1_SUBSCR_OFFERS ■ APE1_SUBSCR_PARAMS 	Select
Usage	APE1_RATED_EVENT	Select

Admin Commands

The Optimized Customer Group Extract supports the following admin commands. For information on how to run admin commands, see the “[Handling Admin Commands](#)” section in the “[High Availability](#)” chapter.

Command	Description	Parameters
SET_LIGHT_TRACER_COMMAND	Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters	<ul style="list-style-type: none"> ■ Activation parameters: XIX. <i>turnTracer “on/off”</i> – Activates or stops the Light Tracer <ul style="list-style-type: none"> • <i>--turnAssert “on/off”</i> – Activates or stops Assertions • <i>--turnLogOnTrace “on/off”</i> – Activates or stops writing log messages in Light Tracer files • <i>--turnFlowTracer/turnAll “on/off”</i> – Activates or stops tracing a single flow or event that is transferred between machines • <i>--turnAll “on/off”</i> – Activates or stops all of the above. The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter). ■ <i>--moduleID “ALL”</i> ■ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • <i>--threadInfo “Y/N”</i> – Specifies whether trace or assertion messages include the thread information. • <i>--timestampInd “Y/N”</i> – Specifies whether trace or assertion messages include the time stamp. • <i>--contextInd “Y/N”</i> – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • <i>--noOfMessages “<number>”</i> – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down. • <i>--period “5/10/20/60/120/480/1440”</i> – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated.

Command	Description	Parameters
		 Note: If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached. <ul style="list-style-type: none"> • --assertAbortInd "Y/N" – Specifies whether the application is to be aborted if the assertion condition fails. ▪ --commandType "changeBufferSize" – Resizes the Light Tracer buffer at run time. <ul style="list-style-type: none"> • --bufferSize "<number>" – The new size for the Light Tracer buffer.
SET_TRACE_SQL_COMMAND	<p>Activates or stops SQL tracing at runtime. The command name with its parameters must be enclosed in quotation marks.</p> <p><i>Example:</i></p> <pre>ADJ1_Send_Admin_Command_Sh illin1027 <process instance> "SET_TRACE_SQL_COMMAND --traceSqlOn \"yes\" --duration \"1\""</pre>	--traceSqlOn "<yes or no>" --duration "<minutes>" If the IS_DURATION_REQUIRED environment variable is set to 'Y', the --duration parameter is mandatory. If the duration specified in the command exceeds the allowed maximum value, SQL tracing is activated for the maximum allowed period of time only (60 minutes).  Note: The SHUT_DOWN_CHECK_FREQ parameter of the Light Tracer in the ADJ1_CONF_SECTION_PARAM table determines the frequency (defined in the number of messages) at which the SQL Trace Manager checks whether the period specified in the --duration parameter has passed. Therefore, SQL tracing is stopped after the specified duration or after this number of messages has been reached, whichever comes later.

Command	Description	Parameters
SWITCH_LIGHT_TRACER_ON ALL	Activates the Light Tracer without the ability to specify the mode or send parameters.	N/A
SWITCH_LIGHT_TRACER_OFF ALL	Stops the Light Tracer	N/A
CPF_GracefulShutdownCommand	Shuts down the Usage Extract gracefully	N/A

Examples:

- `ADJ1_Send_Admin_Command_Sh hpbl820 US_EX_OPT_CUST_EVENT 230 SET_LIGHT_TRACER_COMMAND – turnTracer “on” –moduleId “ALL” –threadInfo “y”`
- `ADJ1_Send_Admin_Command_Sh hpbl820 US_EX_OPT_CUST_EVENT 230 SWITCH_LIGHT_TRACER_OFF ALL`

Recovery Instructions

When the Optimized Customer Group Event Extract process starts, it searches for records in the APR1_EXTR_MONITOR table with its process name, cycle code, and cycle instance to check whether it needs to work in Recovery mode.

First, the job gets the IU (In Use) files and then processes the Ready ones from the AC1_CONTROL table.

32 ADJ1POCEEXT – Optimized Customer Group Event Extract Post Rerate

This chapter describes the Optimized Customer Group Events Extract Post Rerate process.

Description

The Optimized Customer Group Events Extract Post Rerate process extracts rerated event data for specific customers in an optimized way. The process accepts input files with a list of customers and may perform a full partition scan or an indexed access, depending on the size of the request. This process has nearly the same input parameters and executes the same flow as the non-rerate extract; however, it listens to a different file alias. For more information, see the “ADJ1OCEEXT – Optimized Customer Group Event Extract” chapter.

Process Type

Batch job

Run Frequency

By request

Participation in Generic Maps

This section contains information on the generic maps in which the process participates.

Map Name

End-of-Cycle (EOC)

For more information about this map, see *Amdocs Billing Operator Guide*.

Severity Level

Severity Level 2 – The map must hold until the job is fixed.

Activation



Caution: *When the home database is up after a failure, the operator must not activate Usage Extracts until the Copy Cycle Usage process, which copies the data from the alternate database to the home database, is complete.*

Command Line:	RunJobs ADJ1POCEEXT BYREQ
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_OptimizedCustomerEventExtract_Job_Sh
Executable Name:	gcpf1fwcApp

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> US_EX_OPT_CUST_EVENT \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

ADJ1_Send_Admin_Command_Sh hpbl820 US_EX_OPT_CUST_EVENT 230
CPF_GracefulShutdownCommand

Preceding Processes

The following job must run successfully before the Optimized Customer Group Events Extract Post Rerate process is activated:

- *ADJ1EVGRPLSTN* – Event Group Listener

Dependent Processes

The following processes can be run only after the successful completion of the Optimized Customer Group Events Extract Post Rerate:

- *ADJ1MAS* – Merge and Sort
- *ADJ1EXTCLEANE* – Extract Clean-up (in the End-of-Cycle map)

For more information about the Merge and Sort process, see *Amdocs Billing Utilities Run Book*.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*

ADJ1POCEEXT_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1POCEEXT_US_EX_OPT_CUST_EVENT230_20081109_130927_1.log

- *Console log files:*

ADJ1_OptimizedCustomerEventExtract_Job_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1_OptimizedCustomerEventExtract_Job_inner_Sh_US_EX_OPT_CUST_EV
ENT230_20081109_130927.log

- *Operational log files:*

ADJ1POCEEXT_<Job Record Name>_\${ABP_MARKET}_<Date>_<Time>.log

Example:

ADJ1POCEEXT_BYREQ_M3G_20081109_130927.log

- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter)

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information.
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file, and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).

In addition, the console log contains the number of records written to output files. These records are reported per database partition when the process is finished.



Note: The number of records written to the file matches the number of external records and does not match the number of extracted events.

Extract statistics are printed as follows:

```
EXTR STATISTICS: Total output record created [%d], partition  
[%s]
```

where:

- *%d* – The number of records written to files
- *%s* – The name of the database partition
- *Operational log files* – The output of operational scripts.
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Input Files

This section describes the input files.

Name

The name of input file is taken from Audit & Control database.

Location and Format

This process receives the file from Audit & Control database.

Contents

This file contains information about which customer/subscriber data is to be extracted and in what format this data is to be stored.

Output Files

This section describes the output files.

Name

Output files use the following naming conventions:

- *Regular files*
UsageExtractData<*CustomSuffix*><*DateTime*>_<*PID*>_<*ThreadID*>_<*FileNo*>.dat
- *Error files*
errUsageExtractData<*CustomSuffix*><*DateTime*>_<*PID*>_<*ThreadID*>_<*FileNo*>.dat

Location and Format

- *Location* – \$ABP_APP_ROOT/interfaces/output/us_extract.
- *Format* – Semicolon-delimited file. The contents are defined by configuration.

Contents

A regular output file contains event fields specified by configuration, for all extracted records. If an error occurred during processing:

- The regular output file is empty.
- The errUsageExtractData file contains the value of -1.

Parameters

The process takes the same input parameters as the Optimized Customer Group Event Extract (ADJ1OCEEXT). In addition, the following parameter must be set for the process to run properly.

Parameter	Description	Default Value
EXTR_AFTER_RERATE_FLAG	Indicates whether the process is running after rerate	Y

33 ADJ1RCPEXT – Customer Group Accumulators Extract

This chapter describes the Customer Group Accumulators Extract process.

Description

The Customer Group Accumulators Extract process extracts accumulator data for specific customers.

Process Type

Batch job

Run Frequency

By request

Participation in Generic Maps

This section contains information on the generic maps in which the process participates.

Map Name

- End-of-Cycle (EOC)
- Rerun

For more information about these maps, see *Amdocs Billing Operator Guide*.

Severity Level

Severity Level 2 – The map must hold until the job is fixed.

Activation



Caution: When the home database is up after a failure, the operator must not activate Usage Extracts until the Copy Cycle Usage process, which copies the data from the alternate database to the home database, is complete.

Command Line:	RunJobs ADJ1RCPEXT BYREQ
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1-RegularCustomerPIExtract_Job_Sh
Executable Name:	gcpf1fwcApp

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> US_EX_REG_CUST_PI \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

ADJ1_Send_Admin_Command_Sh hpbl820 US_EX_REG_CUST_PI 245
CPF_GracefulShutdownCommand

Preceding Processes

The following jobs must run successfully before this process is activated:

- In the End-of-Cycle map:
ADJ1RRPOPREP – Rerate Population Report
- In the Rerun map:
None. It runs in parallel to other Usage Extracts that participate in the map.

Dependent Processes

The following process can be run only after the successful completion of the Customer Group Accumulators Extract:

- *ADJ1EXTCLEANE* – Extract Clean-up

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*

ADJ1RCPEXT_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1RCPEXT_US_EX_REG_CUST_PI245_20081109_130927_1.log

- *Console log files:*

ADJ1-RegularCustomerPIExtract_Job_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1-RegularCustomerPIExtract_Job_inner_Sh_US_EX_REG_CUST_PI245_2
0081109_130927.log

- *Operational log files:*

ADJ1RCPEXT_<Job Record Name>_\${ABP_MARKET}_<Date>_<Time>.log

Example:

ADJ1RCPEXT_BYREQ_M3G_20081109_130927.log

- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter)

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information.
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file, and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).

In addition, the console log contains the number of records written to output files. These records are reported per database partition when the process is finished.



Note: The number of records written to the file matches the number of external records and does not match the number of extracted accumulators.

Extract statistics are printed as follows:

```
EXTR STATISTICS: Total output record created [%d], partition  
[%s]
```

where:

- *%d* – The number of records written to files
- *%s* – The name of the database partition
- *Operational log files* – The output of operational scripts.
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Input Files

This section describes the input files.

Name

The name of input file is taken from Audit & Control database.

Location and Format

This process receives the file from Audit & Control database.

Contents

This file contains information about the customer/subscriber whose data is to be extracted and in what format this data is to be stored.

Output Files

This section describes the output files.

Name

Output files use the following naming conventions:

- *Regular files*
UsageExtractData<*CustomSuffix*><*DateTime*>_<*PID*>_<*ThreadID*>_<*FileNo*>.dat
- *Error files*
errUsageExtractData<*CustomSuffix*><*DateTime*>_<*PID*>_<*ThreadID*>_<*FileNo*>.dat

Location and Format

- *Location* – \$ABP_APP_ROOT/interfaces/output/us_extract
- *Format* – Semicolon-delimited file. The contents are defined by configuration.

Contents

A regular output file contains event fields specified by configuration, for all extracted records. If an error occurred during processing:

- The regular output file is empty.
- If the MAX_FAILED_CUST parameter is not set (default value is -1), or if the number of erroneous customers is greater than the value of this parameter, the errUsageExtractData file contains the value of -1. Otherwise, the file contains the IDs of all erroneous customers.

Flow

In the Customer mode, the Customer Group Accumulator Extract process (ADJ1RCPEXT) performs the following steps:

- Reads the next ready Customer Group file from Audit & Control. If several processes are running on different machines, the servers use a shared disk so that the input file is available for all the processes.
- Opens the output file.
- Loads the contents of the input file (group ID, customer ID, billing arrangement ID, and subscriber ID, if defined) into a temporary Oracle table. This table is used to retrieve only the requested population from the relevant table through a join.

Before opening the cursor, if the billing arrangement ID is provided in the input file from Amdocs Invoicing, the Extract engine checks whether the billing arrangement is supported according to the format (defined in *f_formatName*). If at least one external record does not support the billing arrangement, the process generates an error, and the flow stops.

- Retrieves the required information in a loop from partition to partition from the Accumulators table.
- Opens a cursor using the Extract Engine (an internal component). The cursor joins the temporary table with the Accumulators table.

Information is sorted by customer and subscriber.

For accumulators, information is also sorted by accumulator key so that the global accumulators are selected before the dimensioned accumulators.

- a. For each customer, the process flushes the output into the file and writes the last customer ID that was flushed into a control table. This information may be used in recovery so that the group is extracted starting only from the last processed customer.
- b. For each record:
 - i. If the extract requires the execution of a mapping case, activates the relevant mapping case using the code generated by the Implementation Compiler from the implementation made in the Rating Logic Configurator.

- ii. Formats the record according to the formatting definitions. Records are formatted regardless of whether the mapping case was activated. If an error occurs during the activation of the mapping case or formatting:
 - ◆ If the MAX_FAILED_CUST parameter is not defined or set to -1, or if the number of erroneous customers written to the error output file is greater than defined in this parameter, stops extracting.
 - ◆ If the value of the MAX_FAILED_CUST parameter is greater than 0, and if the number of erroneous customers has not yet exceeded this value, continues writing the IDs of all erroneous customers into the error output file.
- iii. If the billing arrangement ID is provided in the input file, the Extract engine checks whether the billing arrangement ID in the record is in the list of the billing arrangements that need to be extracted for the customer .If it is not in the list, the record is dropped.
- c. Writes the record into the output file. If an error occurred during processing, the regular output file is truncated. The contents of the errUsageExtractsData file depend on the value of the MAX_FAILED_CUST parameter, as described in the “Output Files” section in this chapter.
 - Closes the file at the end of the group and writes it to Audit & Control.
 - Updates a control table (APR1_EXTR_CUSTOMER_RECOVERY) to indicate that the process has finished successfully.

Environment Variables

The Customer Group Accumulators Extract uses the following environment variable that is initialized by the op_adj_env_sh script.

Variable Name	Description	Valid Values/ Default Value
APR_CUST_PI_XTR_NUM_OF_INSTS	The number of threads that the Extract uses	3

For more information on the script, see the “Amdocs Batch Job Manager Application Script” section in the “Introduction” chapter.

Parameters

The following parameters must be set for the process to run properly. When the job runs as part of a map, the Turbo Charging Cycle Bill Run script populates these parameters automatically. For more information on this script, see the “ADJ1_AP4_CycleBillRun_Sh – Turbo Charging Cycle Bill Run” chapter.

Parameter	Description	Default Value
APPLICATION_ID	The ID of the job that enables it to get its parameters.	US_EX_REG_CUST_PI245
CCF_FILE_NAME	An XML file that contains the relevant configuration.	APR1_CustPIExtractsReg.ccf

Parameter	Description	Default Value
EXTR_CYCLE_CODE	The cycle code for which data is to be extracted.	1
EXTR_CYCLE_INSTANCE	The cycle instance for which data is to be extracted.	1
EXTR_CYCLE_YEAR	The cycle year for which data is to be extracted.	N/A
EXTR_INPUT_MAP_KEY	The input map key. This key is used as a filter. Only the record with this value appears in the DATA_GROUP field of the AC1_CONTROL table.	EXTRACT
EXTR_INSTANCE_ID	The instance ID of the job.	1
EXTR_INSTANCES_NUM	The number of instances that run in parallel.	1
MAP_MODE	The context (map) in which the process is currently running. Valid values are: <ul style="list-style-type: none"> ■ NONE (if the job is run manually and not as part of a map) ■ EOC ■ RERUN ■ QA ■ UNDO 	N/A
RUN_MODE	The run mode of the map that is currently running. The value of this parameter is provided by Amdocs Invoicing. Valid values are: <ul style="list-style-type: none"> ■ FULL_RUN ■ RUN+AR ■ EXTRACTS ■ FULL_RERUN ■ RERUN+AR ■ FULL_QA ■ FULL_QAM 	N/A

Configuration Parameters

In Turbo Charging, the configuration parameters (properties) of the Customer Group Accumulators Extract process are defined in the database.

The following table shows the performance tuning (configuration) parameters of the Customer Group Accumulators Extract process. These parameters are defined in the APR1_CONF_SECTION_PARAM table using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*).



Note: Other parameters must not be changed.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APE	config	QUERY_TRACING_DIR	\${ABP_APP_LOG}	The location of trace XML files that show query results.
APE	config	QUERY_TRACING_MODE	FALSE	Indicates whether trace XML files that show query results must be printed.
US_EX_REG_CUST_PI	EXTRACT	CUST_RECOVERY_MODE	Y	Indicates whether the Customer Recovery mode is enabled.
US_EX_REG_CUST_PI	EXTRACT	EXIT_EVENT_NAME	PIXTRFNSH	The name of the expected trigger.
US_EX_REG_CUST_PI	EXTRACT	MAX_CUST_NUM	10	The number of customers that are processed before writing the data into the output file. This parameter is valid if the Customer Recovery mode is enabled. It is recommended that the operator increase the value of this parameter to at least 100 to reduce the number of times that the database must be accessed.
US_EX_REG_CUST_PI	EXTRACT_ENGINE	RATED_EVENT_FLAG	N	Indicates whether information is extracted for events. If this flag is not set, information is extracted for accumulators.
US_EX_REG_CUST_PI	FILE_OUTPUT	DATA_GROUP	NA	The default value of the Audit & Control data group for the output file.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
US_EX_REG_CUST_PI	PI_FILE_OUTPUT	FILE_ALIAS	RTPI	The default value of the Audit & Control file alias for the output file.
US_EXTRACT	config	IS_RUN_MODE_JOB	\${ADJ1_IS_JOB} }	Indicates whether the current extract is running as a job: <ul style="list-style-type: none"> ■ Y – Yes ■ N – No
US_EXTRACT	config	UQ_MAX_FETCH_BULK_SIZE	100	The maximum number of records to be fetched in one request. This parameter defines the buffer size. For performance reasons, it is recommended that the operator increase the value of this parameter to at least 1000 to reduce the number of fetches.
US_EXTRACT	EXTRACT	DL_TEMP_TABLE_BUFF_SIZE	1000	The size of the record bulk inserted into a temporary table with customer data.
US_EXTRACT	EXTRACT	GET_NETWORK_FILES_INDICATOR_TIMEOUT	180 sec	The period of time for which the Extract waits for a reply from the Audit & Control Manager about files available in the network.
US_EXTRACT	EXTRACT	GET_NETWORK_FILES_MAX_TIMES	5	The number of times that the Extract tries to retrieve files from the network if the response from the Audit & Control Manager indicates that there are such files available. If no files are received after this number of attempts, the Extract stops looking for files over the network.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
US_EXTRACT	EXTRACT	GETTING_FILES_OVER_NETWORK_MODE	Y	<p>Determines the behavior of the Extract in a distributed environment if Amdocs Invoicing and the Extract are on different machines:</p> <ul style="list-style-type: none"> ■ <i>N</i> – When there are no more files, the Extract shuts down. ■ <i>Y</i> – Before shutting down, the Extract checks whether relevant files are available in the network. If they do, it waits for a period of time determined by the TIME_TO_WAIT_AFTER_NETWORK_FILES_IND parameter and then tries to get these files before shutting down.
US_EXTRACT	EXTRACT	INITIALIZATION_SLEEP	600	The number of seconds for which the process sleeps before initialization.
US_EXTRACT	EXTRACT	INSTANCE_ID	<code> \${EXTR_INSTANCE_ID} </code>	The instance ID of the Customer Group Accumulators Extract process. This is number of the instance that is running of the total number of instances.
US_EXTRACT	EXTRACT	INSTANCES_NUM	<code> \${EXTR_INSTANCES_NUM} </code>	The number of Customer Group Accumulators Extract instances.
US_EXTRACT	EXTRACT	MAP_TYPE	<code> \${MAP_MODE} </code>	The type of the map in which the extract is running.
US_EXTRACT	EXTRACT	MAX_FAILED_CUST	-1	<p>The maximum number of customers for whom the extract can fail before the entire group is marked as failed.</p> <p>The default value of '-1' means that the threshold is disabled.</p>

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
US_EXTRACT	EXTRACT	MAX_FETCHED_RECORD	100	<p>The maximum number of records that the Extract Engine (an internal component) can fetch at once. For performance reasons, it is recommended that the operator increase the value of this parameter to at least 1000 to reduce the number of fetches.</p> <p>This parameter enables flexible configuration of the bulk size. To set a bulk size that is different from the size specified in the UQ_MAX_FETCH_BULK_SIZE parameter, set this parameter to a lower value than the one defined for the UQ_MAX_FETCH_BULK_SIZE parameter.</p>
US_EXTRACT	EXTRACT	MAX_IDLE_TIME	600	The maximum number of seconds that the process can be idle.
US_EXTRACT	EXTRACT	REQUEST_ID	\${REQUEST_ID} }	The request ID of the current run.
US_EXTRACT	EXTRACT	TIME_TO_WAIT_AFTER_NETWORK_FILES_IND	5 sec	The period of time for which the Extract waits between getting a response from Audit & Control Manager that there are files available in the network and trying to retrieve those files. This parameter is used in a distributed environment if the GETTING_FILES_OVER_NETWORK_MODE parameter is ‘Y’.
US_EXTRACT	EXTRACT_ENGINE	MAX_SUBS_NUMBER	0	The maximum number of subscribers that can be extracted per Oracle partition without a full table scan. The value of 0 means unlimited number.
US_EXTRACT	FILE_OUTPUT	FILE_DIRECTORY	\${ABP_APP_ROOT}/interfaces/output/us_extract	The default directory where the output file is created.
US_EXTRACT	FILE_OUTPUT	FILE_NAME	UsageExtractData.dat	The naming convention for the output file.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
US_EXTRACT	FILE_OUTPUT	MAX_FILE_SIZE	0	<p>The maximum size of an output file. The value of 0 means unlimited size.</p>  <p><i>Note: The Customer Group Accumulators Extract does not take this parameter into consideration even if the parameter exists in the database. If an error message about this parameter appears in the log, it can be ignored.</i></p>
US_EXTRACT	FILE_OUTPUT	MAX_RECORD_NUMBER	100	<p>The maximum number of records that can be written to a single file.</p>  <p><i>Note: The Customer Group Accumulators Extract does not take this parameter into consideration even if the parameter exists in the database. If an error message about this parameter appears in the log, it can be ignored.</i></p>
US_EXTRACT	PI_FILE_OUTPUT	ROUTING_CRITERIA	BL1_TO_RPR1	The default value of the Audit & Control routing criteria for the output file.

In addition, the process uses Light Tracer configuration parameters, which are defined in the ADJ1_CONF_SECTION_PARAM table. For more information, see [Light Tracer Parameters](#) in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

For each process instance, an entry with the name of the process instance must be created in the APR1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
US_EX_REG_CUST_PI245	US_EX_REG_CUST_PI

Any parameter to be overridden at the process instance level must be added to the APR1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
US_EX_REG_CUST_PI245	FILE_OUTPUT	MAX_RECORD_NUMBER	150

For more information on parameters and the properties mechanism, see the “Configuration Parameter Mechanism” appendix.

Configuration Files

The Usage Extract process requires a configuration file (.ccf) that is defined via an internal configuration tool.

The environment variables that appear in the .ccf file and can be customized are defined in the op_adj_env_sh script (for more information, see the “Environment Variables” section in this chapter).

Database Connections

The process connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	<ul style="list-style-type: none"> ■ APR1_CUSTOMER_EX ■ AC1_CONTROL ■ APR_EXTR_MONITOR ■ APR1_OP_MAP_EVENTS 	<ul style="list-style-type: none"> ■ Insert ■ Insert ■ Insert, Delete, Update ■ Select
Customer	<ul style="list-style-type: none"> ■ APE1_SUBSCR_DATA ■ APE1_SUBSCR_OFFERS ■ APE1_SUBSCR_PARAMS 	Select
Usage	APE1_ACCUMULATORS	Select

Admin Commands

The Customer Group Accumulators Extract process supports the following admin commands. For information on how to execute the commands, see the “Handling Admin Commands” section in the “High Availability” chapter.

Command	Description	Parameters
SET_LIGHT_TRACER_COMMAND	Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters	<ul style="list-style-type: none"> ▪ Activation parameters: <ul style="list-style-type: none"> • <i>turnTracer “on/off”</i> – Activates or stops the Light Tracer • <i>--turnAssert “on/off”</i> – Activates or stops Assertions • <i>--turnLogOnTrace “on/off”</i> – Activates or stops writing log messages in Light Tracer files • <i>--turnFlowTracer/turnAll “on/off”</i> – Activates or stops tracing a single flow or event that is transferred between machines • <i>--turnAll “on/off”</i> – Activates or stops all of the above. The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter). ▪ <i>--moduleID “ALL”</i> ▪ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • <i>--threadInfo “Y/N”</i> – Specifies whether trace or assertion messages include the thread information. • <i>--timestampInd “Y/N”</i> – Specifies whether trace or assertion messages include the time stamp. • <i>--contextInd “Y/N”</i> – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • <i>--noOfMessages “<number>”</i> – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down. • <i>--period “5/10/20/60/120/480/1440”</i> – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated.

Command	Description	Parameters
		 Note: If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached. <ul style="list-style-type: none"> • <code>--assertAbortInd "Y/N"</code> – Specifies whether the application is to be aborted if the assertion condition fails. ▪ <code>--commandType "changeBufferSize"</code> – Resizes the Light Tracer buffer at run time. • <code>--bufferSize "<number>"</code> – The new size for the Light Tracer buffer.
SET_TRACE_SQL_COMMAND	<p>Activates or stops SQL tracing at runtime. The command name with its parameters must be enclosed in quotation marks.</p> <p><i>Example:</i> <code>ADJ1_Send_Admin_Command_Sh illin1027 <process instance> "SET_TRACE_SQL_COMMAND --traceSqlOn \"yes\" --duration \"1\""</code></p>	<code>--traceSqlOn "<yes or no>" --duration "<minutes>"</code> If the IS_DURATION_REQUIRED environment variable is set to 'Y', the --duration parameter is mandatory. If the duration specified in the command exceeds the allowed maximum value, SQL tracing is activated for the maximum allowed period of time only (60 minutes).  Note: The SHUT_DOWN_CHECK_FREQ parameter of the Light Tracer in the ADJ1_CONF_SECTION_PARAM table determines the frequency (defined in the number of messages) at which the SQL Trace Manager checks whether the period specified in the --duration parameter has passed. Therefore, SQL tracing is stopped after the specified duration or after this number of messages has been reached, whichever comes later.

Command	Description	Parameters
SWITCH_LIGHT_TRACER_ON ALL	Activates the Light Tracer without the ability to specify the mode or send parameters.	N/A
SWITCH_LIGHT_TRACER_OFF ALL	Stops the Light Tracer	N/A
CPF_GracefulShutdownCommand	Shuts down the Usage Extract gracefully	N/A

Examples:

- `ADJ1_Send_Admin_Command_Sh hpbl820 US_EX_REG_CUST_PI 245 SET_LIGHT_TRACER_COMMAND – turnTracer “on” –moduleId “ALL” –threadInfo “y”`
- `ADJ1_Send_Admin_Command_Sh hpbl820 US_EX_REG_CUST_PI 245 SWITCH_LIGHT_TRACER_OFF ALL`

Recovery Instructions

When Customer Group Accumulators Extract process starts, it first processes the records that are in the IU (In Use) status from AC1_CONTROL table. The process checks whether the next file appears in the APR1_EXTR_CUSTOMER_RECOVERY control table:

- If it does, the process gets the last processed customer and continues from this point.
- If it does not, the job starts processing from the original file.

After finishing processing IU files, it goes back to its regular flow and processes the Ready ones.

APR1_EXTR_CUSTOMER_RECOVERY Table

The APR1_EXTR_CUST_RECOVERY table is used in Regular Customer and Bill on Demand modes only, if the mode is set in the APR1_CONF_SECTION_PARAM table. It is used to monitor the output files that are created and updated in this mode and enables the process to start its work from the last processed customer.

When an output file is created, it is inserted in the APR1_EXTR_CUST_RECOVERY table according the following parameters:

- Input file ID (the input ID of the file in Audit & Control)
- Output file ID (the ID of the file)
- Offset = 0
- Customer_ID = 0

Every time a new customer is about to be written into the output file, the output manager checks whether the number of customers has reached the threshold (a parameter configured in the APR1_CONF_SECTION_PARAM table). If it has, the process flushes the data into the output file and updates the customer ID, offset, and other parameters in the APR1_EXTR_CUST_RECOVERY table. After processing the file, the output manager deletes the corresponding entry in the table.

If the job was killed and is restarted, it gets the last data that was processed and continues from this point. Therefore, if you wish to start processing a file from the beginning in Regular Customer mode, make sure that the file does not appear in the APR1_EXTR_CUST_RECOVERY table.

34 ADJ1PRCPEXT – Customer Group Accumulators Extract Post Rerate

This chapter describes the Customer Group Accumulators Extract Post Rerate process.

Description

The Customer Group Accumulators Extract Post Rerate extracts rerated accumulator data for specific customers. This process has the same input parameters and executes the same flow as the regular extracts; however, it listens to a different file alias. For more information, see the “ADJ1RCPEXT – Customer Group Accumulators Extract” chapter.

Process Type

Batch job

Run Frequency

By request

Participation in Generic Maps

This section contains information on the generic maps in which the process participates.

Map Name

End-of-Cycle (EOC)

For more information about this map, see *Amdocs Billing Operator Guide*.

Severity Level

Severity Level 2 – The map must hold until the job is fixed.

Activation



Caution: When the home database is up after a failure, the operator must not activate Usage Extracts until the Copy Cycle Usage process, which copies the data from the alternate database to the home database, is complete.

Command Line:	RunJobs ADJ1PRCPEXT BYREQ
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_RegularCustomerPIExtract_Job_Sh
Executable Name:	gcpf1fwcApp

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> US_EX_REG_CUST_PI \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

ADJ1_Send_Admin_Command_Sh hpbl820 US_EX_REG_CUST_PI 245
CPF_GracefulShutdownCommand

Preceding Processes

The following job must run successfully before this process is activated:

- *ADJ1RRPEOC* – Rerate Prepare for End of Cycle

Dependent Processes

The following process can be run only after the successful completion of the Customer Group Accumulators Extract Post Rerate process in the End-of-Cycle map:

- *ADJ1EXTCLEANE* – Extract Clean-up

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*

ADJ1RCPEXT_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1RCPEXT_US_EX_REG_CUST_PI245_20081109_130927_1.log

- *Console log files:*

ADJ1-RegularCustomerPIExtract_Job_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1-RegularCustomerPIExtract_Job_inner_Sh_US_EX_REG_CUST_PI245_20081109_130927.log

- *Operational log files:*

ADJ1PRCPEXT_<Job Record Name>_\${ABP_MARKET}<Date>_<Time>.log

Example:

ADJ1PRCPEXT_BYREQ_M3G_20081109_130927.log

- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter)

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information.
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file, and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).

In addition, the console log contains the number of records written to output files. These records are reported per database partition when the process is finished.



Note: The number of records written to the file matches the number of external records and does not match the number of extracted accumulators.

Extract statistics are printed as follows:

```
EXTR STATISTICS: Total output record created [%d], partition [%s]
```

where:

- *%d* – The number of records written to files
- *%s* – The name of the database partition

- *Operational log files* – The output of operational scripts.
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Input Files

This section describes the input files.

Name

The name of input file is taken from Audit & Control database.

Location and Format

This process receives the file from Audit & Control database.

Contents

This file contains information about the customer/subscriber whose data is to be extracted and in what format this data is to be stored.

Output Files

This section describes the output files.

Name

Output files use the following naming conventions:

- *Regular files*
UsageExtractData<*CustomSuffix*><*DateTime*>_<*PID*>_<*ThreadID*>_<*FileNo*>.dat
- *Error files*
errUsageExtractData<*CustomSuffix*><*DateTime*>_<*PID*>_<*ThreadID*>_<*FileNo*>.dat

Location and Format

- *Location* – \$ABP_APP_ROOT/interfaces/output/us_extract
- *Format* – Semicolon-delimited file. The contents are defined by configuration.

Contents

A regular output file contains event fields specified by configuration, for all extracted records. If an error occurred during processing:

- The regular output file is empty.
- If the MAX_FAILED_CUST parameter is not set (default value is -1), or if the number of erroneous customers is greater than the value of this parameter, the errUsageExtractData file contains the value of -1. Otherwise, the file contains the IDs of all erroneous customers.

35 ADJ1RCAEXT – Customer Group Allowances Extract

This chapter describes Customer Group Allowances Extract process.

Description

The Customer Group Allowances Extract process extracts data of allowances for specific customers.

Process Type

Batch job

Run Frequency

By request

Participation in Generic Maps

This section contains information on the generic maps in which the process participates.

Map Name

- Rerun
- QA

For more information about these maps, see *Amdocs Billing Operator Guide*.

Severity Level

Severity Level 2 – The map must hold until the job is fixed.

Activation



Caution: When the home database is up after a failure, the operator must not activate Usage Extracts until the Copy Cycle Usage process, which copies the data from the alternate database to the home database, is complete.

Command Line:	RunJobs ADJ1RCAEXT BYREQ
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1-RegularCustomerAllowancesExtract_Job_Sh
Executable Name:	gcpf1fwcApp

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> US_EX_REG_CUST_ALW \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

ADJ1_Send_Admin_Command_Sh hpbl820 US_EX_REG_CUST_ALW 250
CPF_GracefulShutdownCommand

Preceding Processes

The following job must run successfully before this process is activated:

- *ADJ1EVGRPLSTN* – Event Group Listener

Dependent Processes

The following process can be run only after the successful completion of the Customer Group Allowances Extract process:

- *ADJ1EXTCLEANE* – Extract Clean-up

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*

ADJ1RCAEXT_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1RCAEXT_US_EX_REG_CUST_ALW250_20081109_130927_1.log

- *Console log files:*

ADJ1-RegularCustomerAllowancesExtract_Job_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1-RegularCustomerAllowancesExtract_Job_inner_Sh_US_EX_REG_CUST_ALW250_20081109_130927.log

- *Operational log files:*

ADJ1RCAEXT_<Job Record Name>_\${ABP_MARKET}<Date>_<Time>.log

Example:

ADJ1RCAEXT_BYREQ_M3G_20081109_130927.log

- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter)

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information.
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file, and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).

In addition, the console log contains the number of records written to output files. These records are reported per database partition when the process is finished.



Note: The number of records written to the file matches the number of external records and does not match the number of extracted allowances.

Extract statistics are printed as follows:

```
EXTR STATISTICS: Total output record created [%d], partition [%s]
```

where:

- *%d* – The number of records written to files
- *%s* – The name of the database partition

- *Operational log files* – The output of operational scripts.
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Input Files

This section describes the input files.

Name

The name of input file is taken from Audit & Control database.

Location and Format

This process receives the file from Audit & Control database.

Contents

The input file contains information about the customer/subscriber whose data is to be extracted and in what format this data is to be stored.

Output Files

This section describes the output files.

Name

Output files use the following naming conventions:

- *Regular files*
UsageExtractData<*CustomSuffix*><*DateTime*>_<*PID*>_<*ThreadID*>_<*FileNo*>.dat
- *Error files*
errUsageExtractData<*CustomSuffix*><*DateTime*>_<*PID*>_<*ThreadID*>_<*FileNo*>.dat

Location and Format

- *Location* – \$ABP_APP_ROOT/interfaces/output/us_extract
- *Format* – Semicolon-delimited file. The contents are defined by configuration.

Contents

A regular output file contains event fields specified by configuration, for all extracted records. If an error occurred during processing:

- The regular output file is empty.
- If the MAX_FAILED_CUST parameter is not set (default value is -1), or if the number of erroneous customers is greater than the value of this parameter, the errUsageExtractData file contains the value of -1. Otherwise, the file contains the IDs of all erroneous customers.

Flow

In the Customer mode, the Customer Group Allowances Extract process (ADJ1RCAEXT) performs the following steps:

1. Reads the next ready Customer Group file from Audit & Control. If several processes are running on different machines, the servers use a shared disk so that the input file is available for all the processes.
2. Opens the output file.
3. Loads the contents of the input file (group ID, customer ID, billing arrangement ID, and subscriber ID, if defined) into a temporary Oracle table. This table is used to retrieve only the requested population from the relevant table through a join.

Before opening the cursor, if the billing arrangement ID is provided in the input file from Amdocs Invoicing, the Extract engine checks whether the billing arrangement is supported according to the format (defined in *f_formatName*). If at least one external record does not support the billing arrangement, the process generates an error, and the flow stops.

4. Retrieves the required information in a loop from partition to partition from the Accumulators table.
5. Opens a cursor using the Extract Engine (an internal component). The cursor joins the temporary table with the Accumulators table.

Information is sorted by customer and subscriber.

For accumulators, information is also sorted by accumulator key so that the global accumulators are selected before the dimensioned accumulators.

- a. For each customer, the process flushes the output into the file and writes the last customer ID that was flushed into a control table. This information may be used in recovery so that the group is extracted starting only from the last processed customer.
- b. For each record:
 - i. If the extract requires the execution of a mapping case, activates the relevant mapping case using the code generated by the Implementation Compiler from the implementation made in the Rating Logic Configurator.
 - ii. Formats the record according to the formatting definitions. Records are formatted regardless of whether the mapping case was activated. If an error occurs during the activation of the mapping case or formatting:
 - ◆ If the MAX_FAILED_CUST parameter is not defined or set to -1, or if the number of erroneous customers written to the error output file is greater than defined in this parameter, stops extracting.
 - ◆ If the value of the MAX_FAILED_CUST parameter is greater than 0, and if the number of erroneous customers has not yet exceeded this value, continues writing the IDs of all erroneous customers into the error output file.

- iii. If the billing arrangement ID is provided in the input file, the Extract Engine checks whether the billing arrangement ID in the record is in the list of the billing arrangements that need to be extracted for the customer .If it is not in the list, the record is dropped.
 - c. Writes the record into the output file. If an error occurred during processing, the regular output file is truncated. The contents of the errUsageExtractsData file depend on the value of the MAX_FAILED_CUST parameter, as described in the “Output Files” section in this chapter.
6. Closes the file at the end of the group and writes it to Audit & Control.
 7. Updates a control table (APR1_EXTR_CUSTOMER_RECOVERY) to indicate that the process has finished successfully.

Environment Variables

The Customer Group Allowances Extract uses the following environment variable that is initialized by the op_adj_env_sh script.

Variable Name	Description	Valid Values/ Default Value
APR_CUST_ALW_XTR_NUM_OF_INSTS	The number of threads that the Extract uses	3

For more information on the script, see the “Amdocs Batch Job Manager Application Script” section in the “Introduction” chapter.

Parameters

The following parameters must be set for the process to run properly. When the job runs as part of a map, the Turbo Charging Cycle Bill Run script populates these parameters automatically. For more information on this script, see the “ADJ1_AP4_CycleBillRun_Sh – Turbo Charging Cycle Bill Run” chapter.

Parameter	Description	Default Value
APPLICATION_ID	The ID of the job that enables it to get its parameters.	US_EX_REG_CUST_ALW250
CCF_FILE_NAME	An XML file that contains the relevant configuration.	APR1_CustAllowanceExtractsReg.ccf
EXTR_CYCLE_CODE	The cycle code for which data is to be extracted.	1
EXTR_CYCLE_INSTANCE	The cycle instance for which data is to be extracted.	1
EXTR_CYCLE_YEAR	The cycle year for which data is to be extracted.	N/A
EXTR_INPUT_MAP_KEY	The input map key. This key is used as a filter. Only the record with this value in appears the DATA_GROUP field of the AC1_CONTROL table.	EXTRACT
EXTR_INSTANCE_ID	The instance ID of the job.	1

Parameter	Description	Default Value
EXTR_INSTANCES_NUM	The number of instances that run in parallel.	1
MAP_MODE	<p>The context (map) in which the process is currently running. Valid values are:</p> <ul style="list-style-type: none"> ■ NONE (if the job is run manually and not as part of a map) ■ EOC ■ RERUN ■ QA ■ UNDO 	N/A
RUN_MODE	<p>The run mode of the map that is currently running. The value of this parameter is provided by Amdocs Invoicing. Valid values are:</p> <ul style="list-style-type: none"> ■ FULL_RUN ■ RUN+AR ■ EXTRACTS ■ FULL_RERUN ■ RERUN+AR ■ FULL_QA ■ FULL_QAM 	N/A

Configuration Parameters

In Turbo Charging, the configuration parameters (properties) of the Customer Group Allowances Extract process are defined in the database.

The following table shows the performance tuning (configuration) parameters of the Customer Group Allowance Extract process. These parameters are defined in the APR1_CONF_SECTION_PARAM table using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*).



Note: Other parameters must not be changed.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APE	config	QUERY_TRACING_DIR	\${ABP_APP_LOG}	The location of trace XML files that show query results.
APE	config	QUERY_TRACING_MODE	FALSE	Indicates whether trace XML files that show query results must be printed.
US_EX_REG_CUST_ALW	AL_FILE_OUTPUT	FILE_ALIAS	RTNONPI	The default value of the Audit & Control file alias for the output file.
US_EX_REG_CUST_ALW	EXTRACT	CUST_RECOVERY_MODE	Y	Indicates whether the Customer Recovery mode is enabled.
US_EX_REG_CUST_ALW	EXTRACT	MAX_CUST_NUM	10	The number of customers that are processed before writing the data into the output file. This parameter is valid if the Customer Recovery mode is enabled. It is recommended that the operator increase the value of this parameter to at least 100 to reduce the number of times that the database must be accessed.
US_EX_REG_CUST_ALW	EXTRACT_ENGINE	RATED_EVENT_FLAG	N	Indicates whether information is extracted for events. If this flag is not set, information is extracted for accumulators.
US_EX_REG_CUST_ALW	FILE_OUTPUT	DATA_GROUP	NA	The default value of the Audit & Control data group for the output file.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
US_EXTRACT	AL_FILE_OUTPUT	ROUTING_CRITERIA	BL1_TO_RPR1	The default value of the Audit & Control routing criteria for the output file.
US_EXTRACT	config	IS_RUN_MODE_JOB	\${ADJ1_IS_JOB} }	Indicates whether the current extract is running as a job: <ul style="list-style-type: none"> ■ Y – Yes ■ N – No
US_EXTRACT	config	UQ_MAX_FETCH_BULK_SIZE	100	The maximum number of records to be fetched in one request. This parameter defines the buffer size. For performance reasons, it is recommended that the operator increase the value of this parameter to at least 1000 to reduce the number of fetches.
US_EXTRACT	EXTRACT	DL_TEMP_TABLE_BUFF_SIZE	1000	The size of the record bulk inserted into a temporary table with customer data.
US_EXTRACT	EXTRACT	GET_NETWORK_FILES_INDICATOR_TIMEOUT	180 sec	The period of time for which the Extract waits for a reply from the Audit & Control Manager about files available in the network.
US_EXTRACT	EXTRACT	GET_NETWORK_FILES_MAX_TIMES	5	The number of times that the Extract tries to retrieve files from the network if the response from the Audit & Control Manager indicates that there are such files available. If no files are received after this number of attempts, the Extract stops looking for files over the network.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
US_EXTRACT	EXTRACT	GETTING_FILES_OVER_NE TWORK_MODE	Y	<p>Determines the behavior of the Extract in a distributed environment if Amdocs Invoicing and the Extract are on different machines:</p> <ul style="list-style-type: none"> ■ <i>N</i> – When there are no more files, the Extract shuts down. ■ <i>Y</i> – Before shutting down, the Extract checks whether relevant files are available in the network. If they do, it waits for a period of time determined by the TIME_TO_WAIT_AFTER_NETWORK_FILES_IND parameter and then tries to get these files before shutting down.
US_EXTRACT	EXTRACT	INITIALIZATION_SLEEP	600	The number of seconds that the process sleeps before starting.
US_EXTRACT	EXTRACT	INSTANCE_ID	<code> \${EXTR_INSTANCE_ID} </code>	The instance ID of the Customer Group Allowances Extract process. This is number of the instance that is running of the total number of instances.
US_EXTRACT	EXTRACT	INSTANCES_NUM	<code> \${EXTR_INSTANCES_NUM} </code>	The number of Customer Group Allowances Extract instances.
US_EXTRACT	EXTRACT	MAP_TYPE	<code> \${MAP_MODE} </code>	The type of the map in which the extract is running.
US_EXTRACT	EXTRACT	MAX_FAILED_CUST	-1	<p>The maximum number of customers for whom the extract can fail before the entire group is marked as failed.</p> <p>The default value of '-1' means that the threshold is disabled.</p>

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
US_EXTRACT	EXTRACT	MAX_FETCHED_RECORD	100	<p>The maximum number of records that the Extract Engine (an internal component) can fetch at once. For performance reasons, it is recommended that the operator increase the value of this parameter to at least 1000 to reduce the number of fetches.</p> <p>This parameter enables flexible configuration of the bulk size. To set a bulk size that is different from the size specified in the UQ_MAX_FETCH_BULK_SIZE parameter, set this parameter to a lower value than the one defined for the UQ_MAX_FETCH_BULK_SIZE parameter.</p>
US_EXTRACT	EXTRACT	MAX_IDLE_TIME	600	The maximum number of second that the process can be idle.
US_EXTRACT	EXTRACT	REQUEST_ID	\${REQUEST_ID} }	The request ID of the current run. In the Cycle mode, the request ID is a concatenation of the cycle code, instance, and year.
US_EXTRACT	EXTRACT	TIME_TO_WAIT_AFTER_NETWORK_FILES_IND	5 sec	The period of time for which the Extract waits between getting a response from Audit & Control Manager that there are files available in the network and trying to retrieve those files. This parameter is used in a distributed environment if the GETTING_FILES_OVER_NETWORK_MODE parameter is ‘Y’.
US_EXTRACT	EXTRACT_ENGINE	MAX_SUBS_NUMBER	0	The maximum number of subscribers that can be extracted per Oracle partition without a full table scan. The value of 0 means unlimited number.
US_EXTRACT	FILE_OUTPUT	FILE_DIRECTORY	\${ABP_AP_ROOT}/interfaces/output/us_extract	The default directory where the output file is created.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
US_EXTRACT	FILE_OUTPUT	FILE_NAME	UsageExtractDat a.dat	The naming convention for the output file.
US_EXTRACT	FILE_OUTPUT	MAX_FILE_SIZE	0	<p>The maximum size of an output file. The value of 0 means unlimited size.</p>  <p><i>Note: The Customer Group Allowances Extract does not take this parameter into consideration even if the parameter exists in the database. If an error message about this parameter appears in the log, it can be ignored.</i></p>
US_EXTRACT	FILE_OUTPUT	MAX_RECORD_NUMBER	100	<p>The maximum number of records that can be written to a single file.</p>  <p><i>Note: The Customer Group Allowances Extract does not take this parameter into consideration even if the parameter exists in the database. If an error message about this parameter appears in the log, it can be ignored.</i></p>

In addition, the process uses Light Tracer configuration parameters, which are defined in the ADJ1_CONF_SECTION_PARAM table. For more information, see **Light Tracer Parameters** in the “**Configuration Parameters**” section in the “**ADJ1EVENTSRV – Event Server**” chapter.

For each process instance, an entry with the name of the process instance must be created in the APR1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
US_EX_REG_CUST_ALW250	US_EX_REG_CUST_ALW

Any parameter to be overridden at the process instance level must be added to the APR1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
US_EX_REG_CUST_ALW250	FILE_OUTPUT	MAX_RECORD_NUMBER	1000

For more information on parameters and the properties mechanism, see the “**Configuration Parameter Mechanism**” appendix.

Configuration Files

The Usage Extract process requires a configuration file (.ccf) that is defined via an internal configuration tool.

The environment variables that appear in the .ccf file and can be customized are defined in the op_adj_env_sh script (for more information, see the “**Environment Variables**” section in this chapter).

Database Connections

The process connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	AC1_CONTROL	Insert
Application	APR_EXTR_MONITOR	Insert, Delete, Update
Application	APR1_CUSTOMER_EX	Insert
Usage	APE1_ACCUMULATORS	Select

Admin Commands

The Customer Group Accumulators Extract supports the following admin commands. For information on how to execute the commands, see the “[Handling Admin Commands](#)” section in the “[High Availability](#)” chapter.

Command	Description	Parameters
SET_LIGHT_TRACER_COMMAND	Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters	<ul style="list-style-type: none"> ▪ Activation parameters: <ul style="list-style-type: none"> • <i>turnTracer “on/off”</i> – Activates or stops the Light Tracer • <i>--turnAssert “on/off”</i> – Activates or stops Assertions • <i>--turnLogOnTrace “on/off”</i> – Activates or stops writing log messages in Light Tracer files • <i>--turnFlowTracer/turnAll “on/off”</i> – Activates or stops tracing a single flow or event that is transferred between machines • <i>--turnAll “on/off”</i> – Activates or stops all of the above. The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter). ▪ <i>--moduleID “ALL”</i> ▪ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • <i>--threadInfo “Y/N”</i> – Specifies whether trace or assertion messages include the thread information. • <i>--timestampInd “Y/N”</i> – Specifies whether trace or assertion messages include the time stamp. • <i>--contextInd “Y/N”</i> – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • <i>--noOfMessages “<number>”</i> – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down. • <i>--period “5/10/20/60/120/480/1440”</i> – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated.

Command	Description	Parameters
		 Note: If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached. <ul style="list-style-type: none"> • <code>--assertAbortInd "Y/N"</code> – Specifies whether the application is to be aborted if the assertion condition fails. ▪ <code>--commandType "changeBufferSize"</code> – Resizes the Light Tracer buffer at run time. • <code>--bufferSize "<number>"</code> – The new size for the Light Tracer buffer.
SET_TRACE_SQL_COMMAND	<p>Activates or stops SQL tracing at runtime. The command name with its parameters must be enclosed in quotation marks.</p> <p><i>Example:</i></p> <pre>ADJ1_Send_Admin_Command_Sh illin1027 <process instance> "SET_TRACE_SQL_COMMAND --traceSqlOn \"yes\" --duration \"1\\""</pre>	<code>--traceSqlOn "<yes or no>" --duration "<minutes>"</code> If the IS_DURATION_REQUIRED environment variable is set to 'Y', the --duration parameter is mandatory. If the duration specified in the command exceeds the allowed maximum value, SQL tracing is activated for the maximum allowed period of time only (60 minutes).  Note: The SHUT_DOWN_CHECK_FREQ parameter of the Light Tracer in the ADJ1_CONF_SECTION_PARAM table determines the frequency (defined in the number of messages) at which the SQL Trace Manager checks whether the period specified in the --duration parameter has passed. Therefore, SQL tracing is stopped after the specified duration or after this number of messages has been reached, whichever comes later.

Command	Description	Parameters
SWITCH_LIGHT_TRACER_ON ALL	Activates the Light Tracer without the ability to specify the mode or send parameters.	N/A
SWITCH_LIGHT_TRACER_OFF ALL	Stops the Light Tracer	N/A
CPF_GracefulShutdownCommand	Shuts down the Usage Extract gracefully	N/A

Examples:

- `ADJ1_Send_Admin_Command_Sh hpbl820 US_EX_REG_CUST_ALW 250 SET_LIGHT_TRACER_COMMAND – turnTracer “on” –moduleId “ALL” –threadInfo “y”`
- `ADJ1_Send_Admin_Command_Sh hpbl820 US_EX_REG_CUST_ALW 250 SWITCH_LIGHT_TRACER_OFF ALL`

Recovery Instructions

When the Customer Group Allowance Extract starts, it first processes the records that are in the IU (In Use) status from AC1_CONTROL table. The process checks whether the next file appears in the APR1_EXTR_CUSTOMER_RECOVERY control table:

- If it does, the process gets the last processed customer and continues from this point.
- If it does not, the job starts processing from the original file.

After finishing processing IU files, it goes back to its regular flow and processes the Ready ones.

APR1_EXTR_CUSTOMER_RECOVERY Table

The APR1_EXTR_CUST_RECOVERY table is used in Regular Customer and Bill on Demand modes only, if the mode is set in the APR1_CONF_SECTION_PARAM table. It is used to monitor the output files that are created and updated in this mode and enables the process to start its work from the last processed customer.

When an output file is created, it is inserted in the APR1_EXTR_CUST_RECOVERY table according the following parameters:

- Input file ID (the input ID of the file in Audit & Control)
- Output file ID (the ID of the file)
- Offset = 0
- Customer_ID = 0

Every time a new customer is about to be written into the output file, the output manager checks whether the number of customers has reached the threshold (a parameter configured in the APR1_CONF_SECTION_PARAM table). If it has, the process flushes the data into the output file and updates the customer ID, offset, and other parameters in the APR1_EXTR_CUST_RECOVERY table. After processing the file, the output manager deletes the corresponding entry in the table.

If the job was killed and is restarted, it gets the last data that was processed and continues from this point. Therefore, if you wish to start processing a file from the beginning in Regular Customer mode, make sure that the file does not appear in the APR1_EXTR_CUST_RECOVERY table.

36 ADJ1PRCAEXT – Customer Group Allowances Extract Post Rerate

This chapter describes Customer Group Allowances Extract Post Rerate process.

Description

The Customer Group Allowances Extract Post Rerate process extracts data of rerated allowances for specific customers. This process has the same input parameters and executes the same flow as the regular extracts; however, it listens to a different file alias. For more information, see the “[ADJ1RCAEXT – Customer Group Allowances Extract](#)” chapter.

Process Type

Batch job

Run Frequency

By request

Participation in Generic Maps

This section contains information on the generic maps in which the process participates.

Map Name

End-of-Cycle (EOC)

For more information about this map, see *Amdocs Billing Operator Guide*.

Severity Level

Severity Level 2 – The map must hold until the job is fixed.

Activation



Caution: When the home database is up after a failure, the operator must not activate Usage Extracts until the Copy Cycle Usage process, which copies the data from the alternate database to the home database, is complete.

Command Line:	RunJobs ADJ1PRCAEXT BYREQ
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1-RegularCustomerAllowancesExtract_Job_Sh
Executable Name:	gcpf1fwcApp

For more information about the activation scripts, see the “[Scripts](#)” section in the “[Introduction](#)” chapter.

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> US_EX_REG_CUST_ALW \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

ADJ1_Send_Admin_Command_Sh hpbl820 US_EX_REG_CUST_ALW 250
CPF_GracefulShutdownCommand

Preceding Processes

The following job must run successfully before this process is activated:

- *ADJ1EVGRPLSTN* – Event Group Listener

Dependent Processes

The following process can be run only after the successful completion of the Customer Group Allowances Extract Post Rerate process:

- *ADJ1EXTCLEANE* – Extract Clean-up

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*

ADJ1RCAEXT_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1RCAEXT_US_EX_REG_CUST_ALW250_20081109_130927_1.log

- *Console log files:*

ADJ1-RegularCustomerAllowancesExtract_Job_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1-RegularCustomerAllowancesExtract_Job_inner_Sh_US_EX_REG_CUST_ALW250_20081109_130927.log

- *Operational log files:*

ADJ1PRCAEXT_<Job Record Name>_\${ABP_MARKET}_<Date>_<Time>.log

Example:

ADJ1PRCAEXT_BYREQ_M3G_20081109_130927.log

- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter)

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information.
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file, and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).

In addition, the console log contains the number of records written to output files. These records are reported per database partition when the process is finished.



Note: The number of records written to the file matches the number of external records and does not match the number of extracted allowances.

Extract statistics are printed as follows:

```
EXTR STATISTICS: Total output record created [%d], partition [%s]
```

where:

- *%d* – The number of records written to files
- *%s* – The name of the database partition

- *Operational log files* – The output of operational scripts.
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Input Files

This section describes the input files.

Name

The name of input file is taken from Audit & Control database.

Location and Format

This process receives the file from Audit & Control database.

Contents

The input file contains information about the customer/subscriber whose data is to be extracted and in what format this data is to be stored.

Output Files

This section describes the output files.

Name

Output files use the following naming conventions:

- *Regular files*
UsageExtractData<*CustomSuffix*><*DateTime*>_<*PID*>_<*ThreadID*>_<*FileNo*>.dat
- *Error files*
errUsageExtractData<*CustomSuffix*><*DateTime*>_<*PID*>_<*ThreadID*>_<*FileNo*>.dat

Location and Format

- *Location* – \$ABP_APP_ROOT/interfaces/output/us_extract
- *Format* – Semicolon-delimited file. The contents are defined by configuration.

Contents

A regular output file contains event fields specified by configuration, for all extracted records. If an error occurred during processing:

- The regular output file is empty.
- If the MAX_FAILED_CUST parameter is not set (default value is -1) or if the number of erroneous customers is greater than the value of this parameter, the errUsageExtractData file contains the value of -1. Otherwise, the file contains the IDs of all erroneous customers.

37 ADJ1BODEXT – Bill on Demand Extract

This chapter describes Bill on Demand Extract process.

Description

The Bill on Demand Extract daemon extracts the data of rated events and accumulators from the Usage database according to the cycle code, cycle instance, and cycle year. It formats the data according to the format name and writes it to output files. These files are registered in Audit & Control.

This special instance of the Usage Extract process is used to provide the data required by Amdocs Invoicing in the Bill on Demand mode. It combines regular Event, Accumulator, and Allowance extracts.

Process Type

Daemon

Run Frequency

Runs continuously

Activation



Caution: When the home database is up after a failure, the operator must not activate Usage Extracts until the Copy Cycle Usage process, which copies the data from the alternate database to the home database, is complete.

Command line:	ADJ1_CycleBODEExtract_Daemon_Shell_Sh -n <APPLICATION_ID> -c “-f <CCF_FILE_NAME>” -e <EWD_EXE>  Note: The -e parameter is optional. It is used in environments with Availability Manager.
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_CycleBODEExtract_Daemon_Shell_Sh
Executable Name:	gcpf1fwcApp

Example:

```
ADJ1_CycleBODEExtract_Daemon_Shell_Sh -n US_EX_BOD350 -c “-f  
APR1_CustEventExtractsBOD.ccf”
```

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> US_EX_BOD \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes

Example:

ADJ1_Send_Admin_Command_Sh hpbl820 US_EX_BOD 350
CPF_GracefulShutdownCommand

Dependent Processes

The following process can work on the output of the Bill on Demand Extract if the bill formatting application requires a different sorting of the data than the one that the Bill on Demand Extract provides by default:

- *ADJ1MASD* – Merge and Sort daemon

For more information about the Merge and Sort daemon, see *Amdocs Billing Utilities Run Book*.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*

ADJ1BODEXT_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1BODEXT_US_EX_BOD350_20090901_110707_0.log

- *Console log files:*

ADJ1_CycleBODEExtract_Daemon_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1_CycleBODEExtract_Daemon_inner_Sh_US_EXTRACT350_20081109_130927.log

- *Operational log files:*

ADJ1BODEXT_<APPLICATION_ID>_\${ABP_MARKET}<Date>_<Time>.log

Example:

ADJ1BODEXT_US_EX_BOD350_M3G_20090901_110704.log

- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter)

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information.
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file, and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).

In addition, the console log contains the number of records written to output files. These records are reported per database partition when the process is finished.



Note: The number of records written to the file matches the number of external records and does not match the number of extracted events, accumulators, or allowances.

Extract statistics are printed as follows:

```
EXTR STATISTICS: Total output record created [%d], partition [%s]
```

where:

- *%d* – The number of records written to files
- *%s* – The name of the database partition

- *Operational log files* – The output of operational scripts.
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Input Files

This section describes the input files.

Name

The name of input file is taken from Audit & Control database.

Location and Format

This process receives the file from Audit & Control database.

Contents

This file contains information about the customer or subscriber whose data is to be extracted and indicates in what format this data is to be stored.

Output Files

This section describes the output files.

Name

Output files use the following naming conventions:

- *Regular files*
UsageExtractData<*CustomSuffix*><*DateTime*>_<*PID*>_<*ThreadID*>_<*FileNo*>.dat
- *Error files*
errUsageExtractData<*CustomSuffix*><*DateTime*>_<*PID*>_<*ThreadID*>_<*FileNo*>.dat

Location and Format

- *Location* – \$ABP_APP_ROOT/interfaces/output/us_extract
- *Format* – Semicolon-delimited file. The contents are defined by configuration.

Contents

A regular output file contains event fields specified by configuration, for all extracted records. If an error occurred during processing:

- The regular output file is empty.
- If the MAX_FAILED_CUST parameter is not set (default value is -1), or if the number of erroneous customers is greater than the value of this parameter, the errUsageExtractData file contains the value of -1. Otherwise, the file contains the IDs of all erroneous customers.

Flow

The Bill on Demand Extract process (ADJ1BODEXT) performs the following steps:

1. Opens three types of acceptors for the following:

- Event files
- Accumulator files
- Non-accumulator files

Each acceptor reads the next ready Customer Group file from Audit & Control.

2. Opens the output file.
3. Loads the contents of the input file (group ID, customer ID, billing arrangement ID, and subscriber ID, if defined) into a temporary Oracle table. This table is used to retrieve only the requested population from the relevant table through a join.

Before opening the cursor, if the billing arrangement ID is provided in the input file from Amdocs Invoicing, the Extract engine checks whether the billing arrangement is supported according to the format (defined in *f_formatName*). If at least one external record does not support the billing arrangement, the process generates an error, and the flow stops.

4. Retrieves the required information in a loop from partition to partition from the Accumulators table.
5. Opens a cursor using the Extract engine (an internal component). The cursor joins the temporary table with the relevant table.

Information is sorted by customer and subscriber.

For accumulators, information is also sorted by accumulator key so that the global accumulators are selected before the dimensioned accumulators.

- a. For each customer, the process flushes the output into the file and writes the last customer ID that was flushed into a control table. This information may be used in recovery so that the group is extracted starting only from the last processed customer.
- b. For each record:
 - If the extract requires the execution of a mapping case, activates the relevant mapping case using the code generated by the Implementation Compiler from the implementation made in the Rating Logic Configurator.
 - Formats the record according to the formatting definitions. Records are formatted regardless of whether the mapping case was activated.
 - ◆ If the MAX_FAILED_CUST parameter is not defined or set to -1, or if the number of erroneous customers written to the error output file is greater than defined in this parameter, stops extracting.
 - ◆ If the value of the MAX_FAILED_CUST parameter is greater than 0, and if the number of erroneous customers has not yet exceeded this value, continues writing the IDs of all erroneous customers into the error output file.

- If the billing arrangement ID is provided in the input file, the Extract engine checks whether the billing arrangement ID in the record is in the list of the billing arrangements that need to be extracted for the customer .If it is not in the list, the record is dropped.
 - c. Writes the record into the output file. If an error occurred during processing, the regular output file is truncated. The contents of the errUsageExtractsData files depend on the value of the MAX_FAILED_CUST parameter, as described in the “Output Files” section in this chapter.
6. Closes the file at the end of the group and writes it to Audit & Control.
 7. Updates a control table (APR1_EXTR_CUSTOMER_RECOVERY) to indicate that the process has finished successfully.

Parameters

The following parameters must be set for the process to run properly.

Parameter	Description	Default Value
APPLICATION_ID	The ID of the job that enables it to get its parameters	US_EX_BOD152
CCF_FILE_NAME	An XML file that contains the relevant configuration	APR1_CustEventExtractsBOD.ccf

Configuration Parameters

In Turbo Charging, the configuration parameters (properties) of the Bill on Demand Extract processes are defined in the database.

The following table shows the performance tuning (configuration) parameters of the Bill on Demand Extract. These parameters are defined in the APR1_CONF_SECTION_PARAM table using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*).



Note: Other parameters must not be changed.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
APE	config	QUERY_TRACING_DIR	\${ABP_APP_LOG}	The location of trace XML files that show query results.
APE	config	QUERY_TRACING_MODE	FALSE	Indicates whether trace XML files that show query results must be printed.
US_EX_BOD	AL_FILE_OUTPUT	FILE_ALIAS	RTNONPI	The default value of the Audit & Control file alias for the Non-Accumulator output files.
US_EX_BOD	AL_FILE_OUTPUT	ROUTING_CRITERIA	RPR1_TO_BL1	The default value of the Audit & Control routing criteria for the Non-Accumulator output files.
US_EX_BOD	EV_FILE_OUTPUT	FILE_ALIAS	BRTEVT	The default value of the Audit & Control file alias for the Event output files.
US_EX_BOD	EV_FILE_OUTPUT	ROUTING_CRITERIA	RPR1_TO_BL1	The default value of the Audit & Control routing criteria for the Event output files.
US_EX_BOD	EXTRACT	CUST_RECOVERY_MODE	Y	Indicates whether the Customer Recovery mode is enabled.
US_EX_BOD	EXTRACT	MAX_CUST_NUM	10	The number of customers that are processed before writing the data into the output file. This parameter is valid if the Customer Recovery mode is enabled.
US_EX_BOD	PI_FILE_OUTPUT	FILE_ALIAS	RTPI	The default value of the Audit & Control file alias for the Accumulator output files.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
US_EX_BOD	PI_FILE_OUTPUT	ROUTING_CRITERIA	RPR1_TO_BL1	The default value of the Audit & Control routing criteria for the Accumulator output files.
US_EXTRACT	config	IS_RUN_MODE_JOB	\${ADJ1_IS_JOB}	Indicates whether the extract is running as a job.
US_EXTRACT	config	UQ_MAX_FETCH_BULK_SIZE	100	The maximum number of records to be fetched in one request. This parameter defines the buffer size.
US_EXTRACT	EXTRACT	DL_TEMP_TABLE_BUFF_SIZE	1000	The size of the record bulk inserted into a temporary table with customer data.
US_EXTRACT	EXTRACT	INITIALIZATION_SLEEP	600	The number of seconds for which the process sleeps before initialization.
US_EXTRACT	EXTRACT	INSTANCE_ID	\${EXTR_INSTANCES_ID}	The instance ID of the Bill on Demand Extract processes. This is number of the instance that is running of the total number of instances.
US_EXTRACT	EXTRACT	INSTANCES_NUM	\${EXTR_INSTANCES_NUM}	The number of the Bill on Demand Extract instances.
US_EXTRACT	EXTRACT	MAX_FAILED_CUST	-1	<p>The maximum number of customers for whom the extract can fail before the entire group is marked as failed.</p> <p>The default value of '-1' means that the threshold is disabled.</p>
US_EXTRACT	EXTRACT	MAX_FETCHED_RECORD	100	<p>The maximum number of records that the Extract engine (an internal component) can fetch at once. This parameter enables flexible configuration of the bulk size. To set a bulk size that is different from the size specified in the UQ_MAX_FETCH_BULK_SIZE parameter, set this parameter to a lower value than the one defined for the UQ_MAX_FETCH_BULK_SIZE parameter.</p>

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
US_EXTRACT	EXTRACT	MAX_IDLE_TIME	600	The maximum number of seconds that the process can be idle.
US_EXTRACT	EXTRACT_ENGINE	MAX_SUBS_NUMBER	0	The maximum number of subscribers that can be extracted per Oracle partition without a full table scan. The value of 0 means unlimited number.
US_EXTRACT	FILE_OUTPUT	FILE_DIRECTORY	<code> \${ABP_APP_R_OOT}/interfaces/output/us_extract</code>	The default directory where the output file is created.
US_EXTRACT	FILE_OUTPUT	FILE_NAME	UsageExtractData.dat	The naming convention for the output file.
US_EXTRACT	FILE_OUTPUT	MAX_FILE_SIZE	0	The maximum size of an output file. The value of 0 means unlimited size.  Note: <i>The Bill on Demand Extract does not take this parameter into consideration even if the parameter exists in the database. If an error message about this parameter appears in the log, it can be ignored.</i>
US_EXTRACT	FILE_OUTPUT	MAX_RECORD_NUMBER	10000	The maximum number of records that can be written to a single file.  Note: <i>The Bill on Demand Extract does not take this parameter into consideration even if the parameter exists in the database. If an error message about this parameter appears in the log, it can be ignored.</i>
US_EXTRACT	FILE_OUTPUT	ROUTING_CRITERIA	BL1_TO_RPR1	The default value of the Audit & Control routing criteria for the output file, used for initialization.

In addition, the process uses Light Tracer configuration parameters, which are defined in the ADJ1_CONF_SECTION_PARAM table. For more information, see [Light Tracer Parameters](#) in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

For each process instance, an entry with the name of the process instance must be created in the APR1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
US_EX_BOD350	US_EX_BOD

Any parameter to be overridden at the process instance level must be added to the APR1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
US_EX_BOD350	EXTRACT	CUST_RECOVERY_MODE	N

For more information on parameters and the properties mechanism, see the “Configuration Parameter Mechanism” appendix.

Configuration Files

The Usage Extract process requires a configuration file (.ccf) that is defined via an internal configuration tool.

This file defines the number of threads for extracting the data:

```
<property name="Task Name" value="CustBODExtractManager" />
...
<property name="Number of instances" value="1" /> is the number of threads that are activated
...
```

Database Connection

The process connects to the following tables during processing:

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	<ul style="list-style-type: none"> ■ APR1_CUSTOMER_EX ■ AC1_CONTROL 	Insert Insert, Update
Customer Usage	<ul style="list-style-type: none"> ■ APE1_SUBSCR_DATA ■ APE1_SUBSCR_OFFERS ■ APE1_SUBSCR_PARAMS 	Select
Usage	<ul style="list-style-type: none"> ■ APE1_ACCUMULATORS ■ APE1_RATED_EVENT 	Select Select

Admin Commands

The Bill on Demand Extract supports the following admin commands. For information on how to execute the commands, see the “Handling Admin Commands” section in the “High Availability” chapter.

Command	Description	Parameters
SET_LIGHT_TRACER_COMMAND	Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters	<ul style="list-style-type: none"> ▪ Activation parameters: <ul style="list-style-type: none"> • <i>turnTracer “on/off”</i> – Activates or stops the Light Tracer • <i>--turnAssert “on/off”</i> – Activates or stops Assertions • <i>--turnLogOnTrace “on/off”</i> – Activates or stops writing log messages in Light Tracer files • <i>--turnFlowTracer/turnAll “on/off”</i> – Activates or stops tracing a single flow or event that is transferred between machines • <i>--turnAll “on/off”</i> – Activates or stops all of the above. The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter). ▪ <i>--moduleID “ALL”</i> ▪ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • <i>--threadInfo “Y/N”</i> – Specifies whether trace or assertion messages include the thread information. • <i>--timestampInd “Y/N”</i> – Specifies whether trace or assertion messages include the time stamp. • <i>--contextInd “Y/N”</i> – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • <i>--noOfMessages “<number>”</i> – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down. • <i>--period “5/10/20/60/120/480/1440”</i> – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated.

Command	Description	Parameters
		 Note: If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached. <ul style="list-style-type: none"> • --assertAbortInd "Y/N" – Specifies whether the application is to be aborted if the assertion condition fails. ▪ --commandType "changeBufferSize" – Resizes the Light Tracer buffer at run time. • --bufferSize "<number>" – The new size for the Light Tracer buffer.
SET_TRACE_SQL_COMMAND	<p>Activates or stops SQL tracing at runtime. The command name with its parameters must be enclosed in quotation marks.</p> <p><i>Example:</i></p> <pre>ADJ1_Send_Admin_Command_Sh illin1027 <process instance> "SET_TRACE_SQL_COMMAND --traceSqlOn \"yes\" --duration \"1\""</pre>	<p>--traceSqlOn "<yes or no>" --duration "<minutes>"</p> <p>If the IS_DURATION_REQUIRED environment variable is set to 'Y', the --duration parameter is mandatory. If the duration specified in the command exceeds the allowed maximum value, SQL tracing is activated for the maximum allowed period of time only (60 minutes).</p>  Note: The SHUT_DOWN_CHECK_FREQ parameter of the Light Tracer in the ADJ1_CONF_SECTION_PARAM table determines the frequency (defined in the number of messages) at which the SQL Trace Manager checks whether the period specified in the --duration parameter has passed. Therefore, SQL tracing is stopped after the specified duration or after this number of messages has been reached, whichever comes later.

Command	Description	Parameters
SWITCH_LIGHT_TRACER_ON ALL	Activates the Light Tracer without the ability to specify the mode or send parameters.	N/A
SWITCH_LIGHT_TRACER_OFF ALL	Stops the Light Tracer	N/A
CPF_GracefulShutdownCommand	Shuts down the Usage Extract gracefully	N/A

Examples:

- `ADJ1_Send_Admin_Command_Sh hpbl820 US_EX_BOD 350 SET_LIGHT_TRACER_COMMAND – turnTracer “on” – moduleID “ALL” –threadInfo “y”`
- `ADJ1_Send_Admin_Command_Sh hpbl820 US_EX_BOD 350 SWITCH_LIGHT_TRACER_OFF ALL`

Recovery Instructions

When the Bill on Demand Extract starts, it first processes the records that are in the IU (In Use) status from AC1_CONTROL table. The process checks whether the next file appears in the APR1_EXTR_CUSTOMER_RECOVERY control table:

- If it does, the process gets the last processed customer and continues from this point.
- If it does not, the daemon starts processing from the original file.

After finishing processing IU files, it goes back to its regular flow and processes the Ready ones.

APR1_EXTR_CUSTOMER_RECOVERY Table

The APR1_EXTR_CUST_RECOVERY table is used in Bill on Demand mode if the mode is set in the APR1_CONF_SECTION_PARAM table. It is used to monitor the output files that are created and updated in this mode and enables the process to start its work from the last processed customer.

When an output file is created, it is inserted in the APR1_EXTR_CUST_RECOVERY table according the following parameters:

- Input file ID (the input ID of the file in Audit & Control)
- Output file ID (the ID of the file)
- Offset = 0
- Customer_ID = 0

Every time a new customer is about to be written into the output file, the output manager checks whether the number of customers has reached the threshold (a parameter configured in the APR1_CONF_SECTION_PARAM table). If it has, the process flushes the data into the output file and updates the customer ID, offset, and other parameters in the APR1_EXTR_CUST_RECOVERY table. After processing the file, the output manager deletes the corresponding entry in the table.

If the job was killed and is restarted, it gets the last data that was processed and continues from this point. Therefore, if you wish to start processing a file from the beginning in Bill on Demand mode, make sure that the file does not appear in the APR1_EXTR_CUST_RECOVERY table.

38 ADJ1EXTCLEANE – Extract Clean-up

This chapter describes Extract Clean-up process.

Description

The Extract Clean-up process cleans the APR1_EXTR_MONITOR and APR1_OP_MAP_EVENTS tables according to the Request ID.

Process Type

Batch job

Run Frequency

By request

Participation in Generic Maps

This section contains information on the generic maps in which the process participates.

Map Name

- End-of-Cycle (EOC)
- Undo
- Rerun
- QA

For more information about these maps, see *Amdocs Billing Operator Guide*.

Severity Level

Severity Level 2 – The map must hold until the job is fixed.

Activation

Command line:	RunJobs ADJ1EXTCLEANE BYREQ
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Usg_Extr_Cleanup_Sh
Executable Name:	N/A

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Preceding Processes

The following jobs must run successfully before this process is activated:

- In the End-of-Cycle map:
 - *ADJ1PRCAEXT* – Customer Group Allowances Extract Post Rerate
 - *ADJ1PRCPEXT* – Customer Group Accumulators Extract Post Rerate
 - *ADJ1RCPEXT* – Customer Group Accumulators Extract

- *ADJ1DISPEOC* – Dispatcher for End of Cycle
- *ADJ1POCEEXT* – Optimized Customer Group Event Extract Post Rerate
- *ADJ1EVGRPLSTN* – Event Group Listener
- In the Undo map:
 - *ADJ1DISPEOC* – Dispatcher for End of Cycle
 - *ADJ1RRPOPREP* – Rerate Population Report
 - *ADJ1RRPEOC* – Rerate Prepare for End of Cycle
- In the Rerun map:
 - *ADJ1PEFNSHNTF* – Accumulators Extract Finish Notification
 - *ADJ1RCPEXT* – Customer Group Accumulators Extract
 - *ADJ1OCEEXT* – Optimized Customer Group Event Extract
- In the QA map:
 - *ADJ1RCAEXT* – Customer Group Allowances Extract
 - *ADJ1OCEEXT* – Optimized Customer Group Event Extract

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

- *Operational log files:*
ADJ1EXTCLEANE_<Job Record Name>_\${ABP_MARKET}_<Date>_<Time>.log

Example:

ADJ1EXTCLEANE_BYREQ_M3G_20081109_130927.log

Location

Log files are located in the following directory:

- *Operational log files* – \$ABP_LOG

Contents

The log files contain the following:

- *Operational log files* – The output of operational scripts

Flow

The Extract Clean-up job deletes entries from the APR1_EXTR_MONITOR and APR1_OP_MAP_EVENTS tables according to the request ID.

Parameters

The following parameters must be set for the process to run properly. When the job runs as part of a map, the Turbo Charging Cycle Bill Run script populates these parameters automatically. For more information on this script, see the “ADJ1_AP_R_CycleBillRun_Sh – Turbo Charging Cycle Bill Run” chapter.

Parameter	Description	Default Value
CYCLE_CODE	The cycle code of the current map run.	N/A
CYCLE_INSTANCE	The cycle instance of the current map run.	N/A
CYCLE_YEAR	The cycle year of the current map run.	N/A
MAP_MODE	The context (map) in which the process is currently running. Valid values are: <ul style="list-style-type: none">▪ NONE (if the job is run manually and not as part of a map)▪ EOC▪ RERUN▪ QA▪ UNDO	N/A
RUN_REQUEST_ID	The request ID of the current map run.	N/A

Database Connections

The process connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	<ul style="list-style-type: none">▪ APR1_EXTR_MONITOR▪ APR1_OP_MAP_EVENTS	Delete

Recovery Instructions

Rerun the job.

39 ADJ1OUTCOL – Outcollect Prepare

This chapter describes the Outcollect Prepare job.

Description

When subscribers visit other countries or networks, they become guests on a different mobile network. Calls served by host systems are first processed by the host system's data center, and then delivered to the subscriber's home system for final billing.

When the subscriber's home system receives event data records from host systems, they are known as *roaming incollects*, as the subscriber has been roaming in a host system. The host system that served the calls views the same calls as *roaming outcollects*.

The Outcollect Prepare job extracts data of rated outcollect events from the Usage database according to special outcollect cycle codes. It formats the data according to the format name and writes it to output files. These files are registered in Audit & Control and can be read by a clearinghouse interface product, such as Amdocs Acquisition & Formatting, Amdocs Roam Clearing Manager, or an external system.

Process Type

Batch job

Run Frequency

- By request
- As part of a map that is run by a scheduler (for example, on a daily basis)

Activation

Command Line:	RunJobs ADJ1OUTCOL BYREQ
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Outcol_Job_Sh
Executable Name:	gcpf1fwcApp

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP> OUTCOL \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand*
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_Send_Admin_Command_Sh

Example:

ADJ1_Send_Admin_Command_Sh hpbl820 OUTCOL 148
CPF_GracefulShutdownCommand

Dependent Processes

The following process can be run only after the successful completion of the Outcollect Prepare job:

- *ADJ1MAS* – Merge and Sort

For more information about this process, see *Amdocs Billing Utilities Run Book*.

Affected Applications

The output files of the Outcollect Prepare process are in the CSV (ASCII delimited) format. Any Amdocs or external clearinghouse interface product that is to accept the files extracted by the Outcollect Prepare process and pack them in the TAP format must be able to read this output format, which is specified in the OFM1_Formatting.xml file. For more information, see *Turbo Charging Detailed Interfaces* and the “External Outputs Formatting Implementation” part of *Rating Logic Configurator Implementation Best Practices*.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*

ADJ1OUTCOL_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1OUTCOL_OUTCOL148_20081109_130927_1.log

- *Console log files:*

ADJ1_Outcol_Job_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1_Outcol_Job_inner_Sh_OUTCOL148_20081109_130927.log

- *Operational log files:*

ADJ1OUTCOL_<Job Record Name>_\${ABP_MARKET}_<Date>_<Time>.log

Example:

ADJ1OUTCOL_BYREQ_M3G_20081109_130927.log

- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter)

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information.
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).
- *Operational log files* – The output of operational scripts.
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Output Files

This section describes the output files.

Name

<Custom Name>_<Process ID>_<Group ID>_<Date Time>_<PID>_<Thread ID>_<Partition Name>_<Process File Number><Group File Sequence>. <Custom Extension>

Location and Format

- *Root Location* – \$ABP_APP_ROOT/interfaces/output/outcollect.
- *Format* – Semicolon-delimited file. The contents are defined by configuration.

Contents

The output file contains event fields specified by configuration, for all extracted records.

Flow

The Outcollect Prepare process (ADJ1OUTCOL) performs the following steps:

1. The Process Manager reads the input parameters and divides the work among the working threads.
2. Each working thread performs the following:
 - a. Accepts the cycle code, instance, and Oracle partition to work on as input.
 - b. Opens the output file.
 - c. Opens a cursor on the Rated Events table using the Extract Engine (an internal component). The cursor extracts the events that belong to the effective providers in the partition:
 - For rated events, performs a full partition scan or index access according to the number of providers to extract in the partition.
 - If an additional WHERE clause was specified, adds it to the cursor statement.
 - d. For each record found in the partition:
 - Formats the record according to the formatting definitions.
 - Writes the record into the output file.
 - e. Each configurable number of records (RECORDS_NUM_BEFORE_UPDATE), the working thread sends the keys of the APE1_RATED_EVENT table rows to the Committer for an update.

When the Committer gets an update message, it updates the input rows in the APE1_RATED_EVENT table with the ‘P’ status (without committing).

- f. Each configurable number of records (RECORDS_NUM_BEFORE_COMMIT), the working thread sends all closed files and the keys of the APE1_RATED_EVENT table rows to the Committer for an update and commit.

When the Committer gets a commit message, it performs the following:

- Updates the input rows in the APE1_RATED_EVENT table with the ‘P’ status.
- Registers the input files in Audit & Control.
- Commits the connection.

Environment Variables

The Outcollect Prepare process uses the following environment variable that is initialized by the op_adj_env_sh script.

Name	Description	Default Value
APR_OUTCOLLECT_EXTRACTOR_THREADS	The number of Extractor threads. Because the number of Committer threads equals the number of Extractor threads, this variable controls the number of Committer threads as well.	1

For more information on the script, see the “Amdocs Batch Job Manager Application Script” section in the “Introduction” chapter.

Parameters

The following parameters must be set for the process to run properly.

Parameter	Description	Default Value
APPLICATION_ID	The ID of the job that enables it to get its parameters	OUTCOL148
CCF_FILE_NAME	An XML file that contains the relevant configuration	APR1_Outcollect.ccf
PROCESS_NUM_OUT_OF_TOTAL	The number of the process instance of the total number of instances	1
TOTAL_NUM_OF_PROCESSES	The total number of instances that must run	1

Configuration Parameters

In Turbo Charging, the configuration parameters (properties) of the Outcollect Prepare process are defined in the database.

The following table shows the performance tuning (configuration) parameters of the Outcollect Prepare process. These parameters are defined in the APR1_CONF_SECTION_PARAM table using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*).



Note: Other parameters must not be changed.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
OUTCOL	config	UQ_MAX_FETCH_BULK_SIZE	100	The maximum number of records to be fetched in one request. This parameter defines the buffer size.
OUTCOL	EXTRACT	ADDITIONAL_WHERE_CLAUSE	 <i>Note: This parameter is optional and appears if there is an additional WHERE clause.</i>	An additional WHERE clause that can be added to the SELECT statement during customization. The clause must follow proper SQL syntax.
OUTCOL	EXTRACT	FORMAT_NAME	Outcollect Event Format	The name of the format that is used by the Extract Engine to format extracted data.
OUTCOL	EXTRACT	FULL_TABLE_SCAN_THRESHOLD	100	The maximum number of providers that can be extracted from a partition using an index access. If the number of providers is greater, a full table scan is used.
OUTCOL	EXTRACT	HINT_CLAUSE	<code>(/*+ full (e) */)</code>  <i>Note: This parameter is optional.</i>	The SQL hint used by the Extract Engine (an internal component) if the number of providers in the partition is greater than the FULL_TABLE_SCAN_THRESHOLD parameter.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
OUTCOL	EXTRACT	MAX_FETCHED_RECORD	10000	<p>The maximum number of records that the Extract Engine (an internal component) can fetch at once.</p> <p>This parameter enables flexible configuration of the bulk size. To set a bulk size that is different from the size specified in the UQ_MAX_FETCH_BULK_SIZE parameter, set this parameter to a lower value than the one defined for the UQ_MAX_FETCH_BULK_SIZE parameter.</p>
OUTCOL	FILE_OUTPUT	FILE_NAME	OUTCOLLECT.dat	The naming convention for the output file.
OUTCOL	FILE_OUTPUT	ROOT_PATH	\${ABP_APP_ROOT}	The default root directory where the output and temporary files are written.
OUTCOL	FILE_OUTPUT	ROUTING_CRITERIA	RPR1_TO_BL1	The routing criteria to be used by Audit & Control for the output files.
OUTCOL	FILE_OUTPUT	SOURCE_FILES_PATH	/interfaces/source	The relative source directory where the output files that have not been registered in Audit & Control are written.
OUTCOL	FILE_OUTPUT	TARGET_FILES_PATH	/interfaces/output/outcollect	<p>The relative path to the output files after they have been registered in Audit & Control. The full path to the output files after registration is ROOT_PATH/TARGET_FILES_PATH/AC_RECOMMENDED_PATH.</p>

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
OUTCOL	SERVER_PARAMETERS	DATALAYERS_BUFFER_SIZE	1000	The size of the data layer buffer.
OUTCOL	SERVER_PARAMETERS	EXTENSION_PERIOD	0	The number of days that the providers remain effective after the expiration date.
OUTCOL	SERVER_PARAMETERS	INSTANCE_ID	<code> \${PROCESS_NUM_OUT_OF_TOTAL }</code>	The instance ID of the Outcollect process. This is the number of the instance that is running of the total number of instances.
OUTCOL	SERVER_PARAMETERS	INSTANCES_NUM	<code> \${TOTAL_NUM_OF_PROCESSES }</code>	The number of instances of the Outcollect Prepare job.
OUTCOL	SERVER_PARAMETERS	RECORDS_NUM_BEFORE_COMMIT	2000	The number of records to extract before updating the records in the APE1_RATED_EVENT table, registering the files in Audit & Control, and committing.
OUTCOL	SERVER_PARAMETERS	RECORDS_NUM_BEFORE_UPDATE	500	The number of records to extract before updating the APE1_RATED_EVENT table with the 'P' status.
OUTCOL148	SERVER_PARAMETERS	CYCLE_LIST	99	The list of cycle codes that the process must handle. Each instance of the Outcollect Prepare job can work on one Usage database connection only. This means that the input CYCLE_LIST parameter must contain cycles that reside in the same Usage database; otherwise, the process will fail.

In addition, the process uses Light Tracer configuration parameters, which are defined in the ADJ1_CONF_SECTION_PARAM table. For more information, see [Light Tracer Parameters](#) in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

The Outcollect Prepare Job reads additional configuration parameters related to providers from the ADJ1_OUTCOL_PROVIDER table. These parameters are:

- *PROVIDER_ID* – The identifier of the external provider
- *CUSTOMER_ID* – The identifier of the customer, which is the internal identifier for the provider
- *CYCLE_CODE* – The cycle code to which the provider belongs
- *GROUP_ID* – The identifier of the group to which the provider belongs
- *MIN_TIME_TO_SEND* – The minimum amount of time that must pass before handling the provider again
- *MAX_RECS_IN_FILE* – The maximum number of records in a file for the provider
- *SEND_EMPTY_NOTIF* – Indicates whether to send an empty file if the provider does not have records for extraction
- *EXPIRATION_DATE* – The expiration date of the provider
- *EFFECTIVE_DATE* – The effective date of the provider
- *PROVIDER_DESC* – The description of the provider
- *RESOURCE_TYPE* – The outcollect resource type, which is used to retrieve the resource information from the Resources table

For each process instance, an entry with the name of the process instance must be created in the APR1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
OUTCOL148	OUTCOL

Any parameter to be overridden at the process instance level must be added to the APR1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
OUTCOL148	SERVER_PARAMETERS	EXTENSION_PERIOD	1

For more information on parameters and the properties mechanism, see the “Configuration Parameter Mechanism” appendix.

Configuration Files

The Outcollect Prepare process requires a configuration file (.ccf) that is defined via an internal configuration tool.

This file defines the following parameters:

- The number of Extractor threads:

```
<property name="Task Name" value="ExtractorBS" /> \
...
<threads_strategy>
...
<property name="Threads" value="${APR_OUTCOLLECT_EXTRACTORBS_NUM_OF_THREADS}" /> is the number of
threads that are activated
...
```

- The number of Committer tasks, which must be equal to the number of Extractor threads:

```
<property name="Task Name" value="CommitterBS" />
...
<property name="Number of instances" value="${APR_OUTCOLLECT_EXTRACTORBS_NUM_OF_THREADS}" />
```

- The file alias:

```
<FileAC>
...
<property name="file alias" value="OUTCOL" />
```

- The file union:

```
<FileAC>
...
<property name="file union" value="USAGE" />
```

The environment variables that appear in the .ccf file and can be customized are defined in the op_adj_env_sh script (for more information, see the “Environment Variables” section in this chapter).

Database Connections

The process connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	AC1_CONTROL	Insert
Customer	<ul style="list-style-type: none"> ■ APE1_SUBSCR_DATA ■ APE1_SUBSCR_OFFERS ■ APE1_SUBSCR_PARAMS 	Select
Reference	ADJ1_OUTCOL_PROVIDER	Select
Usage	<ul style="list-style-type: none"> ■ APE1_RATED_EVENT ■ APR1_OUTCOL_CTRL 	<ul style="list-style-type: none"> ■ Select, Update ■ Select, Update, Insert

Admin Commands

The Outcollect Prepare job supports the following admin commands. For information on how to execute the commands, see the “Handling Admin Commands” section in the “High Availability” chapter.

Command	Description	Parameters
SET_LIGHT_TRACER_COMMAND	Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters	<ul style="list-style-type: none"> ■ Activation parameters: <ul style="list-style-type: none"> • <i>turnTracer “on/off”</i> – Activates or stops the Light Tracer • <i>--turnAssert “on/off”</i> – Activates or stops Assertions • <i>--turnLogOnTrace “on/off”</i> – Activates or stops writing log messages in Light Tracer files • <i>--turnFlowTracer/turnAll “on/off”</i> – Activates or stops tracing a single flow or event that is transferred between machines • <i>--turnAll “on/off”</i> – Activates or stops all of the above. The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter). ■ <i>--moduleID “ALL”</i> ■ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • <i>--threadInfo “Y/N”</i> – Specifies whether trace or assertion messages include the thread information. • <i>--timestampInd “Y/N”</i> – Specifies whether trace or assertion messages include the time stamp. • <i>--contextInd “Y/N”</i> – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • <i>--noOfMessages “<number>”</i> – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down. • <i>--period “5/10/20/60/120/480/1440”</i> – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated.

Command	Description	Parameters
		 Note: If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached. <ul style="list-style-type: none"> • <code>--assertAbortInd "Y/N"</code> – Specifies whether the application is to be aborted if the assertion condition fails. ▪ <code>--commandType "changeBufferSize"</code> – Resizes the Light Tracer buffer at run time. <ul style="list-style-type: none"> • <code>--bufferSize "<number>"</code> – The new size for the Light Tracer buffer.
SET_TRACE_SQL_COMMAND	<p>Activates or stops SQL tracing at runtime. The command name with its parameters must be enclosed in quotation marks.</p> <p><i>Example:</i> <code>ADJ1_Send_Admin_Command_Sh illin1027 <process instance> "SET_TRACE_SQL_COMMAND --traceSqlOn \"yes\" --duration \"1\\""</code></p>	<code>--traceSqlOn "<yes or no>" --duration "<minutes>"</code> If the IS_DURATION_REQUIRED environment variable is set to 'Y', the --duration parameter is mandatory. If the duration specified in the command exceeds the allowed maximum value, SQL tracing is activated for the maximum allowed period of time only (60 minutes).  Note: The SHUT_DOWN_CHECK_FREQ parameter of the Light Tracer in the ADJ1_CONF_SECTION_PARAM table determines the frequency (defined in the number of messages) at which the SQL Trace Manager checks whether the period specified in the --duration parameter has passed. Therefore, SQL tracing is stopped after the specified duration or after this number of messages has been reached, whichever comes later.
SWITCH_LIGHT_TRACER_ON ALL	Activates the Light Tracer without the ability to specify the mode or send parameters.	N/A

Command	Description	Parameters
SWITCH_LIGHT_TRACER_OFF ALL	Stops the Light Tracer	N/A
CPF_GracefulShutdownCommand	Shuts down the Outcollect Prepare job gracefully	N/A

Examples:

- ADJ1_Send_Admin_Command_Sh hpbl820 OUTCOL 148 SET_LIGHT_TRACER_COMMAND –turnTracer “on” –moduleID “ALL” –threadInfo “y”
- ADJ1_Send_Admin_Command_Sh hpbl820 OUTCOL 148 SWITCH_LIGHT_TRACER_OFF ALL

Recovery Instructions

The Outcollect Prepare job deletes the output files that have been written but have not been registered in Audit & Control. Because files are registered in Audit & Control and the rows in the APE1_RATED_EVENT table are updated in one commit operation, the events in deleted files will be handled again the next time the process runs.

Important Remarks

- Providers under the same group must belong to cycles in the same Usage database connection.
- When providers that belong to same group are handled by different Outcollect Prepare instances, and one of the instances had finished before the other one started, the Merge & Sort (ADJ1MAS) process does not merge the output files created by the different instances for these providers. This is because each Outcollect Prepare instance reads a different LAST_FILE_SEQ parameter for this group as the initial value, and the instances may use a different range of file sequences.

To prevent this situation from happening, do one of the following:

- Do not configure multiple providers in the same group.
- Configure the providers that belong to the same group to be in the same partition. This way, their customer segments are in the same customer segment range in the APE1_RATED_EVENT table partitions.
- Configure the providers in the same group to have the same cycle code and run each Outcollect Prepare instance on different cycle codes.

40 ADJ1_AP4_CycleBillRun_Sh – Turbo Charging Cycle Bill Run

This chapter describes the Turbo Charging Cycle Bill Run script.

Description

The ADJ1_AP4_CycleBillRun_Sh script is a generic script that triggers all Turbo Charging operational maps (except for Bill Confirmation). This inner script is activated by the Request Listener process of Amdocs Invoicing.

Run Frequency

Every cycle

Participation in Generic Maps

This section contains information on the generic maps in which the process participates.

Map Name

The process triggers the following operational maps:

- End-of-Cycle (EOC)
- Undo
- Rerun
- QA

For more information about these maps, see *Amdocs Billing Operator Guide*.

Severity Level

N/A

Activation

Command line:	ADJ1_AP4_CycleBillRun_Sh <Map key> <Run mode> <Cycle code> <Cycle instance> <Cycle year> <Flow ID> <Run request ID> <Cycle sequence number>
Amdocs Monitoring & Control:	No
Script Name:	ADJ1_AP4_CycleBillRun_Sh
Executable Name:	N/A

Example:

ADJ1_AP4_CycleBillRun_Sh 000120081200000000000000690FULL_RUN
FULL_RUN 1 12 2008 BLPREP 75 690

Preceding Processes

The following process must run successfully before this process is activated:

- *RQLSNR* – Request Listener of Amdocs Invoicing

For more information about this process, see *Amdocs Invoicing Run Book*.

Dependent Processes

This script inserts information into Operational tables that is required for the processes that participate in Turbo Charging operational maps.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The log files follow this naming convention:

- *Operational log files:*

ADJ1_AP4_CycleBillRun_Sh_Error_<Date>_<Time>.log

Example:

ADJ1_AP4_CycleBillRun_Sh_Error_20091116_130927.log

Location

The log files are located in the following directory:

- *Operational log files* – \$ABP_LOG

Contents

The log files contain the following:

- *Operational log files* – The output of operational scripts

Flow

The flow ID and the run mode determine which Turbo Charging operational map is generated and activated.

In the Bill on Demand (BOD) flow, Amdocs Invoicing does not trigger a Turbo Charging operational map.

When triggering rerate at the cycle level, the Request Listener does not create a file for Turbo Charging because the cycle information is sent via the command line arguments to the ADJ1_AP4_CycleBillRun_Sh script. However, when triggering rerate at the customer level (in the case of Undo), the Undo Initiator still sends a file to Turbo Charging with the list of customers to be rerated.

Environment Variables

The Turbo Charging Cycle Bill Run script populates the following environment variable according to input parameters.

Variable Name	Description	Valid Values/ Default Value
FILE_UNION_FOR_UNDO	The value of the file union for the Undo map	Values are: ■ <i>FCURER</i> – FULL CYCLE UNDO ■ <i>RERUERR</i> – Other UNDO map modes (FULL_RUN and RUN)

Parameters

The Turbo Charging Cycle Bill Run script takes the following parameters.

Parameter	Description	Default Value
cycleCode	The cycle code	N/A
cycleInstance	The cycle instance	N/A
cycleSeqNo	The sequence number of the cycle in the BL1_CYCLE_CONTROL table	N/A
cycleYear	The cycle year	N/A
flowID	The flow that is being run (Bill Preparation, Extracts, Confirmation, and so on)	N/A
mapKey	The prefix of the data group	N/A
runMode	The mode of the Bill Preparation flow	N/A
runRequestId	The ID of the request in the BL1_RUN_REQUEST table	N/A

Database Connections

The script connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Configuration	OP_APP_DATA	Insert, Delete
Configuration	OPHOST	Insert, Delete
Configuration	OPPAR	Insert, Delete
Configuration	OPPARDB	Insert, Delete

Recovery Instructions

Rerun the script.

41 ADJ1MAPCREATE – Create Map

This chapter describes the Create Map job.

Description

The Create Map job enables the recovery of the operational map creation process. It is created together with the map at the beginning of the process. If one of the other stages in the map creation fails, the operator can run the Create Map job with the relevant map key. The job executes the ADJ1_APRL_CycleBillRun_Manual_Sh script and re-creates the operational map according to the parameters received from Amdocs Invoicing.

Process Type

Batch job

Run Frequency

By request, every time there is a need to re-create an operational map

Participation in Generic Maps

The Create Map job can re-create the following operational maps:

- End-of-Cycle (EOC)
- QA
- Rerun
- Undo

For more information about these maps, see *Amdocs Invoicing Run Book*.

Activation

Command line:	RunJobs ADJ1MAPCREATE MAP_KEY
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_APRL_CycleBillRun_Manual_Sh
Executable Name:	N/A

Preceding Processes

For the Create Map job with all the relevant parameters to become available in the Operational tables, the relevant operational map must be triggered by Amdocs Invoicing and executed. The job is created together with the map.

Dependent Processes

This job inserts information into Operational tables. This information is required for the processes that participate in Turbo Charging operational maps.

If a map has failed, its jobs can be run only after the successful completion of the Create Map stage in the operational map execution process.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the console log files are:

`ADJ1_APRL_CycleBillRun_Manual_Sh_<Date>_<Time>.log`

Example:

`ADJ1_APRL_CycleBillRun_Manual_Sh_20091116_130927.log`

Location

Log files are located in the cdlog directory. Map creation log files are located in the \$ABP_APRL_ROOT/log directory.

Contents

The log files contain the job error information and log messages.

Flow

If the map creation stage of a Turbo Charging operational map fails, the operator must re-create the map manually using the Amdocs Monitoring & Control GUI or from the command line. To do so, the operator must run the ADJ1MAPCREATE job with the map key that corresponds to the failed map.

The job deletes the specific Turbo Charging operational map and then re-creates it as requested by Amdocs Invoicing.

Parameters

The following parameters must be set for the job to run properly.

Parameter	Description	Default Value
CYCLE_CODE	The cycle code	N/A
CYCLE_INST	The cycle instance	N/A
CYCLE_SEQ	The sequence number of the cycle in the BL1_CYCLE_CONTROL table	N/A
CYCLE_YEAR	The cycle year	N/A
FLOW_ID	The flow that is being run (Bill Preparation, Extracts, Confirmation, and so on)	N/A
MAP_KEY	The prefix of the data group	N/A
RUN_MODE	The mode of the Bill Preparation flow (for example, FULL_RUN)	N/A
REQUEST_ID	The ID of the request in the BL1_RUN_REQUEST	N/A

Database Connections

The script connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Configuration	APR1_CYC_JOB_RELATION_CFG	Select
Configuration	APR1_JOB_PARAMS_CFG	Select
Configuration	OP_APP_DATA	Insert, Delete
Configuration	OPHOST	Insert, Delete
Configuration	OPPAR	Insert, Delete
Configuration	OPPARDB	Insert, Delete

Recovery Instructions

Rerun the job.

42 ADJ1ACGRPLSTN – Accumulator Group Listener

This chapter describes the Accumulator Group Listener job.

Description

The Accumulator Group Listener job checks whether all invoicing groups for the Customer Group Accumulators Extract (ADJ1RCPEXT) have been published into the AC1_CONTROL table.

Process Type

Batch job

Run Frequency

Every cycle

Participation in Generic Maps

This section contains information on the generic maps in which the process participates.

Map Name

The process triggers the following operational maps:

- End-of-Cycle (EOC)
- Rerun

For more information about these maps, see *Amdocs Billing Operator Guide*.

Severity Level

N/A

Activation

Command line:	RunJobs ADJ1ACGRPLSTN BYREQ
Amdocs Monitoring & Control:	No
Script Name:	ADJ1_PIGroupListenerInvoker_Job_Sh
Executable Name:	bl1grplsnr

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Preceding Processes

In the End-of-Cycle map, this job runs after the successful completion of the following process:

- *ADJ1RRPOPREP* – Rerate Population Report

Dependent Processes

The following process can be run only after the successful completion of this process:

- *ADJ1PEFNSHNTF* – Accumulator Extract Finish Notification

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The logs use the following naming convention:

BL1GRPLSNR_<Date>_<Time>_1.log

Location

Log files are located in the following directory:

- *\$ABP_BL_ROOT/log*

The location and name of the logs can be configured by changing the following entries in the *BL1_CONF_SECTION_PARAM* table:

- *PARAM_NAME* = Filename
- *SECTION_NAME* = Logger.output.SIMPLEFILE

Contents

The file contains the debug and trace information, which is provided along with the source file and the functions that generated the message.

Flow

The Accumulator Group Listener job performs the following:

1. Monitors the *BL1_GROUP_STATUS* table of Amdocs Invoicing using the cycle sequence number and run request ID for records inserted by the Invoicing Population Extract (*BLPOPX*).
2. Waits until the status of all the invoicing groups becomes Finished, which means that the groups have been published into the *AC1_CONTROL* table.
3. Shuts down. The shutdown of this process serves as a trigger for the next-in-line Turbo Charging process (such as the Accumulator Extract Finish Notification) to start processing the files created by the Amdocs Invoicing process.

Parameters

The following job parameters must be set for the job to run properly.

Parameter	Description	Default Value
CYCLE_SEQ_NO	The sequence number of the cycle in the BL1_CYCLE_CONTROL table	N/A
FLOW_ID	The flow that is being run (Bill Preparation, Extracts, Confirmation, and so on)	N/A
RUN_MODE	The mode of the Bill Preparation flow	N/A
RUN_REQUEST_ID	The request ID in the BL1_RUN_REQUEST table	N/A

Database Connections

The job connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	<ul style="list-style-type: none">▪ BL1_REQUEST_MSG▪ BL1_CYCLE_GROUPS	Select

Recovery Instructions

Rerun the job.

43 ADJ1EVGRPLSTN – Event Group Listener

This chapter describes the Event Group Listener job.

Description

The Event Group Listener job checks whether all invoicing groups of customers for the Optimized Customer Group Event Extract Post Rerate (ADJ1POCEEXT) have been published into the AC1_CONTROL table.

Process Type

Batch job

Run Frequency

Every cycle

Participation in Generic Maps

The process triggers the following operational maps:

- End-of-Cycle (EOC)
- Rerun
- QA

For more information about these maps, see *Amdocs Billing Operator Guide*.

Activation

Command line:	RunJobs ADJ1EVGRPLSTN BYREQ
Amdocs Monitoring & Control:	No
Script Name:	ADJ1_EventGroupListenerInvoker_Job_Sh
Executable Name:	bl1grplsnr

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Preceding Processes

In the End-of-Cycle map, this job runs after the successful completion of the following processes:

- *ADJ1RRPEOC* – Rerate Prepare for End of Cycle
- *ADJ1PEFNSHNTF* – Accumulator Extract Finish Notification

In the Rerun map, this job runs after the successful completion of the following process:

- *ADJ1PEFNSHNTF* – Accumulator Extract Finish Notification

Dependent Processes

In the End-of-Cycle map, the following processes run after the successful completion of this process:

- *ADJ1POCEEXT* – Optimized Customer Group Event Extract Post Rerate
- *ADJ1PRCAEXT* – Customer Group Accumulators Extract Post Rerate
- *ADJ1EXTCLEANE* – Extract Clean-up
- *ADJ1MAS* – Merge and Sort

In the Rerun and QA maps, the following processes run after the successful completion of this process:

- *ADJ1OCEEXT* – Optimized Customer Group Event Extract
- *ADJ1RCAEXT* – Customer Group Accumulators Extract

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The logs use the following naming convention:

BL1GRPLSNR_<Date>_<Time>_1.log

Location

Log files are located in the following directory:

- \$ABP_BL_ROOT/log

The location and name of the logs can be configured by changing the following entries in the BL1_CONF_SECTION_PARAM table:

- PARAM_NAME = Filename
- SECTION_NAME = Logger.output.SIMPLEFILE

Contents

The file contains the debug and trace information, which is provided along with the source file and the functions that generated the message.

Flow

The Event Group Listener job performs the following:

1. Monitors the BL1_GROUP_STATUS table of Amdocs Invoicing using the cycle sequence number and run request ID for records inserted by the Event Population Extract (EVTPOPX).
2. Waits until the status of all the invoicing groups becomes Finished, which means that the groups have been published into the AC1_CONTROL table.
3. Shuts down. The shutdown of this process serves as a trigger for the next-in-line Turbo Charging processes to start processing the files created by the Amdocs Invoicing process.

Parameters

The following job parameters must be set for the job to run properly.

Parameter	Description	Default Value
CYCLE_SEQ_NO	The sequence number of the cycle in the BL1_CYCLE_CONTROL table	N/A
FLOW_ID	The flow that is being run (Bill Preparation, Extracts, Confirmation, and so on)	N/A
RUN_MODE	The mode of the Bill Preparation flow	N/A
RUN_REQUEST_ID	The request ID in the BL1_RUN_REQUEST table	N/A

Database Connections

The job connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	■ BL1_REQUEST_MSG ■ BL1_CYCLE_GROUPS	Select

Recovery Instructions

Rerun the job.

44 ADJ1PEFNSHNTF – Accumulator Extract Finish Notification

This chapter describes the Accumulator Extract Finish Notification job.

Description

The Accumulator Extract Finish Notification job informs the Customer Group Accumulators Extract that it can exit if there are no more files found in Audit & Control.

Process Type

Batch job

Run Frequency

Every cycle

Participation in Generic Maps

The process triggers the following operational maps:

- End-of-Cycle (EOC)
- Rerun

For more information about these maps, see *Amdocs Billing Operator Guide*.

Activation

Command line:	RunJobs ADJ1PEFNSHNTF BYREQ
Amdocs Monitoring & Control:	No
Script Name:	ADJ1_PI_Extr_Finish_Ntf_Job_Sh
Executable Name:	NA

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Preceding Processes

The following job must run successfully before this process is activated:

- *ADJ1ACGRPLSTN* – Accumulator Group Listener

Dependent Processes

The following job runs after the successful completion of this process:

- *ADJ1EVGRPLSTN* – Event Group Listener

Affected Applications

The process affects the Customer Group Accumulators Extract (ADJ1RCPEXT). This process starts working as soon as the first invoicing group is published. When no more invoicing groups remain in the AC1_CONTROL table, the Customer Group Accumulator Extract checks the APR1_OP_MAP_EVENTS table, which is updated by the Accumulator Extract Finish Notification job. It can exit only when all the groups have been published and therefore, there are no more groups for the Extract to handle.

Flow

The Accumulator Extract Finish Notification job inserts a special record to APR1_OP_MAP_EVENTS table informing the Customer Group Accumulators Extract that it can exit when there are no more files found in Audit & Control.

Parameters

The following job parameters must be set for the job to run properly. When the job runs as part of a map, the Turbo Charging Cycle Bill Run script populates these parameters automatically. For more information on this script, see the “ADJ1_AP4_CycleBillRun_Sh – Turbo Charging Cycle Bill Run” chapter.

Parameter	Description	Default Value
CYCLE_CODE	The cycle code.	N/A
CYCLE_INST	The cycle instance.	N/A
CYCLE_YEAR	The cycle year.	N/A
MAP_MODE	The context (map) in which the process is currently running. Valid values are: <ul style="list-style-type: none">■ NONE (if the job is run manually and not as part of a map)■ EOC■ RERUN■ QA■ UNDO	N/A

Database Connections

The job connects to the following table during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	APR1_OP_MAP_EVENTS	Insert

Recovery Instructions

Rerun the job.

45 ADJ1CYCLESTAT – Cycle State

This chapter describes the Cycle State process.

Description

Because the Event Server processes heavily rely on data from the ADJ1_CYCLE_STATE table, it is crucial to populate this table correctly.

The ADJ1CYCLESTAT job ensures proper initial population of the ADJ1_CYCLE_STATE table by doing the following based on the LOGICAL_DATE:

- Ensuring that at least one historical cycle exists
- Ensuring that the current open cycle exists
- Ensuring that at least two future cycles exist as required for Event Server processes

The job operates in two modes:

- *Full* – Works on all cycles defined in the ADJ_CYCLESTAT table
- *Cycle* – Receives the CYCLE_CODE as an input parameter

This job supports both monthly and weekly cycle frequencies, and a variety of frequency multipliers.

Process Type

Batch job

Run Frequency

By request. This job must run as part of a sanity check for an environment.

Activation

Command Line:	<ul style="list-style-type: none">■ RunJobs ADJ1CYCLESTAT ALL■ RunJobs ADJ1CYCLESTAT CYCLE1■ RunJobs ADJ1CYCLESTAT CYCLE2■ RunJobs ADJ1CYCLESTAT CYCLE3■ RunJobs ADJ1CYCLESTAT CYCLE4
Amdocs Monitoring & Control:	No
Script Name:	ADJ1_CycleState_Job_Sh
Executable Name:	N/A

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Affected Applications

The process affects the following application:

- *Event Server* – The main processing daemon of the Turbo Charging event processing platform

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Log files are created and handled in the default manner.

Name

- *Log files:*

ADJ1CYCLESTAT_<Date>_<Time>.log

Example:

ADJ1CYCLESTAT_20090624_115035.log

- *Console log files:*

ADJ1_CycleState_Job_inner_Sh_<Date>_<Time>.log

Example:

ADJ1_CycleState_Job_inner_Sh_20081109_130927.log

- *Operational log files:*

ADJ1CYCLESTAT_<Job Record Name>_\${ABP_MARKET}_<Date>_<Time>.log

Example:

ADJ1CYCLESTAT_ALL_M3G_20090624_114834.log

Location

Log files are located in the following directories:

- *Log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands)
- *Operational log files* – The output of operational scripts

Flow

The Cycle State process executes the following steps:

1. Gets the logical date from the LOGICAL_DATE table.

```
SELECT TO_CHAR(LOGICAL_DATE, 'DD/MM/YYYY')
FROM   LOGICAL_DATE
WHERE  LOGICAL_DATE_TYPE='R';
```

2. If the job runs in Full mode, gets all cycle data from the ADJ1_CYCLES table. If the job runs in Cycle mode, gets only the data of the required cycle.

```
SELECT CYCLE_CODE, CYCLE_FREQUENCY, FREQUENCY_MULTIPLIER,
       CLOSE_DAY, CUST_DB_CONNECT_CODE, USAGE_DB_CONNECT_CODE
  FROM  ADJ1_CYCLES
 ORDER BY CYCLE_CODE ASC;
```

3. Per cycle code, checks whether a record that is valid for the Logical Date exists in the ADJ1_CYCLE_STATE table.

- If the record is missing, writes a warning message to the log and skips the next cycle.
- Otherwise:
 - i. Handles the previous cycle instance:
 - ◆ If a record for the previous cycle instance does not exist in the ADJ1_CYCLE_STATE table, creates a new record based on the current cycle instance's START_DATE and END_DATE.
 - ◆ If such a record exists, updates it and sets the following parameters:
 - ◆ LOCKED_FOR_NEW='Y'
 - ◆ LOCKED_FOR_UPDATE='Y'
 - ◆ MAINTENANCE_STATUS='C'
 - ii. Handles future cycle instances and verifies that there are two open future records in the ADJ1_CYCLE_STATE table. If one of the records does not exist, creates it.

Parameters

The following parameters must be set for the process to run properly.

Parameter	Description	Default Value	Method of Changing the Parameter Manually
CYCLE_CODE	The cycle code	None	Operational job parameter. Can be updated through Amdocs Monitoring & Control.
MIN_REQ_NUM_OF_PREV_CYCLES	Minimum required number of previous cycles	1	Operational job parameter. Can be updated through Amdocs Monitoring & Control.
MIN_REQ_NUM_OF_FUTURE_CYCLES	Minimum required number of future cycles	2	Operational job parameter. Can be updated through Amdocs Monitoring & Control.

The GN1_TASK_CONNECT table must contain an entry with the following values:

- TaskName=“ADJ1CYCLESTAT”
- RunMode=“F”
- SessionArg=“DEFAULT”

In addition, the GN1_CONNECT_PARAM table must contain a corresponding entry.

Database Connections

The process connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	ADJ1_CYCLE_STATE	Update, Insert
Reference	ADJ1_CYCLES	Select

Important Remarks

- In some cases, where the ADJ1_CYCLES.cycle_close_day is 29, 30, or 31, the record for the previous (closed) cycle in the ADJ1_CYCLE_STATE table is created with the wrong START_DATE. Because the cycle is closed, this does not impact on the process.
- The process uses the ADJCONFIG connection code (otherwise, the process fails).

46 ADJ1CYCLEMNT – Cycle Maintenance

This chapter describes the Cycle Maintenance process.

Description

The main scope of the process is to:

- Receive commands from external sources pertaining to changes in the cycle area
- Update the Turbo Charging Cycle States (ADJ1_CYCLE_STATE) table
- Inform all parties that need to know of the new state of the ADJ1_CYCLE_STATE table
- Check whether the relevant processes have executed the Refresh Cycle State command and determine the rest of the flow based on this information

Process Type

Batch job

Run Frequency

End of cycle

Participation in Generic Maps

This section contains information on the generic maps in which the process participates.

Map Name

End-of-Cycle (EOC)

For more information about this map, see *Amdocs Billing Operator Guide*.

Severity Level

Severity level 2 – The map must hold until the job is fixed.

Activation

Command Line:	RunJobs ADJ1CYCLEMNT <APPLICATION_ID>
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_CYCLEMAINT_Job_Sh
Executable Name:	gcpf1fwcApp

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

When the job is done, it shuts down on its own.

Dependent Processes

The following processes can be run only after the successful completion of this process:

- *ADJ1RRPOPREP* – Rerate Population Report
- *ADJ1EVENTSRV* – Event Server for End of Cycle

Affected Applications

The process affects the following applications that refresh the data from *ADJ1_CYCLE_STATE* table:

- Event Server
- Rerate Prepare
- Usage Query Proxy
- Usage Query Server

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Log files are created and handled in the default manner.

Name

The names of the log files are:

- *Log files:*
ADJ1CYCLEMNT_<APPLICATION_ID>_%D_%T_%n.log
Example:
ADJ1CYCLEMNT_CYC_MAINT1010_20081109_130927_1.log
- *Console log files:*
ADJ1_CYCLEMAINT_Job_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log
Example:
ADJ1_CYCLEMAINT_Job_inner_Sh_CYC_MAINT1010_20081109_130927.log
- *Operational log files:*
ADJ1CYCLEMNT _<Job Record Name>_\${ABP_MARKET}_<Date>_<Time>.log
Example:
ADJ1CYCLEMNT_BYREQ_M3G_20081109_130927.log
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter)

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information.
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).
- *Operational log files* – The output of operational scripts.
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Flow

The Cycle Maintenance process executes the following flow:

1. Handles the input parameters as follows:

- For LOCK_FOR_NEW, Cycle Code, Cycle Instance, and Cycle Year, executes the following statement:

```
Update ADJ1_CYCLE_STATE
Set "Locked_for_new" = "Y"
Where "Cycle_code" = <input cycle code> and
"Cycle_instance" = <input cycle instance> and
"Cycle_year" = <input cycle year>
```

- For LOCK_FOR_UPD, Cycle Code, Cycle Instance, and Cycle Year, executes the following statement:

```
Update ADJ1_CYCLE_STATE
Set "Locked_for_Update" = "Y"
Where "Cycle_code" = <input cycle code> and
"Cycle_instance" = <input cycle instance> and
"Cycle_year" = <input cycle year>
```

- For CHG_MNT_STATUS, Cycle Code, Cycle Instance, Cycle Year and Maintenance Status executes the following statement:

```
Update ADJ1_CYCLE_STATE
Set "Maintenance_Status" = <input status>
Where "Cycle_code" = <input cycle code> and
"Cycle_instance" = <input cycle instance> and
"Cycle_year" = <input cycle year>
```

- For CHG_MNT_STATUS, Cycle Code, Cycle Instance, Cycle Year, and Maintenance Status executes the following statement:

```
Update ADJ1_CYCLE_STATE
Set "Maintenance_Status" = <input status>
Where "Cycle_code" = <input cycle code> and
"Cycle_instance" = <input cycle instance> and
"Cycle_year" = <input cycle year>
```

2. Sends the REFRESH_CYCLE_STATE admin command with the cycle code and cycle instance to the Availability Manager, which notifies all processes.
3. Checks whether the target processes have refreshed the cycle information in their memory.

When Geo Redundancy support is enabled, if the Cycle Maintenance process at the active updates or inserts a new row into the ADJ1_CYCLE_STATE table, it informs the standby site of the change by inserting a new row into the ADJ1_IS_STANDBY_UPDATED table at the active site. Both changes are replicated to the standby site. The “ADJ1UROPS – Update Refresh at Standby Site” job runs at the standby site to detect this change and trigger a Refresh Cycle State command via the Availability Manager to the Event Server and Usage Query Server.

Environment Variables

The following environment variable affects the process. This environment variable is defined in the ADJ1_CYCLEMAINT_Job_inner_Sh script.

Name	Description	Default Value
CYCLE_MAINT_WORKING_MODE	Specifies the working mode of the Cycle Maintenance process	MNT

Parameters

The following parameters must be set for the process to run properly.

Parameter	Description	Default Value
APPLICATION_ID	The process instance.	CYC_MAINT128
CCF_FILE_NAME	An XML file that contains the relevant configuration.	APR_CycleMaint.ccf
CMD	The command to execute. Can be one of the following: ■ LOCK_FOR_NEW ■ LOCK_FOR_UPDATE ■ CHG_MNT_STATUS	LOCK_FOR_NEW

Parameter	Description	Default Value
CYCLE_CODE	Cycle code used to update the relevant entry in the ADJ1_CYCLE_STATE table.	1
CYCLE_INST	Cycle instance used to update the relevant entry in the ADJ1_CYCLE_STATE table.	1
CYCLE_YEAR	Cycle year used to update the relevant entry in the ADJ1_CYCLE_STATE table.	2009
MAINTENANCE_STATUS	New maintenance status (used in the CHG_MNT_STATUS command). Can be one of the following: <ul style="list-style-type: none">■ A – Active■ B – Backed up■ C – Cleaned	None

Configuration Parameters

The Cycle Maintenance process has the following configuration parameters.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
CYC_MAINT	config	ENABLE_RESTART_PROCESSES	Y	Indicates whether the Event Servers must restart if they fail to refresh or do not respond to the refresh command. By default, restart is enabled. To disable it, this parameter must be added to the CFG1_CONF_SECTION_PARAM table with the ‘N’ value.

The process also uses Light Tracer configuration parameters, which are defined in the ADJ1_CONF_SECTION_PARAM table. For more information, see [Light Tracer Parameters](#) in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Configuration Files

The Cycle Maintenance process requires a configuration file (a .ccf file) that is defined via an internal configuration tool.

Database Connections

The process connects to the following table during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	ADJ1_CYCLE_STATE	Update
Application	ADJ1_IS_STANDBY_UPDATED	Update
TRB CL	TRB_SUB_LOG	Select

Admin Commands

The Cycle Maintenance process supports the following admin commands. For information on how to execute the commands, see the “Handling Admin Commands” section in the “High Availability” chapter.

Command	Description	Parameters
SET_LIGHT_TRACER_COMMAND	Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters	<ul style="list-style-type: none"> ▪ Activation parameters: <ul style="list-style-type: none"> • <i>turnTracer “on/off”</i> – Activates or stops the Light Tracer • <i>--turnAssert “on/off”</i> – Activates or stops Assertions • <i>--turnLogOnTrace “on/off”</i> – Activates or stops writing log messages in Light Tracer files • <i>--turnFlowTracer/turnAll “on/off”</i> – Activates or stops tracing a single flow or event that is transferred between machines • <i>--turnAll “on/off”</i> – Activates or stops all of the above. The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter). ▪ --moduleID “ALL” ▪ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • <i>--threadInfo “Y/N”</i> – Specifies whether trace or assertion messages include the thread information. • <i>--timestampInd “Y/N”</i> – Specifies whether trace or assertion messages include the time stamp. • <i>--contextInd “Y/N”</i> – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • <i>--noOfMessages “<number>”</i> – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down. • <i>--period “5/10/20/60/120/480/1440”</i> – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated.

Command	Description	Parameters
		 Note: If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached. <ul style="list-style-type: none"> • <code>--assertAbortInd "Y/N"</code> – Specifies whether the application is to be aborted if the assertion condition fails. ▪ <code>--commandType "changeBufferSize"</code> – Resizes the Light Tracer buffer at run time. • <code>--bufferSize "<number>"</code> – The new size for the Light Tracer buffer.
SET_TRACE_SQL_COMMAND	Activates or stops SQL tracing at runtime. The command name with its parameters must be enclosed in quotation marks. <i>Example:</i> <pre>ADJ1_Send_Admin_Command_Sh illin1027 <process instance> "SET_TRACE_SQL_COMMAND --traceSqlOn \"yes\" --duration \"1\""</pre>	<code>--traceSqlOn "<yes or no>" --duration "<minutes>"</code> If the IS_DURATION_REQUIRED environment variable is set to 'Y', the --duration parameter is mandatory. If the duration specified in the command exceeds the allowed maximum value, SQL tracing is activated for the maximum allowed period of time only (60 minutes).  Note: The SHUT_DOWN_CHECK_FREQ parameter of the Light Tracer in the ADJ1_CONF_SECTION_PARAM table determines the frequency (defined in the number of messages) at which the SQL Trace Manager checks whether the period specified in the --duration parameter has passed. Therefore, SQL tracing is stopped after the specified duration or after this number of messages has been reached, whichever comes later.
SWITCH_LIGHT_TRACER_ON ALL	Activates the Light Tracer without the ability to specify the mode or send parameters.	N/A
SWITCH_LIGHT_TRACER_OFF ALL	Stops the Light Tracer	N/A

Examples:

- `ADJ1_Send_Admin_Command_Sh hpbl820 CYC_MAINT 1010 SET_LIGHT_TRACER_COMMAND -turnTracer "on" – moduleId "ALL" –threadInfo "y"`
- `ADJ1_Send_Admin_Command_Sh hpbl820 CYC_MAINT 1010 SWITCH_LIGHT_TRACER_OFF ALL`

Troubleshooting

The following error can occur during the process run.

Error	Preventing Action
Missing parameter	Ensure that the required parameters are populated as described in the “Parameters” section of this chapter.

Recovery Instructions

If an error occurs while the Cycle Maintenance process is handling a transaction, the error is reported to the Transaction Broker. The Error Handling mechanism of the Transaction Broker is responsible for the correction and recycling of the transaction.

47 ADJ1CYCMNTEOD – Cycle Maintenance EOD

This chapter describes the Cycle Maintenance End-of-Day (EOD) process.

Description

The Cycle Maintenance EOD process handles requests from the Transaction Broker and inserts new entries into the ADJ1_CYCLE_STATE table.

Process Type

Batch job

Run Frequency

Daily

Participation in Generic Maps

This section contains information on the generic maps in which the process participates.

Map Name

End-of-Day (EOD)

For more information about this map, see *Amdocs Billing Operator Guide*.

Severity Level

Severity level 2 – The map must hold until the job is fixed.

Activation

Command Line:	RunJobs ADJ1CYCMNTEOD END_OF_DAY
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_CYCLEMAINTEOD_Job_Sh
Executable Name:	gcpf1fwcApp

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

When the job is done, it shuts down on its own.

Preceding Processes

The following job must run successfully before this process is activated:

- *BL1CYCLEMAINT* – Amdocs Invoicing Cycle Maintenance

For more information on this job, see *Amdocs Invoicing Run Book*.

Affected Applications

The process affects the following applications that refresh the data from ADJ1_CYCLE_STATE table:

- Event Server
- Rerate Prepare
- Usage Query Proxy
- Usage Query Server

Log Files

It is recommended to check the log file after each run to verify that the process has succeeded.

Log files are created and handled in the default manner.

Name

The names of the log files are:

- *Log files:*

ADJ1CYCMNTEOD_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1CYCMNTEOD_CYC_MAINTEOD109_20081109_130927_1.log

- *Console log files:*

ADJ1_CYCLEMAINTEOD_Job_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1_CYCLEMAINTEOD_Job_inner_Sh_CYC_MAINT1010_20081109_130927.log

- *Operational log files:*

ADJ1CYCMNTEOD_<Job Record Name>_\${ABP_MARKET}_<Date>_<Time>.log

Example:

ADJ1CYCMNTEOD_BYREQ_M3G_20081109_130927.log

Location

Log files are located in the following directories:

- *Log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG

For information about the configuration parameters that define the location of log and trace files, see the “[Configuration Parameters](#)” section in the “[ADJ1EVENTSRV – Event Server](#)” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands)
- *Operational log files* – The output of operational scripts

Flow

The Cycle Maintenance EOD process handles requests from the Transaction Broker as follows:

1. Connects to the Transaction Broker as an EPCYCLE_UPD subscriber. The Cycle Maintenance process subscribes to the NEW_CYCLE_INSTANCE activity (as specified in the TRB1_APPL_DEF table).
2. Retrieves the transactions. For each transaction received (0...n):
 - a. Analyzes the records.
 - b. For each cycle instance entry:
 - i. Creates an entry in the Turbo Charging Cycle States (ADJ1_CYCLE_STATE) table.
 - ii. Adds information from the Turbo Charging Cycles table (ADJ1_CYCLES) for the requested cycle.
 - iii. Inserts data into the database. See *Turbo Charging Data Model*.
3. When all the transactions have been handled, commits them to the database.

When Geo Redundancy support is enabled, the Cycle Maintenance EOD process performs the same flow as the “[ADJ1CYCLEMNT – Cycle Maintenance](#)” process.

Environment Variables

The following environment variable affects the process. This environment variable is defined in the ADJ1_CYCLEMAINT_Job_inner_Sh script.

Name	Description	Default Value
CYCLE_MAINT_WORKING_MODE	Specifies the working mode of the Cycle Maintenance process	TRB

Parameters

The following parameters must be set for the process to run properly.

Parameter	Description	Default Value
APPLICATION_ID	The process instance.	CYC_MNTEOD109
CCF_FILE_NAME	An XML file that contains the relevant configuration.	APR_CycleMaint.ccf

Configuration Files

The Cycle Maintenance EOD process requires a configuration file (a .ccf file) that is defined via an internal configuration tool.

Database Connections

The process connects to the following table during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	ADJ1_CYCLE_STATE	Update, Insert
Application	ADJ1_IS_STANDBY_UPDATED	Update
TRB CL	TRB_SUB_LOG	Select

Troubleshooting

The following error can occur during the process run.

Error	Preventing Action
Cannot register to the Transaction Broker	Ensure that the Transaction Broker is configured for this job.

Recovery Instructions

Rerun the job.

48 ADJ1UROPS – Update Refresh at Standby Site

This chapter describes the Update Refresh at Standby Site job.

Description

If Geo Redundancy is enabled, the Update Refresh at Standby Site job checks for changes in the ADJ1_CYCLE_STATE table by monitoring the ADJ1_IS_STANDBY_UPDATED table.

When Geo Redundancy support is enabled, if the Cycle Maintenance process updates or inserts a row into the ADJ1_CYCLE_STATE table, a new row is also inserted into the ADJ1_IS_STANDBY_UPDATED table at the active site. Both changes are replicated to the standby site. The Update Refresh at Standby Site job monitors the ADJ1_IS_STANDBY_UPDATED table at the standby site. If the job finds an entry with the ‘N’ (New), ‘F’ (Failure), or ‘P’ (Processing) status, it sends a command through the Availability Manager to the standby Event Server and Usage Query Server to refresh the cycle state.

Process Type

Batch job

Run Frequency

By request

Activation

Command Line:	RunJobs ADJ1UROPS BYREQ
Amdocs Monitoring & Control:	No
Script Name:	ADJ1_UpdateRefreshOnPassivSite_Sh
Executable Name:	N/A

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

ADJ1_UpdateRefreshOnPassivSite_sh.log

Location

\${ABP_LOG}

Contents

The log files contain the process error information.

Flow

If Geo Redundancy is enabled, the Update Refresh at Standby Site job runs periodically at the standby site and performs the following tasks:

1. Checks whether the site is standby. If the site is active, exits.
2. In the ADJ1_IS_STANDBY_UPDATED table, deletes the entries in the ‘D’ (Done) status that are older than one day
3. Selects the entries in the ‘N’ (New) or ‘F’ (Failed) status and changes their status to ‘P’ (Processing)
4. Activates an Availability Manager job that sends the Refresh Cycle State admin command to the Event Server and Usage Query Server at the standby site
5. Gets the status of the command execution and updates the entries that were in the ‘P’ according to the return value: ‘F’ (Failed) or D (Done)

Database Connections

The process connects to the following table during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	ADJ1_IS_STANDBY_UPDATED	Update Delete

49 ADJ1_ConfirmCycle_sh – Turbo Charging Confirm Cycle

This chapter describes the Turbo Charging Confirm Cycle script.

Description

In the Bill Confirmation map, the pm1ConfirmCycle_sh script of Amdocs Invoicing runs with the cycle code, instance, year, and partition ID. This script calls an inner script, ADJ1_ConfirmCycle_sh. This Turbo Charging Confirm Cycle script closes (locks for update and insert) the given cycle instance in the ADJ1_CYCLE_STATE table.

Process Type

N/A

Run Frequency

Every cycle

Participation in Generic Maps

This section contains information on the generic maps in which the process participates.

Map Name

Bill Confirmation

For more information about this map, see *Amdocs Invoicing Run Book*.

Activation

Command line:	pm1ConfirmCycle_sh <cycle code> <cycle ID> <cycle year> <partition ID>
Amdocs Monitoring & Control:	No
Script Name:	pm1ConfirmCycle_sh
Executable Name:	N/A

Preceding Processes

The ADJ1_ConfirmCycle_sh script is called by the pm1ConfirmCycle_sh script.

Dependent Processes

The ADJ1_ConfirmCycle_sh script calls the ADJ1_EOC_USG_CTRL_Sh script.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

- *Operational log files:*

pm1ConfirmCycle_<Date>_<Time>.log

Example:

pm1ConfirmCycle_20081109_130927.log

Location

Log files are located in the following directory:

- *Operational log files – \$ABP_LOG*

Contents

The log files contain the following:

- *Operational log files – The output of operational scripts*

Flow

The Bill Confirmation script updates data of given cycle in ADJ1_CYCLE_STATE to mark it as closed.

Parameters

See the “Activation” section in this chapter.

Database Connections

The script connects to the following table during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	ADJ1_CYCLE_STATE	Update

Recovery Instructions

Rerun the script.

50 ADJ1_EOC_USG_CTRL_Sh – EOC Usage Control

This chapter describes the EOC Usage Control script.

Description

The ADJ1_EOC_USG_CTRL_Sh script compresses, moves, or truncates partitions of the APE1_RATED_EVENT table. It is called by the ADJ1_ConfirmCycle_sh script as part of the Bill Confirmation map, after a cycle has been marked as closed.

Run Frequency

Every cycle

Participation in Generic Maps

This section contains information on the generic maps in which the process participates.

Map Name

Bill Confirmation

For more information about this map, see *Amdocs Invoicing Run Book*.

Activation

Command line:	ADJ1_EOC_USG_CTRL_Sh.sh -CURRENT_CYCLE <CYCLE_CODE><CYCLE_INSTANCE> <CYCLE_YEAR> [additional options]
Amdocs Monitoring & Control:	No
Script Name:	ADJ1_EOC_USG_CTRL_Sh.sh
Executable Name:	N/A

Preceding Processes

The ADJ1_EOC_USG_CTRL_Sh script is called by the ADJ1_ConfirmCycle_sh script.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

ADJ1_EOC_USG_CTRL_<Date>_<Time>.log

Example:

ADJ1_EOC_USG_CTRL_20120205_124441.log

Location

Log files are located in the \$ABP_APB_ROOT/log directory.

Contents

The log files contain the script error information and log messages.

Flow

The EOC Usage Control script receives input from the command line and from the ADJ1_PROC_PARAMS table (see the “Parameters” section in this chapter).

The script procedures are located in ADJ1_PARTITION_PG package.

Compress and Move Operations

The parameters for compress and move operations are specified in the ADJ_PROC_PARAMS table. Partitions can be compressed or moved in the current tablespace or in a remote tablespace (according to the ADJ1_EOC_USG_TBS_CTRL table).

After parsing all the cycle-related parameters and before executing the compress or move operation on certain cycles, the script makes sure that for these cycles, the MARKED_FOR_UPDATE and MARKED_FOR_NEW fields of the ADJ1_CYCLE_STATE table are set to ‘Y’.

Compress and move operations are performed using Oracle’s EXCHANGE PARTITION feature while collecting statistics.

Truncate Operation

The parameters for the truncate operation are specified in the ADJ_PROC_PARAMS table. Partitions can be truncated in the current tablespace or in a remote tablespace (according to the ADJ1_EOC_USG_TBS_CTRL table).

Because data can be located in multiple storages, some partitions may be located in a different tablespace than others. In this case, the truncate operation uses the ADJ1_EOC_USG_TBS_CTRL table to obtain the original tablespace name for each cycle.

The truncate operation moves the partitions to the original tablespace and empties them.

After parsing all the cycle-related parameters and before executing the truncate operation on certain cycles, the script makes sure that for these cycles, the MARKED_FOR_UPDATE and MARKED_FOR_NEW fields of the ADJ1_CYCLE_STATE table are set to ‘Y’.

Environment Variables

The script uses the following environment variables.

Name	Description	Default Value
ADJ1_EOC_USG_CTRL_CO MPRESS_ENABLED	Indicates whether the end-of-cycle compress operations are enabled.	N
ADJ1_EOC_USG_CTRL_CO MPRESS_INPUT	<p>The cycle instances to compress. The cycles can be specified in one of the following formats:</p> <ul style="list-style-type: none"> ■ A range of cycle instances to compress, enclosed in square brackets: <code>[cycle_year=<CYCLE_YEAR>,cycle_code=<CYCLE_CODE>,range=<CYCLE_INSTANCE>-<CYCLE_INSTANCE>]</code> <i>Example:</i> <code>[cycle_year=2012, cycle_code=9, range=3-7]</code> ■ A set of cycle instances to compress, enclosed in square brackets: <code>[cycle_year=<CYCLE_YEAR>,cycle_code=<CYCLE_CODE>,cycle_instance = <CYCLE_INSTANCE>,<CYCLE_INSTANCE>, ...]</code> <i>Example:</i> <code>[cycle_year=2012, cycle_code=9, cycle_instance= 3,5,4,9]</code> ■ An automatically determined cycle instance to compress, according to the <code>ADJ1_EOC_USG_CTRL_COMPRESS_INTERVAL</code> variable: <code>[-1]</code> 	[-1]
ADJ1_EOC_USG_CTRL_CO MPRESS_INTERVAL	The number of cycle instances to go back from the current cycle instance in order to determine the cycle instance to compress.	4
ADJ1_EOC_USG_CTRL_CO MPRESS_PARALLEL_ENAB LED	Indicates whether partitions can be compressed in parallel.	Y
ADJ1_EOC_USG_CTRL_CTR L_TABLE_NAME	The control table for exchanging partitions. Each record in the table contains the cycle instance, cycle year, the target tablespace, and the original tablespace.	ADJ1_EOC_U SG_TBS_CTR L
ADJ1_EOC_USG_CTRL CU RRENT_TBS_ENABLED	Indicates whether to perform exchange operations in the current tablespace ('Y') or use a control table ('N').	Y
ADJ1_EOC_USG_CTRL_EX CHANGE_ENABLED	Indicates whether exchanging partitions is enabled.	N
ADJ1_EOC_USG_CTRL_IND X_CURRENT_TBS_ENABLE D	Indicates whether to perform index exchange operations in the current tablespace ('Y') or use a control table ('N').	Y
ADJ1_EOC_USG_CTRL_MO VE_ENABLED	Indicates whether the end-of-cycle move operations are enabled.	N

Name	Description	Default Value
ADJ1_EOC_USG_CTRL_MO VE_INPUT	<p>The cycle instances to move. The cycles can be specified in one of the following formats:</p> <ul style="list-style-type: none"> ■ A range of cycle instances to move, enclosed in square brackets: $[cycle_year=<CYCLE_YEAR>,cycle_code=<CYCLE_C ODE>,range=<CYCLE_INSTANCE>-<CYCLE_INSTANCE>]$ <i>Example:</i> $[cycle_year=2012, cycle_code=9, range=3-7]$ ■ A set of cycle instances to move, enclosed in square brackets: $[cycle_year=<CYCLE_YEAR>,cycle_code=<CYCLE_C ODE>,cycle_instance = <CYCLE_INSTANCE>,<CYCLE_INSTANCE>, ...]$ <i>Example:</i> $[cycle_year=2012, cycle_code=9, cycle_instance= 3,5,4,9]$ ■ An automatically determined cycle instance to move, according to the ADJ1_EOC_USG_CTRL_COMPRESS_INTERVAL variable: $[-1]$ 	[-1]
ADJ1_EOC_USG_CTRL_MO VE_INTERVAL	The number of cycle instances to go back from the current cycle instance in order to determine the cycle instance to move.	5
ADJ1_EOC_USG_CTRL_TR UNCATE_ENABLED	Indicates whether the end-of-cycle truncate operations are enabled.	N
ADJ1_EOC_USG_CTRL_TR UNCATE_INPUT	<p>The cycle instances to truncate. The cycles can be specified in one of the following formats:</p> <ul style="list-style-type: none"> ■ A range of cycle instances to truncate, enclosed in square brackets: $[cycle_year=<CYCLE_YEAR>,cycle_code=<CYCLE_C ODE>,range=<CYCLE_INSTANCE>-<CYCLE_INSTANCE>]$ <i>Example:</i> $[cycle_year=2012, cycle_code=9, range=3-7]$ ■ A set of cycle instances to truncate, enclosed in square brackets: $[cycle_year=<CYCLE_YEAR>,cycle_code=<CYCLE_C ODE>,cycle_instance = <CYCLE_INSTANCE>,<CYCLE_INSTANCE>, ...]$ <i>Example:</i> $[cycle_year=2012, cycle_code=9, cycle_instance= 3,5,4,9]$ ■ An automatically determined cycle instance to truncate, according to the ADJ1_EOC_USG_CTRL_COMPRESS_INTERVAL variable: $[-1]$ 	[-1]

Name	Description	Default Value
ADJ1_EOC_USG_CTRL_TRUNCATE_INTERVAL	The number of cycle instances to go back from the current cycle instance in order to determine the cycle instance to truncate.	6
ADJ1_EOC_USG_CTRL_USAGE_TABLE_NAME	The usage table for the end-of-cycle usage control (for example, APE1_RATED_EVENT).	APE1_RATED_EVENT

Parameters

The following parameters must be set for the process to run properly.

Command-Line Parameters

The following parameters are specified in the command line.

Parameter	Description	Default Value
-CURRENT_CYCLE <CYCLE_CODE> <CYCLE_INSTANCE> <CYCLE_YEAR>	Specifies the current cycle: cycle code, instance, and year. This is a mandatory parameter.	N/A
-h	Prints command line examples and additional information.	N/A
-DBG	Activates the debug mode with enhanced logging.	N/A

Database Parameters

For parameters that are specified in the ADJ1_PROC_PARAM table, see the “Environment Variables” section in this chapter.

Database Connections

The script connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	ADJ1_CYCLE_STATE	Select
Application	ADJ1_EOC_USG_TBS_CTRL	Select
Application	ADJ1_PROC_PARAM	Select
Usage	APE1_RATED_EVENT	Select

51 ADJ1TRUNC – EOD Truncate Usage

This chapter describes the EOD Truncate Usage job.

Description

The EOD Truncate Usage job can run as part of daily clean-up procedures to truncate old partitions of the APE1_RATED_EVENT table.

Process Type

Batch job

Run Frequency

By request, for example, as part of EOD clean-up

Participation in Generic Maps

This section contains information on the generic maps in which the process participates.

Map Name

The job can run as part of the EOD map.

For more information about this map, see *Amdocs Invoicing Run Book*.

Activation

Command line:	RunJobs ADJ1TRUNC ByReq
Amdocs Monitoring & Control:	No
Script Name:	ADJ1_Truncate_EOD_Job_Sh
Executable Name:	N/A

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*

ADJ1_EOD_TRUNCATE_USAGE\${APR_CURRENT_TIME}.log

Example:

ADJ1_EOD_TRUNCATE_USAGE20120513_090635.log

- *Operational log files:*

ADJ1TRUNC_<Job Record Name>_\${ABP_MARKET}<Date><Time>.log

Example:

ADJ1TRUNC_BYREQ_M3G_20081109_130927.log

Location

Log files are located in the following directories:

- *Log files* – \$ABP_APP_ROOT/log
- *Operational log files* – \$ABP_LOG

Contents

The log files contain the script error information and log messages.

Flow

The EOD Truncate Usage job executes ADJ1_EOD_Truncate_Usage_Sh.sh script, which performs the following steps:

1. Gets the input parameters:
 - The database connection string
 - The number of days to go back from the current date, defined in the ADJ1_PROC_PARAMS table (see the “Environment Variables” section in this chapter)
2. Performs validations
3. Runs a stored procedure, which performs the following actions:
 - a. Calculates the past date and its partition using the LOGICAL_DATE table and the environment variables
 - b. Connects to the database and truncates the relevant partition or partitions

Environment Variables

The job uses the following environment variable.

Name	Description	Default Value
EOD_BACKWARD_DROP_COUNT	The number of days to go back from the current date, which indicates the latest partition of the APE1_RATED_EVENT table to be truncated and dropped. All the partitions from this partition until the beginning of the table are dropped.	30
EOD_BACKWARD_TRUNCATE_COUNT	The number of days to go back from the current date, which indicates the partition of the APE1_RATED_EVENT table to be truncated. <i>Example:</i> If the current date is 05-10-2012, and the value of the environment variable is 4, the job truncates the partition for 05-06-2012.	3

Parameters

The following parameter must be set for the process to run properly.

Parameter	Description	Default Value
SESSION_ARG_TO_MATCH	The database connection string to be taken from the GN1_TASK_CONNECT and GN1_CONNECT_PARAMS tables.	N/A

Database Connections

The job connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	ADJ1_PROC_PARAM	Select
Application	LOGICAL_DATE	Select
Usage	APE1_RATED_EVENT	Select

52 ADJ1COMPUSG – Compress Usage

This chapter describes the Compress Usage process.

Description

After a cycle instance has expired (that is, has been moved to a read-only state and is closed for updates), the usage data can be compressed using the Oracle compression utility. Due to the fact that in Turbo Charging, the data in the Rated Events and Accumulators tables is fielded, it can be assumed that the compression ratio will be relatively high, which will result in saving storage space.

The Compress Usage job performs the following:

- Ensures that the ADJ1_CYCLE_STATE table contains an entry with the given cycle code, cycle instance, and cycle year
- Ensures that the value of the LOCKED_FOR_NEW and LOCKED_FOR_UPDATE columns of the ADJ1_CYCLE_STATE table is ‘Y’
- Calls a stored procedure that compresses the relevant partitions of the usage tables

Process Type

Batch job

Run Frequency

By request

Activation

Command Line:	RunJobs ADJ1COMPUSG BYREQ
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_CompressUsage_Job_Sh
Executable Name:	N/A

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Flow

The Compress Usage process executes the following steps:

1. Per cycle code, cycle instance, and cycle year, checks whether a record exists in the ADJ1_CYCLE_STATE table. If the record is missing, writes a warning message to the log.
2. Checks that the cycle is open for update (that the value of the LOCKED_FOR_NEW and LOCKED_FOR_UPDATE columns of the ADJ1_CYCLE_STATE table is ‘Y’).
3. Calls the ADJ1_PARTITION_PG.COMPRESS_CYCLE_PARTITIONS stored procedure.

The stored procedure gets the table name, cycle code, and cycle instance, and compresses the relevant partitions.

Parameters

The following parameters must be set for the process to run properly.

Parameter	Description	Default Value	Method of Changing the Parameter Manually
CYCLE_CODE	The cycle code	None	Operational job parameter. Can be updated through Amdocs Monitoring & Control.
CYCLE_INSTANCE	The cycle instance	None	Operational job parameter. Can be updated through Amdocs Monitoring & Control.
CYCLE_YEAR	The cycle year	None	Operational job parameter. Can be updated through Amdocs Monitoring & Control.
TABLES_LIST	The list of tables, separated by spaces	None	Operational job parameter. Can be updated through Amdocs Monitoring & Control.

The GN1_TASK_CONNECT table must contain an entry with the following values:

- TaskName=“ADJ1COMPUSG”
- RunMode=“F”
- SessionArg=“DEFAULT”

In addition, the GN1_CONNECT_PARAM table must contain a corresponding entry.

Database Connections

The process connects to the following table during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	ADJ1_CYCLE_STATE	Select

Important Remarks

The process uses the ADJCONFIG connection code (otherwise, the process fails).

53 ADJ1MOVEUSG – Move Usage

This chapter describes the Move Usage process.

Description

After a cycle instance has expired, and its data is accessed rarely (only via usage queries), the data in the Rated Events and Accumulators tables may be moved to a cheaper storage. This is generally done only when heavy activities (such as extracts) are no longer performed on the data.

The Move Usage job performs the following:

- Ensures that the ADJ1_CYCLE_STATE table contains an entry with the given cycle code, cycle instance, and cycle year
- Ensures that the value of the LOCKED_FOR_NEW and LOCKED_FOR_UPDATE columns of the ADJ1_CYCLE_STATE table is ‘Y’
- Calls a stored procedure that moves the relevant partitions of the usage tables from one tablespace to another

Process Type

Batch job

Run Frequency

By request

Activation

Command Line:	RunJobs ADJ1MOVEUSG BYREQ
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_MoveUsage_Job_Sh
Executable Name:	N/A

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Flow

The Move Usage process executes the following steps:

1. Per cycle code, cycle instance, and cycle year, checks whether a record exists in the ADJ1_CYCLE_STATE table. If the record is missing, writes a warning message to the log.
2. Checks that the cycle is open for updates (that the value of the LOCKED_FOR_NEW and LOCKED_FOR_UPDATE columns of the ADJ1_CYCLE_STATE table is ‘Y’).
3. Calls the ADJ1_PARTITION_PG.MOVE_CYCLE_PARTITIONS stored procedure.
The stored procedure moves the chosen partitions from the current tablespace to a new tablespace.

Parameters

The following parameters must be set for the process to run properly.

Parameter	Description	Default Value	Method of Changing the Parameter Manually
CYCLE_CODE	The cycle code	None	Operational job parameter. Can be updated through Amdocs Monitoring & Control.
CYCLE_INSTANCE	The cycle instance	None	Operational job parameter. Can be updated through Amdocs Monitoring & Control.
CYCLE_YEAR	The cycle year	None	Operational job parameter. Can be updated through Amdocs Monitoring & Control.
TABLES_LIST	The list of tables, separated by spaces	None	Operational job parameter. Can be updated through Amdocs Monitoring & Control.
TABLE_SPACE_NAME	The name of the destination tablespace	None	Operational job parameter. Can be updated through Amdocs Monitoring & Control.

The GN1_TASK_CONNECT table must contain an entry with the following values:

- TaskName=“ADJ1MOVEUSG”
- RunMode=“F”
- SessionArg=“DEFAULT”

In addition, the GN1_CONNECT_PARAM table must contain a corresponding entry.

Database Connections

The process connects to the following table during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	ADJ1_CYCLE_STATE	Select

Important Remarks

The process uses the ADJCONFIG connection code (otherwise, the process fails).

54 ADJ1TRNCUSG – Truncate Usage

This chapter describes the Truncate Usage process.

Description

After a cycle instance has expired, and its data is accessed rarely (only via usage queries), the data in the Rated Events and Accumulators tables may be truncated so that the tables can be re-used. This is generally done only when heavy activities (such as extracts) are no longer performed on the data.

The Truncate Usage job performs the following:

- Ensures that the ADJ1_CYCLE_STATE table contains an entry with the given cycle code, cycle instance, and cycle year
- Ensures that the value of the LOCKED_FOR_NEW and LOCKED_FOR_UPDATE columns of the ADJ1_CYCLE_STATE table is ‘Y’
- Calls a stored procedure that truncates the relevant partitions of the usage tables

Process Type

Batch job

Run Frequency

By request

Activation

Command Line:	RunJobs ADJ1TRNCUSG BYREQ
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_TrncUsage_Job_Sh
Executable Name:	N/A

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Flow

The Truncate Usage process executes the following steps:

1. Per cycle code, cycle instance, and cycle year, checks whether a record exists in the ADJ1_CYCLE_STATE table. If the record is missing, writes a warning message to the log.
2. Checks that the cycle is open for update (that the value of the LOCKED_FOR_NEW and LOCKED_FOR_UPDATE columns of the ADJ1_CYCLE_STATE table is ‘Y’).
3. Calls the ADJ1_PARTITION_PG.TRUNCATE_CYCLE_PARTITIONS stored procedure.

The stored procedure truncates the selected partitions.

Parameters

The following parameters must be set for the process to run properly.

Parameter	Description	Default Value	Method of Changing the Parameter Manually
CYCLE_CODE	The cycle code	None	Operational job parameter. Can be updated through Amdocs Monitoring & Control.
CYCLE_INSTANCE	The cycle instance	None	Operational job parameter. Can be updated through Amdocs Monitoring & Control.
CYCLE_YEAR	The cycle year	None	Operational job parameter. Can be updated through Amdocs Monitoring & Control.
TABLES_LIST	The list of tables, separated by spaces	None	Operational job parameter. Can be updated through Amdocs Monitoring & Control.

The GN1_TASK_CONNECT table must contain an entry with the following values:

- TaskName=“ADJ1TRNCUSG”
- RunMode=“F”
- SessionArg=“DEFAULT”

In addition, the GN1_CONNECT_PARAM table must contain a corresponding entry.

Database Connections

The process connects to the following table during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	ADJ1_CYCLE_STATE	Select

Important Remarks

The process uses the ADJCONFIG connection code (otherwise, the process fails).

55 ADJ1ROLLACCUM – Roll Accumulators

This chapter describes the Roll Accumulators job.

Description

The Roll Accumulators job participates in the creation of new accumulators for the current open cycle instance in the APE1_ACCUMULATORS table, after the previous cycle instance was closed.

The job sends a special Roll Accumulators event per subscriber or agreement to the Event Servers, which roll (copy) the subscriber's accumulators from the previous cycle instance to the new cycle instance. The Event Servers create the new accumulators and insert them into the APE1_ACCUMULATORS table.

Process Type

Batch job

Run Frequency

Daily, for every cycle instance that was closed on this day.

If the job fails for a specific cycle instance, it can be rerun on the same day or on one of the next days, to complete its work.

Participation in Generic Maps

This section contains information of the process participation in generic maps.

Map Name

End-of-Cycle (EOC)

For more information about this map, see *Amdocs Billing Operator Guide*.

Severity Level

Severity Level 2 – The map must hold until the job is fixed.

Activation



Caution: When the home database is up after a failure, the operator must not activate the Roll Accumulators process until the Copy Cycle Usage process has copied the accumulators from the alternate database to the home database.

Command line:	RunJobs ADJ1ROLLACCUM ENDDAY
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_ROLL_ACCUM_Job_Sh
Executable Name:	gepf1fwcApp

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

When the job is done, it shuts down on its own. If necessary, it can be shut down from the command line.

In the case of a manual graceful shutdown, the job shuts down only after it finishes processing all the table partitions that it has already begun processing.

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP address> ROLL_ACCUM \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes
Script Name	ADJ1_Send_Admin_Command_Sh

Example:

```
ADJ1_Send_Admin_Command_Sh 10.232.53.45 ROLL_ACCUM 899
CPF_GracefulShutdownCommand
```

Preceding Processes

The following job must run successfully before this process is activated:

- *ADJ1CRA* – Copy Rolling Accumulators

Dependent Processes

The following process can be run only after the successful completion of this process in the End-of-Cycle map:

- *ADJ1CYCMEMCLN* – Cycle Memory Clean-up

Affected Applications

The process affects the following applications:

- *Event Server* – When the Event Server receives a Roll Accumulators event from the Roll Accumulators job, it copies the accumulators of a subscriber or agreement to the next cycle instance.
- *Implementation Compiler* – The Implementation Compiler generates special code, which is used by the Event Server to roll over accumulators when it receives a Roll Accumulators event from the Roll Accumulators job. The Roll Accumulators job itself does not use the generated code.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*

ADJ1DB2E_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1DB2E_ROLL_ACCUM899_20081109_130927_1.log

- *Console log files:*

ADJ1_ROLL_ACCUM_Job_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log

Example:

ADJ1_ROLL_ACCUM_Job_inner_Sh_ROLL_ACCUM899_20081109_130927.log

- *Operational log files:*

ADJ1ROLLACCUM_<Job Record

Name>_\${ABP_MARKET}_<Date>_<Time>.log

Example:

ADJ1ROLLACCUM_END_OF_DAY_M3G_20081109_130927.log

- *Event log files:*

ADJ1DB2E_EVENT_LOG_<APPLICATION_ID>_%D_%T_%n.log

Example:

ADJ1DB2E_EVENT_LOG_ROLL_ACCUM899_20081109_130927_1.log

- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/.
- *Console log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/.
- *Operational log files* – \$ABP_LOG.
- *Event log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/ . It is also possible to load event log data into the APR1_DB2E_EVENT_LOG_LOAD database table by activating the ADJ1_DB2E_EventLogFileDBLoader_Sh script, When this script is run without any parameters, it informs the operator which parameters are required.
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter).

For information about the configuration parameters that define the location of log and trace files, see the “[Configuration Parameters](#)” section in the “[ADJ1EVENTSRV – Event Server](#)” chapter.

Contents

- *Log files* – Contains the process log messages of the following types (levels):
 - Trace
 - Info
 - Error
 - Fatal
- *Console log files* – Contains information that the process prints upon initialization before it opens the regular log file and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).
- *Operational log files* – Contains operational messages.
- *Event log files* – Information about the events sent to the Event Server and the answers received from the Event Server for each event. The type of information that must be printed to the event log is specified by the DB2E_EVENT_LOG_FORMAT configuration parameter. For more information, see the “[Configuration Parameters](#)” section in the “[ADJ1DB2E – DB2E](#)” chapter.
- *Light Tracer files* – See the “[Light Tracer Files](#)” section in the “[Introduction](#)” chapter.

Flow

The Roll Accumulators process executes the following steps:

1. Opens a cursor that selects all accumulators from the closed cycle instance’s Accumulators table that are candidates for the rolling activity (cross-cycle accumulators). An accumulator is recognized as a cross-cycle one when its Cross Cycle Indicator (the CROSS_CYCLE_IND column in the APE1_ACCUMULATORS table) is set to ‘Y’ = Yes or ‘S’= Special.
2. For each accumulator, checks whether it has already been created in the next cycle instance.
3. Generates an event for every subscriber or agreement with at least one cross-cycle accumulator that has not been rolled over to the next cycle instance.
4. Routes the event to the relevant Event Server according to the customer segment to which the subscriber or agreement belongs. The event is sent directly to the Event Processing Module of the Event Server; the Guiding to Customer Module is not required. For more information about the activities of the process if one of the Event Servers is down, see the “[Recovery Instructions](#)” section in this chapter.

5. Maintains a session for each event sent to the Event Processing Module.
6. Monitors the responses from the Event Server and verifies that the rolling operation has actually been performed.

If the number of events that return an error or do not return a response is high (configurable), the process shuts down. The process is not critical and may be shut down and rerun later.

The Event Server handles the generated event like any other event. This event triggers copying the cross-cycle accumulator to the current open cycle.

Environment Variables

The Roll Accumulators job uses the following environment variables.

Name	Description	Default Value
APR1_ROLL_ACCUM_JOB_SUCCESS_STATUSES	A space-separated list of application statuses for which the Roll Accumulators job exits with status 0, thereby enabling the operational map to continue.	N/A
ADJ_ROLL_ACCUM_MAX_RETRIES_NUM	The maximum number of reruns of the Roll Accumulators process within the Roll Accumulators job if the Event Server was disconnected or some responses from the Event Server arrived too late	5
ADJ_ROLL_ACCUM_SLEEP_SECONDS_BETWEEN_RETRIES	The sleep time between reruns of the Roll Accumulators process within the Roll Accumulators job	0

Parameters

The following job parameters must be set for the job to run properly.

Parameter	Description	Default Value
APPLICATION_ID	The process instance.	ROLL_ACCUM 899
CCF_FILE_NAME	An XML file that contains the relevant configuration.	Apr1_DB2E.ccf
CYCLE_CODE	The cycle code.	N/A
CYCLE_INSTANCE	The number of the closed cycle instance.	N/A
CYCLE_YEAR	The year of the closed cycle instance.	N/A

Parameter	Description	Default Value
PROCESS_NUM_OUT_OF_TOTAL	The number of the current process instance out of TOTAL_NUM_OF_PROCESSES. This parameter enables running several instances of this job. Different process instances handle different populations of subscribers.	N/A
TOTAL_NUM_OF_PROCESSES	The total number of instances of this job.	N/A

Configuration Parameters

The Roll Accumulators process inherits all the configuration parameters of the DB2E process (see the “Configuration Parameters” section in the “ADJ1DB2E – DB2E” chapter).

In addition, it has the following configuration parameters.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ROLL_ACCUM	config	BULK_ENTRIES_PER_DB_READER	15000	The number of bulk entries that are pre-allocated and used by each DbReader thread.
ROLL_ACCUM	config	EXPIRE_ROLL_ACCUM_EVENT_TIME_SEC	30	The expiration time after which the Roll Accumulators process attempts to resend the data.
ROLL_ACCUM	config	MAX_RESENGS_PER_BULK	5	The maximum number of resends for a bulk, not including the first transmission. If a bulk has reached this number of resends and must be resent, the unanswered and rejected events in the bulk are expired. That is, the answered events in the bulk are considered successful, and the unanswered and rejected events are considered erred.
ROLL_ACCUM	config	NUM_OF_ERRORS_PER_PROCESS_THRESHOLD	0 (no limit)	The maximum number of erred (rejected and expired) events, after which the process shuts down.  <i>Note: A number of events may be sent even after the threshold is exceeded because events are sent and answers are received asynchronously and by different threads.</i>
ROLL_ACCUM	config	UH_MODE_AT_STARTUP	ROLL_ACCUM This value must not be changed.	The process mode at start-up.

Also, the process uses Light Tracer configuration parameters, which are defined in the ADJ1_CONF_SECTION_PARAM table. For more information, see [Light Tracer Parameters](#) in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

For each process instance, an entry with the name of the process instance must be created in the APR1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
ROLL_ACCUM899	ROLL_ACCUM

Any parameter to be overridden at the process instance level must be added to the APR1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
ROLL_ACCUM899	config	NUM_OF_ERRORS_PER_PROCESS_THRESHOLD	1000

For more information on parameters and the properties mechanism, see the “Configuration Parameter Mechanism” appendix.

Configuration Files

The Roll Accumulators process requires the same configuration file as the DB2E process (Apr1_DB2E.ccf). The Roll Accumulators process and the DB2E process have very similar functionality: both read data from the database, create events from this data, send the events to the relevant Event Servers, wait for answers from the Event Servers, and update their progress status in a control table. Therefore, much of the source code of these two processes is common to both of them, and they use the same .ccf file. The file is defined via an internal configuration tool.

Database Connections

The process connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Usage	APE1_ACCUMULATORS	Select
Application	APR1_ROLL_ACCUM_CTRL	Insert, Update, Select

Admin Commands

Although the Roll Accumulators process is a job, it can receive and handle admin commands while it is still running. If the job has already finished running, the command is ignored.

The Roll Accumulators job supports the following admin commands. For information on how to execute the commands, see the “Handling Admin Commands” section in the “High Availability” chapter.

Command	Description	Parameters
SUSPEND_GROUPS_COMMAND	Suspends event transmission for specific groups of resources or customers	--segmentsType "<Resources, Customers, or All>" --cycleCode "<Cycle code>" --fromGroup "<From group number>" --toGroup "<To group number>" --cycleCode "<Cycle code>" --fromGroup "<From group number>" --toGroup "<To group number>" ... <i>Examples:</i> <ul style="list-style-type: none">• ADJ1_Send_Admin_Command_Sh 10.232.53.45 ROLL_ACCUM 899 SUSPEND_GROUPS_COMMAND --segmentsType "Resources" --cycleCode "0" --fromGroup "30" --toGroup "36" --cycleCode "0" --fromGroup "51" --toGroup "51"• ADJ1_Send_Admin_Command_Sh 10.232.53.45 ROLL_ACCUM 899 SUSPEND_GROUPS_COMMAND --segmentsType "All"• SUSPEND_GROUPS_COMMAND --segmentsType "Customers" --cycleCode "1" --fromGroup "18" --toGroup "30"
RESUME_GROUPS_COMMAND	Resumes event transmission for specific groups of resources or customers	--segmentsType "<Resources, Customers, or All>" --cycleCode "<Cycle code>" --fromGroup "<From group number>" --toGroup "<To group number>" --cycleCode "<Cycle code>" --fromGroup "<From group number>" --toGroup "<To group number>" ... <i>Examples:</i> <ul style="list-style-type: none">• ADJ1_Send_Admin_Command_Sh 10.232.53.45 ROLL_ACCUM 899 RESUME_GROUPS_COMMAND --segmentsType "Resources" --cycleCode "0" --fromGroup "30" --toGroup "36" --cycleCode "0" --fromGroup "51" --toGroup "51"• ADJ1_Send_Admin_Command_Sh 10.232.53.45 ROLL_ACCUM 899 RESUME_GROUPS_COMMAND --segmentsType "All"• ADJ1_Send_Admin_Command_Sh 10.232.53.45 ROLL_ACCUM 899 RESUME_GROUPS_COMMAND --segmentsType "Customers" --cycleCode "1" --fromGroup "18" --toGroup "30"

Command	Description	Parameters
SUSPEND_TARGET_SERVER_COMMAND	Suspends production and transmission of events to a specific Event Server	<pre>--segmentsType "<Resources, Customers, or All>" -- targetServerProcessGroup "<Target Event Server process group>"</pre>  <p>Note: If the --segmentsType parameter is Resources, only resource groups of this Event Server are suspended. If it is Customers, only customer groups of this Event Server are suspended. If it is "All", all the groups of this Event Server are suspended.</p> <p>Examples:</p> <ul style="list-style-type: none"> • ADJ1_Send_Admin_Command_Sh 10.232.53.45 ROLL_ACCUM 899 SUSPEND_TARGET_SERVER_COMMAND --segmentsType "All" --targetServerProcessGroup "3000" • ADJ1_Send_Admin_Command_Sh 10.232.53.45 ROLL_ACCUM 899 SUSPEND_TARGET_SERVER_COMMAND --segmentsType "Customers" --targetServerProcessGroup "3100"
RESUME_TARGET_SERVER_COMMAND	Resumes production and transmission of events to a specific Event Server	<pre>--segmentsType "<Resources, Customers, or All>" -- targetServerProcessGroup "<Target Event Server process group>"</pre>  <p>Note: If the --segmentsType parameter is Resources, only resource groups of this Event Server are resumed. If it is Customers, only customer groups of this Event Server are resumed. If it is "All", all the groups of this Event Server are resumed.</p> <p>Examples:</p> <ul style="list-style-type: none"> • ADJ1_Send_Admin_Command_Sh 10.232.53.45 ROLL_ACCUM 899 RESUME_TARGET_SERVER_COMMAND --segmentsType "All" --targetServerProcessGroup "3000" • ADJ1_Send_Admin_Command_Sh 10.232.53.45 ROLL_ACCUM 899 RESUME_TARGET_SERVER_COMMAND --segmentsType "Customers" --targetServerProcessGroup "3100"

Command	Description	Parameters
SET_LIGHT_TRACER_COMMAND	Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters	<ul style="list-style-type: none"> ■ Activation parameters: <ul style="list-style-type: none"> • <code>--turnTracer "on/off"</code> – Activates or stops the Light Tracer • <code>--turnAssert "on/off"</code> – Activates or stops Assertions • <code>--turnLogOnTrace "on/off"</code> – Activates or stops writing log messages in Light Tracer files • <code>--turnFlowTracer/turnAll "on/off"</code> – Activates or stops tracing a single flow or event that is transferred between machines • <code>--turnAll "on/off"</code> – Activates or stops all of the above. <p>The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter).</p> <ul style="list-style-type: none"> ■ <code>--moduleID "ALL"</code> ■ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • <code>--threadInfo "Y/N"</code> – Specifies whether trace or assertion messages include the thread information. • <code>--timestampInd "Y/N"</code> – Specifies whether trace or assertion messages include the time stamp. • <code>--contextInd "Y/N"</code> – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • <code>--noOfMessages "<number>"</code> – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down. • <code>--period "5/10/20/60/120/480/1440"</code> – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated. <p> Note: If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached.</p> <ul style="list-style-type: none"> • <code>--assertAbortInd "Y/N"</code> – Specifies whether the application is to be aborted if the assertion condition fails.

Command	Description	Parameters
		<ul style="list-style-type: none"> ■ <code>--commandType "changeBufferSize"</code> – Resizes the Light Tracer buffer at run time. • <code>--bufferSize "<number>"</code> – The new size for the Light Tracer buffer. <p><i>Examples:</i></p> <ul style="list-style-type: none"> • <code>ADJ1_Send_Admin_Command_Sh hpbl820 ROLL_ACCUM 899 SET_LIGHT_TRACER_COMMAND --turnTracer "on" --moduleId "ALL" --threadInfo "y"</code> • <code>ADJ1_Send_Admin_Command_Sh hpbl820 ROLL_ACCUM 899 SET_LIGHT_TRACER_COMMAND --turnTracer "off" --moduleId "ALL"</code>
SET_TRACE_SQL_COMMAND	<p>Activates or stops SQL tracing at runtime. The command name with its parameters must be enclosed in quotation marks.</p> <p><i>Example:</i> <code>ADJ1_Send_Admin_Command_Sh illin1027 <process instance> "SET_TRACE_SQL_COMMAND -traceSqlOn \"yes\" --duration \"1\""</code></p>	<code>--traceSqlOn "<yes or no>" --duration "<minutes>"</code> If the <code>IS_DURATION_REQUIRED</code> environment variable is set to 'Y', the <code>--duration</code> parameter is mandatory. If the duration specified in the command exceeds the allowed maximum value, SQL tracing is activated for the maximum allowed period of time only (60 minutes).  <p><i>Note: The <code>SHUT_DOWN_CHECK_FREQ</code> parameter of the Light Tracer in the <code>ADJ1_CONF_SECTION_PARAM</code> table determines the frequency (defined in the number of messages) at which the SQL Trace Manager checks whether the period specified in the <code>--duration</code> parameter has passed. Therefore, SQL tracing is stopped after the specified duration or after this number of messages has been reached, whichever comes later.</i></p>
SWITCH_LIGHT_TRACER_ON ALL	Activates the Light Tracer without the ability to specify the mode or send parameters	N/A
SWITCH_LIGHT_TRACER_OFF ALL	Stops the Light Tracer	N/A
CPF_GracefulShutdownCommand	Shuts down Roll Accumulators gracefully	N/A

Troubleshooting

See the “[Troubleshooting](#)” section of the “[ADJ1DB2E – DB2E](#)” chapter.

Recovery Instructions

When an Event Server to which the Roll Accumulators process must connect is down, the process continues to send events to the other Event Servers. However, in this case, it writes informative messages to the log. To complete the rolling of the accumulators that have not been rolled yet, the Roll Accumulators process (within the Roll Accumulators job) is rerun several times just in case the problematic Event Servers are up again. If the last rerun of the process within the job still finishes with an error status, the Roll Accumulators job finishes with an error status:

- If the Roll Accumulators job finishes with a WARNING status, the operational map must restart the job (this can be done by a script), unless all the accumulators have already been rolled successfully.
- If the job finishes with an ERROR status, the following activities must be performed:
 - a. Stopping the End-of-Cycle operational map
 - b. Investigating and fixing the cause of the error
 - c. Restarting the Roll Accumulators job

Perform the following steps to rerun the process after a failure:

1. Fix the problems that were reported in the log files.
2. Restart the Roll Accumulators process.

The Roll Accumulators process does *not* resend to the Event Server the events of subscribers or agreements for which all the accumulators have already been rolled over to the next cycle instance. The Roll Accumulators process sends to the Event Servers events of subscribers or agreements for which there are still accumulators that have not been rolled over.

Important Remarks

This section includes additional information on the Roll Accumulators process.

Interface with Availability Manager

The Roll Accumulators process interfaces with the Availability Manager in the same manner as DB2E. For more information, see the “[Interface with Availability Manager](#)” section in the “[ADJ1DB2E – DB2E](#)” chapter.

Defining Multiple Instances of Roll Accumulators Job

The APE1_ACCUMULATORS table partitions can be processed in parallel by several instances of the Roll Accumulator job, which can run on different machines or on the same machine.

The TOTAL_NUM_OF_PROCESSES parameter of the Roll Accumulators job defines the total number of instances among which the partitions of the APE1_ACCUMULATORS table are divided.

The PROCESS_NUM_OUT_OF_TOTAL parameter of the job defines the number of the process instance out of the total number of process instances (TOTAL_NUM_OF_PROCESSES).

Each partition in the APE1_ACCUMULATORS table that contains accumulators of the specific cycle code and cycle instance for which the job is run, is assigned to a process instance for handling. The total number of partitions that are handled by a process instance is close to the number of partitions that are handled by another process instance.

There is no need to define which customer groups are to be handled by each instance of the Roll Accumulators job.

56 ADJ1CRA – Copy Rolling Accumulators

This chapter describes the Copy Rolling Accumulators process.

Description

Whenever a customer's cycle changes (upon the customer's request or due to operational changes), the APE1_CUST_CYCLE_HISTORY table is updated with the following information:

- Customer ID
- Current cycle
- New cycle

The Copy Rolling Accumulators process copies the cross-cycle accumulators of customers or subscribers from the old cycle to the new cycle. The process generates SQL statements into SQL files and activates them. Each file holds the data that is relevant for a specific database instance.

To distinguish copied accumulators from regular records, the DIMENSION_ID field of the APE1_ACCUMULATORS table for copied accumulators is set to 99999.

The Copy Rolling Accumulators job operates in three modes:

- *EOC* – Runs after the rerate and copies the cross-cycle accumulators of the cycle's population for which the cycle has changed. It also marks the relevant customers or subscribers for rerate (in the new cycle).
- *Undo* – Copies the accumulators of all customers whose cycle has changed and whose events have been rerated since last process run.
- *Recovery* – In the case of an error with SQL file generation, reruns the Copy Rolling Accumulators SQL files that have not been processed.

Process Type

Batch job

Run Frequency

By request or as part of a map

Participation in Generic Maps

This section contains information on the generic maps in which the process participates.

Map Name

- End-of-Cycle (EOC)
- Undo

For more information about these maps, see *Amdocs Billing Operator Guide*.

Severity Level

Severity Level 1 – The map can continue to the next job.

Activation

Command Line:	<ul style="list-style-type: none">■ RunJobs ADJ1CRA EOC■ RunJobs ADJ1CRA Undo■ RunJobs ADJ1CRA Recover
Amdocs Monitoring & Control:	No
Script Name:	ADJ1_CopyRollingAccumulators_Job_Sh
Executable Name:	N/A

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Preceding Processes

The following job must run successfully before this process is activated:

- *ADJIRRPEOC* – Rerate Prepare for End of Cycle

Out-of-the-box, the process runs as part of the End-of-Cycle and Undo maps after the rerate process is done. However, the operational maps can be customized to run the process with different dependencies.

Dependent Processes

The following process can be run only after the successful completion of this process in the End-of-Cycle map:

- *ADJ1ROLLACCUM* – Roll Accumulators

Affected Applications

The process affects the following applications:

- *Event Server* – Requires information on how to handle the copied accumulators
- *Accumulator Extracts* – Do not extract records from the APE1_ACCUMULATORS table if DIMENSION_ID = 99999
- *Usage Query Server* – Does not extract records from the APE1_ACCUMULATORS table if DIMENSION_ID = 99999
- *Roll Accumulators* – Requires information on how to handle the copied accumulators and creates technical events in the new cycle

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Log files are created and handled in the default way.

Name

- *Log files:*

ADJ1CRA_<Date>_<Time>.log

Example:

ADJ1CRA_20090624_115035.log

- *Console log files:*

ADJ1_CopyRollingAccumulators_Job_inner_Sh_<Date>_<Time>.log

Example:

ADJ1_CopyRollingAccumulators_Job_inner_Sh_20081109_130927.log

- *Operational log files:*

ADJ1CRA_<Job Record Name>_\${ABP_MARKET}_<Date>_<Time>.log

Example:

ADJ1CRA_EOC_M3G_20090624_114834.log

Location

Log files are located in the following directories:

- *Log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process error information
- *Console log files* – Information that the process prints upon initialization before it opens the regular log file and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands)
- *Operational log files* – The output of operational scripts

Output Files

The process generates the following files in the \$ABP_ADJ_ROOT/work/CopyRollingAcc directory:

- *Data and control file:*

ACC_RECORDS_<Cycle_Code>_<Cycle_Instance>_<Cycle_Year>_<Date>_<Time>_<Unique_ID>.dat

Example:

ACC_RECORDS_1_12_2008_20090818_171314_29527.dat

- *SQL files:*

CRA_<DB_Connection_Code>_<Cycle_Code>_<Cycle_Instance>_<Cycle_Year>_<Date>_<Time>_<Unique_ID>.sql

Example:

CRA_ADJUSG1_1_12_2008_20090818_180736_13384.sql

When the process ends successfully, the suffix ‘done’ is added to the file name.

Examples:

ACC_RECORDS_1_12_2008_20090818_171314_29527.dat.done

CRA_ADJUSG1_1_12_2008_20090818_180736_13384.sql.done

Flow

The Copy Rolling Accumulators process executes the following steps:

1. Sets all database connections.
2. Gets the connection string to the source cycle.
3. If RUN_MODE = ‘Recovery’:
 - a. Runs all unprocessed SQL files (all files with the following pattern:
CRA_*_<Cycle_Code>_<Cycle_Instance>_<Cycle_Year>_*.sql)
 - b. Updates the control table (according to the JOB_RUN_ID on the SQL file)
 - c. Exits
4. Inserts a control record to the ADJ1_CPY_ROLL_ACCUM_CTRL table.
5. Calculates the Change Cycle Effective Date variable as *the close date of the source cycle + 1 second*.
6. Truncates the relevant partition of the ADJ1_CRA_POPULATION_TMP table based on the cycle code and instance.

7. Populates the ADJ1_CRA_POPULATION_TMP with the following data:
 - In the End-of-Cycle mode, joins the data from the following tables:
 - APE1_CUST_CYCLE_HISTORY
 - ADJ1_CYCLE_STATE
 - ADJ1_CYCLES
 - In the *Undo* mode, joins the data from the following tables:
 - APE1_CUST_CYCLE_HISTORY
 - ADJ1_CYCLE_STATE
 - ADJ1_CYCLES
 - APE1_RERATE_POPULATION, where:
 - ◆ PROCESS_STS = ‘D’
 - ◆ *SYS_UPDATE_DATE* is the last time that the process ran successfully for the given cycle code, instance, and year (taken from the ADJ1_CPY_ROLL_ACCUM_CTRL table).
8. Builds the select statement. Because the structure of the APE1_ACCUMULATORS table is flexible, the select statement is built using the ALL_TAB_COLUMNS table.
9. Creates a buffer of records from the APE1_ACCUMULATORS table by joining the APE1_ACCUMULATORS table with the ADJ1_CRA_POPULATION_TMP table.
10. Generates the SQL files, creating one file per database instance.
11. Runs the generated SQL files and appends their file names with ‘.done’ after successful execution.
12. Updates the ADJ1_CPY_ROLL_ACCUM_CTRL table.
13. Deletes all old records from the ADJ1_CRA_POPULATION_TMP table.
14. Exits.

Environment Variables

The Copy Rolling Accumulators process uses the following environment variable initialized in the op_adj_env_sh script.

Name	Description	Default Value
ADJ1_COPY_PREPAID_ACCUMULATORS	Defines whether records from the APE1_ACCUMULATORS table whose CROSS_CYCLE_IND = ‘S’ are to be copied to the new cycle	N

Parameters

The following parameters must be set for the process to run properly.

Parameter	Description	Default Value	Method of Changing the Parameter Manually
CYCLE_CODE	Cycle code	None	Operational job parameter. Can be updated through Amdocs Monitoring & Control.
CYCLE_INSTANCE	Cycle instance	None	Operational job parameter. Can be updated through Amdocs Monitoring & Control.
CYCLE_YEAR	Cycle year	None	Operational job parameter. Can be updated through Amdocs Monitoring & Control.
MAP_MODE	Map mode: ■ EOC ■ Undo ■ Recovery	None	Operational job parameter. Can be updated through Amdocs Monitoring & Control.

Configuration Parameters

After a cycle change, if the first cycle instance of the new cycle is short, the Copy Rolling Accumulators process copies the accumulators to the next cycle instance, according to the **SHORT_CYCLE_PER_CYCLE_FREQUENCY_PERIOD** parameter defined in the **APE1_CONF_SECTION_PARAM** table.

Database Connections

The process connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	ADJ1_CYCLE_STATE	Select
Application	APE1_ACCUMULATORS	Delete/Select/Insert
Oracle	ALL_TAB_COLUMNS	Select
Reference	ADJ1_CYCLES	Select
Usage	ADJ1_CPY_ROLL_ACCUM_CTRL	Insert, Update, Select
Usage	ADJ1_CRA_POPULATION_TMP	Select, Insert, Truncate
Usage	APE1_CUST_CYCLE_HISTORY	Select
Usage	APE1_RERATE_POPULATION	Select
Usage	APE1_SUBSCRIBER_RERATE	Insert

Important Remarks

- If the target cycle instance is closed when the Copy Rolling Accumulators process runs, accumulators from the old cycle are not rolled to the new cycle, and the subscriber's accumulators become inaccurate.
- The ADJ1_CRA_POPULATION_TMP table must be partitioned by CYCLE_CODE and CYCLE_INSTANCE.

57 ADJ1CYCMEMLN – Cycle Memory Clean-up

This chapter describes the Cycle Memory Clean-up process.

Description

The Cycle Memory Clean-up process is responsible for deleting the accumulators belonging to a closed cycle instance from the shared memory during end-of-cycle activities.

Process Type

Batch job

Run Frequency

Every cycle

Participation in Generic Maps

This section contains information on the generic maps in which the process participates.

Map Name

End-of-Cycle (EOC) map

For more information about this map, see *Amdocs Billing Operator Guide*.

Severity Level

Severity Level 1 – The map can continue to the next job without fixing this job.

Activation

Command Line:	RunJobs ADJ1CYCMEMCLN <Job Rec>
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_CYCLEMEMCLN_Job_Sh
Executable Name:	gn1avm_admin

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

When the job is done, it shuts down on its own.

Preceding Processes

The following jobs must run successfully before this process is activated, and the following daemons must be active for this process to run:

- *ADJ1ROLLACCUM* – Roll Accumulators
- *AVM* – Availability Manager

Affected Applications

The Cycle Memory Clean-up process affects the following application:

- *Event Server* – Cycle Memory Clean-up sends the REMOVE MEMORY command through Amdocs Monitoring & Control to the respective Event Server

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

ADJ1CYCMEMCLN_ADJ1CYCMEMCLN_%D_%T_%n.log

Example:

ADJ1CYCMEMCLN_ADJ1CYCMEMCLN_20081109_130927_1.log

Location

Log files are located in \$ABP_LOG.

Contents

The log files contain the process error information.

Flow

In the End-of-Cycle map, accumulators of the closed cycle instance are deleted from the shared memory as follows:

1. After the Cycle Maintenance (ADJ1CYCLEMAINT) job has finished, the current cycle instance is locked.
2. The Roll Accumulators (ADJ1ROLLACCUM) job copies the rolling accumulators from the current cycle instance to the next cycle instance.
3. The Cycle Memory Clean-up job sends the REMOVE MEMORY command through the Availability Manager to the Event Server that handled the current cycle code, cycle instance, and cycle year.
4. Accumulators that belong to the closed cycle instance are removed from the memory of the Event Server.

Environment Variables

The Clean Shared Memory (Accumulators) for Closed Cycles uses the following environment variables.

Variable Name	Description	Valid Values/ Default Value
OP_ORA_USER	The user name for the Operational database	N/A
OP_ORA_PASS	The password to the Operational database	N/A
OP_ORA_INST	The instance of the Operational database	N/A

Parameters

The following parameters must be set for the Cycle Memory Clean-up job to run properly.

Parameter	Description	Default Value
CYCLE_CODE	The cycle code of the cycle whose accumulators must be deleted from memory	1
CYCLE_INST	The cycle instance of the cycle whose accumulators must be deleted from memory	1
CYCLE_YEAR	The cycle year of the cycle whose accumulators must be deleted from memory	2009

Configuration Parameters

The Availability Manager instance must be configured in the GN1_SYS_PROC_INSTANCE_CFG table for PROCESS_TYPE_ID=77.

For each process instance, an entry with the name of the process instance must be created in the APR1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_GROUP
CYC_MEMCLN170	CYC_MEMCLN

Database Connections

The Cycle Memory Clean-up job connects to the Operational database using the connection string defined by the following parameter:

ART_AMC_START_DB_CONNECT_STRING=\$OP_ORA_USER/\$OP_ORA_PASS
@\$OP_ORA_INST

In addition, it connects to the following tables to register the progress of the REMOVE MEMORY command.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	AVM1_COMMAND_PROGRESS	Update, Insert
Application	AVM1_ACTIVITIES	Update, Insert

58 ADJ1CPCYCUSG – Copy Cycle Usage

This chapter describes the Copy Cycle Usage job.

Description

The Copy Cycle Usage job is responsible for copying data from the alternate database to the home database for all the cycle codes that originally resided on the home database, after failure and recovery of the home database.

A database contains the usage data of one or more cycle codes. If the entire Oracle RAC (the primary and the secondary database instance) fails, the Event Server redirects the persistent data, such as events and accumulators, to an alternate database instance (on a different host and connected to a different storage) so that it can continue working and processing. This implies that after a failure of the home database, some of the usage data for these cycles resides in the alternate database.

After the home database is restored, and the database administrator approves using the home database, the following occurs:

1. The Event Servers and Update Handlers receive a command to reconnect to the home database and start writing all usage data to the home database.
2. The Copy Cycle Usage job runs to copy the data that resides in the alternate database back to the home database. If the Copy Cycle Usage job is activated before the completion of the previous step, it shuts down immediately.

In addition, the Copy Cycle Usage job activates the Update Handler in the Mini-Full mode, which updates the subscriber data in the home database. For more information, see the “ADJ1UHMINFULL – Update Handler in Mini-Full Mode” chapter.

3. The Copy Cycle Usage job finishes only after it has copied the relevant data and verified that DB2E and Notification processes have completed the processing of the relevant records in the alternate database.
4. When the Copy Cycle Usage process is finished, the data that has been successfully copied to the home database can be deleted from the alternate database. For more information, see the “ADJ1CCUCU – Copy Cycle Usage Clean-up” chapter.

The Copy Cycle Usage job copies the data from the following tables:

- From the following tables, the data of the last closed cycle instance and two open cycle instances is copied from the alternate database to the home database:
 - *APE1_ACCUMULATORS* – Records are copied from the alternate database to the home database *only* if the record in the alternate database is more up-to-date than the record in the home database, or if the record does not exist in the home database at all.
 - *APE1_RATED_EVENT* – All the records of the cycle code are copied from the alternate database to the home database.
 - *APE1_REJECTED_EVENT* – All the records of the cycle code are copied from the alternate database to the home database.
 - *APE1_SUBSCRIBER_RERATE* – All the records of the cycle code are copied from the alternate database to the home database.

- *APE1_RR_BLACK_LIST* – All the records of the cycle code are copied from the alternate database to the home database.
- *APE1_EVENT_DUP_KEYS* – All the records of the cycle code are copied from the alternate database to the home database.
- From the following tables, all the records of the cycle code are copied from the alternate database to the home database:
 - APR1_DISPATCHING_RECORDS
 - APR1_DISPATCHER_CTRL
 - APE1_TECH_EVENTS
- From the APE1_SUBSCRIBER_DATA table in the alternate database, only the subscribers are copied to the AUH1_REQUEST table in the home database. After all the data of all cycles has been copied, the Update Handler is activated in Mini-Full mode.

Process Type

Batch job

Run Frequency

The job is run upon request, after a database failure and recovery, for all the cycle codes of the recovered home database.

If the process crashes or has been stopped by request, it can be rerun to complete its work after the problem has been fixed. At run time, the job updates a control table (APR1_COPY_CYCLE_USAGE_CTRL) with information regarding the data that it has copied successfully from the alternate database to the home database. Therefore, if the job is rerun, it continues its work from where it left off in its previous run, if the previous run was aborted. For more information, see the “Recovery Instructions” section in this chapter.

Activation

Command line:	export UH_MINI_FULL_INSTANCE=<instance ID of the Update Handler in Mini-Full mode> RunJobs ADJ1CPCYCUSG BYREQ
Amdocs Monitoring & Control:	Yes
Script Name:	ADJ1_COPY_CYCLE_USAGE_Job_Sh
Executable Name:	gcpf1fwcApp

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Shutdown

When the job is done, it shuts down on its own. If necessary, it can be shut down from the command line.

In the case of a manual graceful shutdown, the job shuts down only after it finishes processing all the table partitions that it has already begun processing.

Command Line:	ADJ1_Send_Admin_Command_Sh <Target machine name or IP address> CPCYCUSG \${PROCESS_INSTANCE_NUMBER} CPF_GracefulShutdownCommand
Amdocs Monitoring & Control:	Yes
Script Name	ADJ1_Send_Admin_Command_Sh

Dependent Processes

When the home database is back up after a failure, the following processes can be run only after the successful completion of this process for the cycle code:

- *ADJ1CCUCU* – Copy Cycle Usage Clean-up
- *ADJIROLLACCUM* – Roll Accumulators (can run only after the accumulators have been copied)
- Usage Extracts
- Usage Query mechanism (although the accumulator query is enabled after the accumulators have been copied)
- *ADJ1RER* – Rejected Event Recycler (can run only after the Rejected Event table has been copied)
- *ADJ1RRP* – Rerate Prepare

Affected Applications

The process affects the following application:

- *Event Server* – After the home database has been recovered, the following functionality of the Event Server is suspended until the various stages of the Copy Cycle Usage process are completed for the cycle code:
 - *Creating accumulators* – Suspended until the accumulators have been copied, except for when a new subscriber is created or a customer is loaded in eager mode.
 - *Rolling accumulators* – Suspended until the accumulators have been copied. The operator must not activate the Roll Accumulators process.
 - *Recovering accumulators* – Suspended until the accumulators have been copied.
 - *Recycling rejected events* – Suspended until the Copy Cycle Usage process is complete. The operator must not activate the Rejected Event Recycler process.

- *Checking for duplicates* – Suspended until the Rated Event table has been copied.
- *Rerating* – Suspended until the Copy Cycle Usage process is complete. The operator must not activate the Rerate Prepare process.
- *Update Handler in Mini-Full mode* – The Copy Cycle Usage job populates the AUH1_REQUESTS table before the Update Handler is activated in Mini-Full mode. To ensure that the new request does not overlap with existing requests, the Copy Cycle Usage job uses ‘-1’ as the request ID.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Log files:*
ADJ1CPCYCUSG_<APPLICATION_ID>_%D_%T_%n.log
Example:
ADJ1CPCYCUSG_CPCYCUSG145_20081109_130927_1.log
- *Console log files:*
ADJ1_COPY_CYCLE_USAGE_Job_inner_Sh_<APPLICATION_ID>_<Date>_<Time>.log
Example:
ADJ1_COPY_CYCLE_USAGE_Job_inner_Sh_CPCYCUSG145_20081109_130927.log
- *Operational log files:*
ADJ1CPCYCUSG_<Job Record Name>_\${ABP_MARKET}_<Date>_<Time>.log
Example:
ADJ1CPCYCUSG_BY_REQ_M3G_20081109_130927.log
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Location

Log files are located in the following directories:

- *Log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/

- *Operational log files* – \$ABP_LOG
- *Light Tracer files* – As defined by the LIGHT_TRACER.FILE_PATH and LIGHT_TRACER.FLOW_TRACE.FILE_PATH configuration parameters (see Light Tracer Parameters in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter)

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

- *Log files* – Contains the process log messages of the following types (levels):
 - Trace
 - Info
 - Error
 - Fatal
- *Console log files* – Contains information that the process prints upon initialization before it opens the regular log file and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).
- *Operational log files* – Contains operational messages.
- *Light Tracer files* – See the “Light Tracer Files” section in the “Introduction” chapter.

Flow

This section describes the flow of the Copy Cycle Usage process.

Process Start-up

At start-up, the process performs the following:

1. Checks the Resource Map of the Availability Manager to make sure that all the cycles of the same connection code are in stage 2 (confirmation messages from the Event Servers and the Update Handlers have arrived).
 - If so, continues
 - If not, shuts down immediately
2. Gets parameter values from the GN1_CONF_SECTION_PARAM, ADJ1_CONF_SECTION_PARAM, and APR1_CONF_SECTION_PARAM tables.
3. Creates connections to the alternate database, home database, and applicative database for each worker thread of the process.

Main Loop

The Copy Cycle Usage job has a producer thread and one or more worker threads. This section describes the main loop of each thread type.

Producer Thread

The producer thread performs the following flow:

1. Goes through the map received from the Availability Manager and searches for all the cycles that have the same connection code (on the home and alternate databases).
2. For each cycle code, retrieves the last closed cycle instance and two open cycle instances from the ADJ1_CYCLE_STATE table.
3. Loops over all the tables to be copied:
 - a. Fetches all the partition names for the table from the database
 - b. For each partition of each table, calculates the number of records to be copied
 - c. Prepares a message with the table name and partition name to be copied
 - d. Sends the message to the worker queue

Worker Threads

Each worker thread performs the following flow:

1. Takes a message from the queue and starts processing the corresponding table and partition. Each worker thread works on all the usage tables, but on different table partitions.
2. For each handled partition of a table, opens a cursor in the alternate database for fetching the data buffer by buffer. For each fetched buffer, the data is inserted as is or after some manipulation into the home database.
3. For each completed partition, updates the completion status in the APR1_COPY_CYCLE_USAGE_CTRL table.
4. For each completed message, checks whether it can notify the Availability Manager about a finished stage.

Copy Stages

Data is copied from the alternate database to the home database in four stages. In each stage, a different set of tables is copied:

1. Stage 1:
 - APE1_ACCUMULATORS
 - APE1_SUBSCR_DATA (by placing a request in the AUH1_REQUESTS table and activating the Update Handler in Mini-Full mode)

2. Stage 2 (the APR1_DISPATCHER_CTRL table can be copied only after the entire APR1_DISPATCHING_RECORDS table has been copied):
 - APR1_DISPATCHING_RECORDS
 - APR1_DISPATCHER_CTRL
3. Stage 3:
 - APE1_RATED_EVENT
 - APE1_EVENT_DUP_KEYS
 - APE1_REJECTED_EVENT
 - APE1_SUBSCRIBER_RERATE
 - APE1_RR_BLACK_LIST
 - APE1_TECH_EVENTS
4. Stage 4:

A check that Notification and DB2E have finished working on the alternate database

After the data of all the tables that related to a stage is copied, a message is sent to the Availability Manager.

Process Shutdown

The process disconnects from the databases, clears the memory, and shuts down.

Environment Variables

The Copy Cycle Usage process uses the following environment variables that are initialized by the op_adj_env_sh script.

Name	Description	Default Value
APR_COPY_CYCLE_USAGE_NUM_OF_THREADS	The number of Copy Cycle Usage worker threads	1
UH_MINI_FULL_INSTANCE	<ul style="list-style-type: none">■ <i>Numeric</i> – The instance ID of the Update Handler in Mini-Full mode to be invoked■ <i>Otherwise</i> – The process name	N/A
CPCYUSG_UH_MINI_FULL_PROFILE	The profile of the Update Handler in Mini-Full mode to be invoked	UHImplMiniFullRating

For more information on the script, see the “Amdocs Batch Job Manager Application Script” section in the “Introduction” chapter.

Parameters

The following job parameters must be set for the job to run properly.

Parameter	Description	Default Value
APPLICATION_ID	The process instance	CPCYCUSG145
CCF_FILE_NAME	An XML file that contains the relevant configuration	APR1_COPY_CYCLE_USAGE.ccf
CONNECTION_CODE	The connection code to the home database	N/A
ALTER_CONNECTION_CODE	The connection code to the alternate database	N/A

Configuration Parameters

The Copy Cycle Usage process has the following configuration parameters.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
CPCYCUSG	config	ACCUMS_NUM_OF_FETCH_ROWS	1000	The maximum number of accumulator records that can be fetched into one buffer
CPCYCUSG	config	DISP_CTRL_FETCH_BUFFER_SIZE	1000	The maximum number of Dispatcher control records that can be fetched into one buffer
CPCYCUSG	config	DISP_REC_FETCH_BUFFER_SIZE	1000	The maximum number of dispatching records that can be fetched into one buffer
CPCYCUSG	config	EVENT_DUP_KEYS_NUM_OF_FETCH_ROWS	1000	The maximum number of duplicate key records that can be fetched into one buffer
CPCYCUSG	config	RATED_EVENT_NUM_OF_FETCH_ROWS	1000	The maximum number of rated event records that can be fetched into one buffer
CPCYCUSG	config	REJECTED_EVENT_NUM_OF_FETCH_ROWS	1000	The maximum number of rejected event records that can be fetched into one buffer
CPCYCUSG	config	RR_BLACK_LIST_NUM_OF_FETCH_ROWS	1000	The maximum number of rerate blacklist records that can be fetched into one buffer
CPCYCUSG	config	SLEEP_TIME_BETWEEN_CHECKS_SEC	1000000	The sleep time between the checks for Notification and DB2E leftovers
CPCYCUSG	config	SUBSCRIBER_RERATE_NUM_OF_FETCH_ROWS	1000	The maximum number of subscriber rerate records that can be fetched into one buffer
CPCYCUSG	config	TECH_EVENTS_NUM_OF_FETCH_ROWS	1000	The maximum number of technical event records that can be fetched into one buffer

In addition, the process uses Light Tracer configuration parameters, which are defined in the ADJ1_CONF_SECTION_PARAM table. For more information, see [Light Tracer Parameters](#) in the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

For each process instance, an entry with the name of the process instance must be created in the APR1_CONF_HIERARCHY table.

Example:

PROCESS_NAME	PROCESS_TYPE
CPCYCUSG145	CPCYCUSG

Any parameter to be overridden at the process instance level must be added to the APR1_CONF_SECTION_PARAM table.

Example:

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE
CPCYCUSG145	config	ACCUMS_NUM_OF_FETCH_ROW S	500

For more information on parameters and the properties mechanism, see the “Configuration Parameter Mechanism” appendix.

Configuration Files

The Copy Cycle Usage process requires a configuration file (a .ccf file) that is defined via an internal configuration tool.

The environment variables that appear in the .ccf file and can be customized are defined in the op_adj_env_sh script (for more information, see the “Environment Variables” section in this chapter).

Database Connections

The process connects to the following tables during processing.

For each usage table, a connection is opened both in the home database and in the alternate database for the cycle code. The number of connections depends on the number of worker threads defined by the APR_COPY_CYCLE_USAGE_NUM_OF_THREADS environment variable. Each thread has connections to all the tables below.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Applicative	ADJ1_CYCLE_STATE	Select on the applicative database
Applicative	AUH1_REQUEST	Insert on the applicative database
Usage	APE1_ACCUMULATORS	Select on the alternate database, Insert/Update on the home database
Usage	APE1_EVENT_DUP_KEYS	Select on the alternate database, Insert on the home database
Usage	APE1_RATED_EVENT	Select on the alternate database, Insert on the home database
Usage	APE1_REJECTED_EVENT	Select on the alternate database, Insert on the home database

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Usage	APE1_SUBSCR_DATA	Select on the alternate database, Insert on the home database to AUH1_UPDATES
Usage	APE1_SUBSCRIBER_RERATE	Select on the alternate database, Insert on the home database
Usage	APE1_TECH_EVENTS	Select on the alternate database, Insert on the home database
Usage	APR1_COPY_CYCLE_USAGE_CTRL	Insert, Select on the home database
Usage	APR1_DISPATCHER_CTRL	Select on the alternate database, Insert on the home database
Usage	APR1_DISPATCHING_RECORDS	Select on the alternate database, Insert on the home database
Usage	APE1_RR_BLACK_LIST	Select on the alternate database, Insert on the home database

Admin Commands

The Copy Cycle Usage job supports the following admin commands. For information on how to execute the commands, see the “Handling Admin Commands” section in the “High Availability” chapter.

Command	Description	Parameters
SET_LIGHT_TRACER_COMMAND	Activates or stops the Light Tracer in various modes, and sends various Light Tracer parameters	<ul style="list-style-type: none"> ▪ Activation parameters: <ul style="list-style-type: none"> • <code>--turnTracer "on/off"</code> – Activates or stops the Light Tracer • <code>--turnAssert "on/off"</code> – Activates or stops Assertions • <code>--turnLogOnTrace "on/off"</code> – Activates or stops writing log messages in Light Tracer files • <code>--turnFlowTracer/turnAll "on/off"</code> – Activates or stops tracing a single flow or event that is transferred between machines • <code>--turnAll "on/off"</code> – Activates or stops all of the above. The command line can contain a single command or a combination of these commands (with the “on” or “off” parameter). ▪ <code>--moduleID "ALL"</code> ▪ Optional parameters (to be used when activating the Light Tracer): <ul style="list-style-type: none"> • <code>--threadInfo "Y/N"</code> – Specifies whether trace or assertion messages include the thread information. • <code>--timestampInd "Y/N"</code> – Specifies whether trace or assertion messages include the time stamp. • <code>--contextInd "Y/N"</code> – Specifies whether the trace or assertion messages include the context ID (for example, the event ID). • <code>--noOfMessages "<number>"</code> – Specifies the maximum number of messages that can be written into a trace before the Light Tracer shuts down. • <code>--period "5/10/20/60/120/480/1440"</code> – Specifies the maximum time period (in minutes) for which the Light Tracer can be activated.

Command	Description	Parameters
		 Note: If both the maximum number of messages and the period are specified, the Light Tracer shuts down when the first limit is reached. <ul style="list-style-type: none"> • --assertAbortInd "Y/N" – Specifies whether the application is to be aborted if the assertion condition fails. ▪ --commandType "changeBufferSize" – Resizes the Light Tracer buffer at run time. <ul style="list-style-type: none"> • --bufferSize "<number>" – The new size for the Light Tracer buffer.
SET_TRACE_SQL_COMMAND	Activates or stops SQL tracing at runtime. The command name with its parameters must be enclosed in quotation marks. <i>Example:</i> ADJ1_Send_Admin_Command_Sh illin1027 <process instance> "SET_TRACE_SQL_COMMAND --traceSqlOn \\"yes\\" --duration \\"1\\""	--traceSqlOn "<yes or no>" --duration "<minutes>" If the IS_DURATION_REQUIRED environment variable is set to 'Y', the --duration parameter is mandatory. If the duration specified in the command exceeds the allowed maximum value, SQL tracing is activated for the maximum allowed period of time only (60 minutes).  Note: The SHUT_DOWN_CHECK_FREQ parameter of the Light Tracer in the ADJ1_CONF_SECTION_PARAM table determines the frequency (defined in the number of messages) at which the SQL Trace Manager checks whether the period specified in the --duration parameter has passed. Therefore, SQL tracing is stopped after the specified duration or after this number of messages has been reached, whichever comes later.
SWITCH_LIGHT_TRACER_ON ALL	Activates the Light Tracer without the ability to specify the mode or send parameters.	N/A
SWITCH_LIGHT_TRACER_OFF ALL	Stops the Light Tracer	N/A
CPF_GracefulShutdownCommand	Shuts down the process gracefully	N/A

Examples:

- ADJ1_Send_Admin_Command_Sh hpbl820 CPCYCUSG 145 SET_LIGHT_TRACER_COMMAND –turnTracer "on" –moduleId "ALL" –threadInfo "y"
- ADJ1_Send_Admin_Command_Sh hpbl820 CPCYCUSG 145 SWITCH_LIGHT_TRACER_OFF ALL

Recovery Instructions

If the Copy Cycle Usage job crashes or is stopped by request, the subsequent Copy Cycle Usage run for the same cycle code detects that some table partitions were not completed. In this case, the job completes copying these partitions according to the records in the control table.

One of the possible reasons for Copy Cycle Usage is another failure of the home database. In this case, the Event Server and all other processes start inserting new records into the alternate database, so the existing Copy Cycle Usage control table is no longer valid. Therefore, before restarting the job, the operator must manually truncate the Copy Cycle Usage control table and clean all the data that the Copy Cycle Usage job has already inserted into the home database.

Important Remarks

- Generated libraries and reference data must not be refreshed while the Copy Cycle Usage process is running.
- The following is assumed about the database partition structure:
 - A partition contains only one cycle code and one cycle instance.
 - If the value of the customer segment is larger than the maximum value (1023), its records are written to partition A<cycle code>_B<cycle instance>_CMAX. For example, all the records with cycle code 1, cycle instance 2, and customer segment 9999 are written to partition A1_B2_CMAX.
- The progress of the Copy Cycle Usage process can be checked by running the following SQL query on the Usage database that is related to the failed cycle:

```
*****
getCopyDBProgress.sql
*****  
  
SELECT TABLE_NAME,CYCLE_CODE, SUM(NUM_OF_RECORDS) AS COPIED_REC,SUM(TOTAL_NUM_OF_RECORDS) AS
TOTAL_REC, (100*SUM(NUM_OF_RECORDS) /SUM(TOTAL_NUM_OF_RECORDS)) AS
PERCENTAGE
from APR1_COPY_CYCLE_USAGE_CTRL
GROUP BY TABLE_NAME ,CYCLE_CODE
```

59 ADJ1CCUCU – Copy Cycle Usage Clean-up

This chapter describes the Copy Cycle Usage Clean-up job.

Description

The Copy Cycle Usage Clean-up job removes data from the alternate database after the Copy Cycle Usage job is finished.

Each cycle code has a home database and an alternate database. The assumption is that all the cycles from a specific home database instance are assigned the same alternate database instance. If the entire Oracle RAC (the primary and the secondary database instance) fails, the Event Server redirects the persistent data, such as events and accumulators, to an alternate database (on a different host and connected to a different storage) so that it can continue working and processing.

After the home database is restored, the Copy Cycle Usage job copies all data that the Event Servers inserted to the alternate database (as a temporary solution) to the home database. When the Copy Cycle Usage job is finished, and the database administrator approves using the Copy Cycle Usage Clean-up job, the Copy Cycle Usage Clean-up job deletes this data from the alternate database. Only the restored data that is related to specific home database instances is deleted from the alternate database.

The Copy Cycle Usage Clean-up job deletes data from the following tables in the alternate database:

- APE1_ACCUMULATORS
- APE1_EVENT_DUP_KEYS
- APE1_RATED_EVENT
- APE1_REJECTED_EVENT
- APE1_SUBSCRIBER_RERATE
- APE1_NOTIFICATIONS
- APE1_NOTIFICATIONS_CTRL
- APR1_DISPATCHER_CTRL
- APR1_DISPATCHING_RECORDS
- APE1_SUBSCR_DATA
- APE1_SUBSCR_OFFERS
- APE1_SUBSCR_PARAMS
- AUH1_UPDATES
- APE1_RR_BLACK_LIST
- APE1_TECH_EVENTS

In addition, the Copy Cycle Usage Clean-up job deletes data from the AUH1_REQUESTS table in the home database.

These tables are cleaned according to the list of cycles that must move to the alternate database in the case of a failure in the home database. The job uses the following tables to retrieve the list of cycle codes and get the cycles that have been copied:

- ADJ1_CYCLE
- APR1_COPY_CYCLE_USAGE_CTRL

Having found the records that belong to the copied cycles, the job truncates them by table name and partition name, or deletes them by cycle code. Then, it marks the records in the APR1_COPY_CYCLE_USAGE_CTRL table as cleaned (only for the tables that have entries in the APR1_COPY_CYCLE_USAGE_CTRL table). These records are then deleted from the APR1_COPY_CYCLE_USAGE_CTRL table by the TLS1CLEANUP job (a generic clean-up utility) at the end of the day or by request.

Process Type

Batch job

Run Frequency

The job is run upon request, after a database failure and recovery, for every cycle code that has been copied from the Usage tables of the alternate database to the home database.

Activation

Command Line:	RunJobs ADJ1CCUCU BYREQ
Amdocs Monitoring & Control:	No
Script Name:	ADJ1_CopyCycleUsageCleanup_Job_Sh
Executable Name:	N/A

For more information about the activation scripts, see the “Scripts” section in the “Introduction” chapter.

Preceding Processes

The following job must run successfully before this process is activated:

- ADJ1CPCYCUSG – Copy Cycle Usage

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Log files are created and handled in the default way.

Name

ADJ1CCUCU_SCRIPT_<Date>_<Time>.log

Example:

ADJ1CCUCU_SCRIPT_21110629_165002.log

Location

The files are located in \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/.

For information about the configuration parameters that define the location of log and trace files, see the “[Configuration Parameters](#)” section in the “[ADJ1EVENTSRV – Event Server](#)” chapter.

Contents

The log file contains the process error information.

Flow

The Copy Cycle Usage Clean-up process executes the following steps:

1. Sets all the database connections.
2. Gets the connection string to the source cycle.
3. Connects to the application database.
4. Goes over the ADJ1_CYCLES table and retrieves all the cycle codes that use the same connection to the Usage database that was restarted.
5. For each cycle code, does the following:
 - a. Truncates or deletes the data that has been copied to the home database:
 - i. From the APR1_COPY_CYCLE_USAGE_CTRL table, retrieves the following values:
 - ◆ TABLE_NAME
 - ◆ PARTITION_NAME
 - ◆ CYCLE_CODE
 - ◆ TOTAL_NUM_OF_RECORDS
 - ◆ NUM_OF_RECORDS
 - ii. Checks whether the following conditions are met:
 - ◆ CYCLE_CODE is equal to the value retrieved from the ADJ1_CYCLES table.
 - ◆ The record is not marked in the CLEANED_UP column in the APR1_COPY_CYCLE_USAGE_CTRL table.
 - ◆ TOTAL_NUM_OF_RECORDS is equal to NUM_OF_RECORDS (indicating that all the data of a specific partition has been copied).

- iii. If the conditions are met, connects to the alternate database to truncate the table list by partition name or delete data by cycle code.

The following tables are truncated by table name and partition:

- ◆ APE1_ACCUMULATORS
- ◆ APE1_EVENT_DUP_KEYS
- ◆ APE1_RATED_EVENT
- ◆ APE1_REJECTED_EVENT
- ◆ APE1_SUBSCRIBER_RERATE
- ◆ APR1_DISPATCHER_CTRL
- ◆ APR1_DISPATCHING_RECORDS
- ◆ APE1_RR_BLACK_LIST

The data in the APE1_SUBSCR_DATA table is deleted by CYCLE_CODE. The data in the APE1_TECH_EVENTS table is deleted by CYCLE_CODE and DAY_IN_MONTH.

- iv. Marks the truncated records by setting the value of the CLEANED_UP field in the APR1_COPY_CYCLE_USAGE_CTRL to ‘Y’ and updates the SYS_UPDATE_DATE column with the current date.
- b. Deletes the data from the tables that are not copied to the home database:
 - Deletes the data from the following tables in the alternate database by cycle code:
 - ◆ APE1_SUBSCR_OFFERS
 - ◆ APE1_SUBSCR_PARAMS
 - ◆ APE1_NOTIFICATIONS
 - ◆ APE1_NOTIFICATIONS_CTRL
 - Deletes the data from the AUH1_REQUESTS table in the home database by request ID that equals ‘-1’.
 - Deletes the entire AUH1_UPDATES table from the alternate database.

Database Connections

The job connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)	Comments
Applicative	AUH1_REQUESTS	Delete	This table does not have an entry in the APR1_COPY_CYCLE_USAGE_CTRL table.
Reference	ADJ1_CYCLES	Select	N/A

Database Type	Table Name	Connection Type (Update, Insert, and so on)	Comments
Usage	APE1_ACCUMULATORS	Truncate	N/A
Usage	APE1_EVENT_DUP_KEYS	Truncate	N/A
Usage	APE1_NOTIFICATIONS	Delete	N/A
Usage	APE1_NOTIFICATIONS_CTRL	Delete	N/A
Usage	APE1_RATED_EVENT	Truncate	N/A
Usage	APE1_REJECTED_EVENT	Truncate	N/A
Usage	APE1_SUBSCR_DATA	Select, Delete	N/A
Usage	APE1_SUBSCR_OFFERS	Delete	This table does not have an entry in the APR1_COPY_CYCLE_USAGE_CTRL table.
Usage	APE1_SUBSCR_PARAMS	Delete	This table does not have an entry in the APR1_COPY_CYCLE_USAGE_CTRL table.
Usage	APE1_SUBSCRIBER_RERATE	Truncate	N/A
Usage	APE1_TECH_EVENTS	Delete	N/A
Usage	APR1_COPY_CYCLE_USAGE_CTRL	Update, Select, Insert	A control table
Usage	APR1_DISPATCHER_CTRL	Truncate	N/A
Usage	APR1_DISPATCHING_RECORDS	Truncate	N/A
Usage	AUH1_UPDATES	Delete	This table does not have an entry in the APR1_COPY_CYCLE_USAGE_CTRL table.
Usage	APE1_RR_BLACK_LIST	Truncate	N/A

Important Remarks

- The data belonging to a cycle code is truncated only if the all conditions described in the “Flow” section in this chapter are met.
- The tables to be cleaned up must be partitioned correctly. For a list of the tables, see the “Description” section in this chapter.

- The job does not support databases that are not partitioned. The following is assumed about the database partition structure:
 - A partition contains only one cycle code and one cycle instance.
 - If the value of the customer segment is larger than the maximum value (1023), its records are written to partition A<*cycle code*>_B<*cycle instance*>_CMAX. For example, all the records with cycle code 1, cycle instance 2, and customer segment 9999 are written to partition A1_B2_CMAX.
- When the Copy Cycle Usage Clean-up job is finished, the database administrator must make sure that the CLEANED_UP column has been added to the APR1_COPY_CYCLE_USAGE_CTRL table.

60 ADJ1SETENVAR – Set Environment Variables

This chapter describes the Set Environment Variables process.

Description

The Set Environment Variable job wraps a utility that is called by the op_adj_env_sh script. The utility sets the values of environment variables based on the profile of the runtime environment (production, NFT, or testing), process type, and process instance. For example, in a testing environment, the utility sets the EVENT_TRACING_MODE environment variable to ‘TRUE’, while in a production or NFT environment, it sets the value of this variable to ‘FALSE’.

The Set Environment Variables job can be activated manually to test the work of the ADJ1_SetProcessEnvironmentVariables_Job_inner_Sh script and define the connection codes (ADJAPPL and ADJCONFIG) that are required for the script.

Process Type

Batch job

Run Frequency

By request or via the op_adj_env_sh script for all Turbo Charging processes

Activation

Command Line:	ADJ1_SetProcessEnvironmentVariables_Job_Shell_Sh <PROCESS_CODE> <RUN_TIME_PROFILE>
Amdocs Monitoring & Control:	No
Script Name:	ADJ1_SetProcessEnvironmentVariables_Job_Sh

For more information about the activation scripts, see the “[Scripts](#)” section in the “[Introduction](#)” chapter.

Affected Applications

The Set Environment Variables job affects all Turbo Charging processes.

When any Turbo Charging process runs, the op_adj_env_sh script is called. This script, in turn, calls the ADJ1_SetProcessEnvironmentVariables_Job_inner_Sh script.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Log files are created and handled in the default way. When the Set Environment variables script is activated by the op_adj_env_sh script, it prints its log messages to the log files of the relevant process.

Name

- *Log files:*

ADJ1_SetEnvironmentVariables_\${TIME_STAMP}.log

Example:

ADJ1_SetEnvironmentVariables_20090624_115035.log

- *Console log files:*

ADJ1_SetProcessEnvironmentVariables_Job_inner_Sh_<Date>_<Time>.log

Example:

ADJ1_SetProcessEnvironmentVariables_Job_inner_Sh_20081109_130927.log

- *Operational log files:*

ADJ1SETENVAR_<Job Record Name>_\${ABP_MARKET}_<Date>_<Time>.log

Example:

ADJ1SETENVAR_BYREQ_M3G_20090624_114834.log

Location

Log files are located in the following directories:

- *Log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Console log files* – \$ABP_APP_ROOT/log/<APPLICATION_ID>/<process start time stamp>/
- *Operational log files* – \$ABP_LOG

For information about the configuration parameters that define the location of log and trace files, see the “Configuration Parameters” section in the “ADJ1EVENTSRV – Event Server” chapter.

Contents

The log files contain the following:

- *Log files* – The process parameters, the environment variables that were generated, and their values. In addition, these log files contain the process error information.

Example:

```
=====
Process parameters:
PROCESS_TYPE_CODE = ES
PROCESS_CODE      = ES100
RUN_TIME_PROFILE = ALL
=====
AC_DEBUG_FLAG=N
RUN_TIME_PROFILE [ALL] PROCESS_TYPE_CODE [ALL] PROCESS_CODE [ALL]
DESCRIPTION [APR1_CustPIExtractsReg.ccf ]
=====
ADJ_DEBUG_FLAG=N
RUN_TIME_PROFILE [ALL] PROCESS_TYPE_CODE [ALL] PROCESS_CODE [ALL]
DESCRIPTION [APR1_CustPIExtractsReg.ccf ]
=====
```

- *Console log files* – Information that the process prints upon initialization before it opens the regular log file and some messages of the process itself that are not printed via the Logger (for example, printed by the cerr, cout, and assert commands).
- *Operational log files* – The output of operational scripts.

Flow

The Set Environment Variables job executes the following steps:

1. If it was activated manually, initializes the log file.
2. Checks that the database connection exists. If the required connection codes (ADJAPPL and ADJCONFIG) do not exist, calls an operational script that generates the required database connection for the ADJ1SETENVAR task.
3. Checks whether the \$ADJ1_RUN_TIME_PROFILE and \$ADJ1_PROCESS_CODE variables are defined.
 - If the \$ADJ1_RUN_TIME_PROFILE variable is not defined, sets its value to ‘ALL’.
 - If the \$ADJ1_PROCESS_CODE variable is not defined, does the following:
 - If the \$APPLICATION_ID variable is not defined, sets the value of the \$ADJ1_PROCESS_CODE variable to ‘ALL’.
 - If the \$APPILICATION_ID variable is defined, sets the value the \$ADJ1_PROCESS_CODE variable to the value of the \$APPLICATION_ID variable.
4. Gets the value of the PROCESS_TYPE_CODE column from the GN1_SYS_PROCESS_TYPE_CFG table using the value of the ADJ1_PROCESS_CODE or APPLICATION_ID variable. If failure occurs, sets the value of the PROCESS_TYPE_CODE parameter to ‘ALL’.
5. Creates a temporary executable file.
6. Runs an SQL query on the ADJ1_PROC_PARAMS table and writes its output to the temporary executable file.
7. Grants executable permissions to the temporary file.
8. Runs the temporary executable file to set the values of environment variables based on the profile of the runtime environment (production or testing), process type, and process instance.
9. Deletes the temporary executable file.

Environment Variables

The process uses the following environment variable:

- *ADJ1_RUN_TIME_PROFILE* – Describes the environment profile. It must be defined during the installation of the environment. Any value can be set, as long as it matches the data in the ADJ1_PROC_PARAMS table.

Parameters

The following parameters must be set for the process to run properly

Parameter	Description	Default Value	Method of Changing the Parameter Manually
ADJ1_PROCESS_CODE	Process code, such as ES100 or F2E500	ALL	Operational job parameter. Can be changed through Amdocs Monitoring & Control.
ADJ1_RUN_TIME_PROFILE	Profile of the runtime environment	ALL	Operational job parameter. Can be changed through Amdocs Monitoring & Control.

Database Connections

The process connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	ADJ1_PROC_PARAMS	Select
Config	GN1_SYS_PROCESS_TYPE_CFG	Select

61 ADJ1IC_DependCheck – Dependency Checker for Implementation Compiler Libraries

This chapter describes the Dependency Checker utility for Implementation Compiler libraries.

Description

The Dependency Checker utility for Implementation Compiler libraries can be used to determine whether the Implementation Compiler Workflow must be rerun in an environment. This script must run in customization runtime environments of Turbo Charging after a core hot fix is installed.

The Dependency Checker script checks for header dependency issues between the current libraries, and the SDK and customization libraries that are installed in an environment, as part of the storage to which the environment is connected or as private libraries. The utility analyzes and reports only the items that are related to the *mapper* module.

The script returns one of the following values:

- *0* – No dependency issues were found. No need to rerun the Implementation Compiler Workflow and create new Implementation Compiler libraries.
- *100* – At least one Implementation Compiler library was compiled using a different version of a header file than the one that was used in another SDK or customization library. The Implementation Compiler Workflow must be rerun to rebuild the libraries and resolve the header dependency issues.
- *1* – The script failed.

Run Frequency

By request

Activation

Command line:	ADJ1IC_DependCheck.ksh -l <directory containing Implementation Compiler libraries> {<additional options>}
Amdocs Monitoring & Control:	No
Script Name:	ADJ1IC_DependCheck.ksh
Executable Name:	ADJ1IC_DependCheck.ksh

Shutdown

The Dependency Checker exits upon the completion of the requested task.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Runtime log file* – LibChecker.log
- *Script log file* – Console.log

Location

The log files are located in \${ABP_UTL_ROOT}/log/Lib_Checker/{Date-Time}.

Contents

The log files contain information about the status of the Dependency Checker utility and errors that occurred while running, including an error report if the Dependency Checker finds any issues.

Output Files

This section describes the report files that are created if the Dependency Checker identifies any issues.

Name

The following report files are created:

- Report.log
- Detailed_Report.log
- Report.zip

Location

Report files are located in the following directory:

~/var/m3g/projs/utl/report/Lib_Checker_Dependent/<yyyymmdd_hhmmss>/

Contents

The report files contain the following information:

- *Summary report* – Reports all the dependency issues, categorized by parameter or attribute type. When the Dependency Checker is run, the report contains only items of the ‘Header CC’ type.

Example:

```
=====
===== ISSUE TYPE: Header CC
=====
(1) ERROR - File-Check-Sum issue, File: <adj/Utils.h> is inconsistent.
There are: <1> suspected libraries
(/adjusr1/adj/env/elowrk13/abp_home/core/spbin/lib/libgaprrpfw.so)
(2) ERROR - File-Check-Sum issue, File: <adj/TimestampMacros.h> is
inconsistent.
There are: <1> suspected libraries
(/adjusr1/adj/env/elowrk13/abp_home/core/spbin/lib/libgaprrpfw.so)
```

- *Detailed report* – Reports most of the issues, categorized by the exception library. For example, if one library was compiled with version X of ACE while the other libraries were compiled with version Y of ACE, only the library with version X is reported. When the Dependency Checker is run, the report contains only items of the ‘ERROR – Header CC – File-Check-Sum’ type.

Example:

The following libraries have at least one issue:

```
=====
=====      LIBRARY      : libgaprrpfw.so
=====      Full Path   :
/adjusr1/adj/env/elowrk13/abp_home/core/spbin/lib/libgaprrpfw.so
=====      Module / BB: aprfw / gaprrpfw
=====      Cust Layer : 1
=====
(1) ERROR - Header CC - File-Check-Sum
File: <Relative-Path = adj/Utils.h> is inconsistent.

(2) ERROR - Header CC - File-Check-Sum
File: <Relative-Path = adj/TimestampMacros.h> is inconsistent.

(3) ERROR - Header CC - File-Check-Sum
File: <Relative-Path = adj/GadjRootTimestamp.h> is inconsistent.
```

- *Zip report* – Contains several CSV files and an automatic VB script. The user copies the Zip report to the PC and executes the VB script, which automatically creates a structured Excel file with the same information as in the Summary report.

Flow

The Dependency Checker utility runs the Lib Checker utility with specific parameters for the Implementation Compiler. The Lib Checker performs the following main flow:

1. Loads all the libraries associated with the *mapper* module into the cache memory.
2. Compares the loaded libraries with those in the runtime environment and saves all the compatibility issues in the cache memory.
3. Creates the report files, summarizing the compatibility issues.

Parameters

The following parameters must be set for the process to run properly.

Option	Option Name	Purpose	Additional Parameters
-h	Help	Provides information about the various options.	N/A
-l	Library Dependencies	Checks the Implementation Compiler libraries in the specified directory against the SDK and the customization libraries in the environment.	<directory containing Implementation Compiler libraries>
-n	Working Environment	Creates a working environment. This option must be included the first time that the script is executed.	N/A
-v	Debug Mode	Runs the script in debug mode, with additional details in the log.	N/A



Note: The -l, -n, and -v options are independent. Each option can be used by itself or in any combination with the other options.

Configuration Files

The Dependency Checker utility does not have configuration files of its own. For information about the configuration files of the Lib Checker utility, which is called by the Dependency Checker script, see the “Lib Checker” chapter in *Amdocs Billing Utilities Run Book*.

Screen Messages

The following messages are sent to the operator after the Dependency Checker has finished.

Message Text	Recommended Action
The lib checker did not find any issue, the environment is ready to use	No need to rerun the Implementation Compiler Workflow and create new Implementation Compiler libraries.
ATTENTION!!!!!! The lib checker found few ERRORS, which MUST be verified immediately. For more details, please view log files (*Report.log) under directory: .../var/m3g/projs/utl/report/Lib_Checker/{date-time} Report file created in zip file: .../var/m3g/projs/utl/report/Lib_Checker/{date-time}/Report.zip	The Implementation Compiler Workflow must be rerun to rebuild the libraries and resolve the header dependency issues.
Lib checker failed to execute and did not create the report. Please look on the log files under directory: .../var/m3g/projs/utl/log/Lib_Checker/{date-time}	The script failed and must be rerun.

62 UTL1MULTILEXT – Multiple Library Extract

This chapter describes the Multiple Library Extract process.

Description

The Multiple Library Extract job is responsible for running the Library Extract (UTL1LIBEXT) and Library Post-Extract (UTL1POSTLIBEX) jobs in a distributed environment of multiple hosts connected to the same database.

The Multiple Library Extract job runs the Library Extract job on all the hosts. The main host, on which the Multiple Library Extract job is executed, connects to all the other hosts via the SSH protocol using the third-party J2SSH Maverick APIs. On each host, the Library Extract job is run in two stages: preparation and execution. If the Library Extract job ran successfully on all the hosts, the Multiple Library Extract job runs the Library Post-Extract job. The Library Post-Extract job is executed only once, on the same host as the Multiple Library Extract job. If the Library Post-Extract job finished successfully, the libraries generated by the Implementation Compiler are distributed from the ADJ1_MPR_PRODUCTS table to specific paths on all the given hosts (see the “Parameters” section in this chapter), and the status of the executed jobs is updated in the ADJ1_LIB_EXT_VER_CONTROL and ADJ1_LIB_EXTRACT_CONTROL tables.

The Multiple Library Extract job is multi-threaded. The number of threads in the thread pool is smaller than or equal to the number of hosts, up to 10 threads.

Process Type

Batch job

Run Frequency

The Multi Library Extract job runs:

- At the end of the day, as part of the Release Distribution map
- By request

Participation in Generic Maps

The Library Extract job participates in the Release Distribution map, but only if there is a new Implementation Repository release.

Activation

The Multiple Library Extract job must be activated only as part of the Release Distribution map, and only if it is required for managing the Library Extract job on multiple machines.

For the job to execute its regular flow, the following environment variable must be set on the main host before the Multiple Library Extract job is run:

```
export UTL_IS_TO_RUN_MULTIPLE_LIB_EXTRACT=true
```

Otherwise, the job finishes successfully without performing any actions. This option is applicable to environments in which running the Multiple Library Extract job is not required.



Caution: To control the execution of the Library Extract and Library Post-Extract jobs, the Multiple Library Extract job must be run only once, on the main host. The main host must be capable of running the Library Extract and Library Post-Extract job locally.

Preceding Processes

The following job must run successfully before the Multiple Library Extract job is activated:

- *UTL1RELEASE* – Reference Table Distribution Release

For more information about this process, see *Amdocs Billing Utilities Run Book*.

Dependent Processes

The following process can run only after the successful completion of the Multiple Library Extract job:

- *UTL1LIBEXT* – Library Extract

Affected Applications

The Multiple Library Extract job affects the Event Server. The runtime environment switches to work with the updated libraries only after the completion and validation of the extract process.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Java Foundation log files* – JF1.log
- *Utility log files* – UTL1.log
- *Log files* – MPR1_MultipleLibExtract.log
- *Console log files* – MPR1_Console.log

Contents

The log files contain the following:

- *Java Foundation log files* – Information and error messages of Java Foundation utilities.
- *Utility log files* – Information and error messages of Revenue Management utilities.
- *Log files* – Information and error messages of the Multiple Library Extract job.
- *Console log files* – The class path and initialization information, error messages, and summary of the Multiple Library Extract job. This is the main log to be checked if the job finished successfully. This log contains the information about each thread from the thread pool as well as statistics and status for each host (the status of each host is recorded formally in the ADJ1_LIB_EXT_VER_CONTROL and ADJ1_LIB_EXTRACT_CONTROL tables).

Output Files

This section describes the output files of the process.

Name

libmpr<name>_<version_id>.so

Location

The location of the files is defined manually for each host using the ‘PATH_*’ parameters in the GN1_CONNECT_PARAMS table. For an example, see the “Parameters” section in this chapter.

Contents

Each file contains a library generated by the Implementation Compiler.

Flow

The Multiple Library Extract job performs the following flow:

1. Runs the MPR1RunJobMultipleLibExtract_Sh script. By default, the job finishes successfully immediately without performing any actions. For the job to perform its flow, the UTL_IS_TO_RUN_MULTIPLE_LIB_EXTRACT environment variable must be set ‘true’.
2. Runs the GN1SetConnPrm script to retrieve the host parameters securely. Each parameter is set as an environment variable with a special prefix. For more information, see the “Parameters” section in this chapter.

3. Runs the MPR1RunMultipleLibExtract_Sh script. This script invokes the amdocs.mpr.batch.MPRMultipleLibExtractBatch batch class as a trigger for the Multiple Library Extract Java code, which performs the following steps:
 - a. Prepares the host parameters.
 - b. Invokes the MultiLibExtractManager class to manage the library extract activities. The MultiLibExtractManager class creates the HostJobsRunner class that holds a thread pool to run each HostJob class in turn. Each HostJob class performs the following actions for a specific host:
 - i. Connects to the host using account/password@host via the SSH protocol using the APIs in the maverick-all.jar.
 - ii. Prepares the initial script of the environment to enable the Maverick APIs to run commands, one after another.
 - iii. Sets the full path of the target directory to which the libraries must be distributed on the given host (as specified in the PATH parameter).
 - iv. Runs the Library Extract job on the remote host:
 - ◆ If this is not the main host, and the Library Extract job finished successfully, the host is marked as successful.
 - ◆ If this is the main host, and if the Library Extract job finished successfully on all other hosts, the Library Post-Extract job is run locally. Then, the entire flow is marked as a success or failure.

Environment Variables

The Multiple Library Extract job uses the following environment variables.

Variable Name	Description	Valid Values/ Default Value
UTL_IS_TO_RUN_MULTIPLE_LIB_EXTRACT	Indicates whether the job must run through the entire flow ('true') or only finish successfully in order to skip to the next job ('false').	Valid values: true/false Default value: false

Parameters

For every host intended to run the Library Extract job managed by the Multiple Library Extract job, several parameters must be defined in the GN1_CONNECT_PARAMS table for the UTLHOSTS connect code (the prefix can be changed per environment) that matches the MULTLIBEXT task name in the GN1_TASK_CONNECT table.

Example:

```
Insert into GN1_TASK_CONNECT
  (TASK_NAME, DB_CODE, RUN_MODE, SESSION_ARG, SYS_CREATION_DATE, CONNECT_CODE)
Values
  ('UTL1MULTILEXT', 'M3G', 'F', 'LIBDISTRIB', TO_DATE('02/19/2013 11:16:58', 'MM/DD/YYYY
HH24:MI:SS'), 'UTLHOSTS');
COMMIT;

Insert into GN1_CONNECT_PARAMS
  (CONNECT_CODE, SERVER_TYPE, SYS_CREATION_DATE, HOST_NAME, CONN_PARAMS, DB_USER_TYPE,
DB_USER_XMKT_IND, DB_USER_MKT)
Values
  ('UTLHOSTS', 'FTP', TO_DATE('02/19/2013 11:16:58', 'MM/DD/YYYY HH24:MI:SS'), 'illin068',
'<Parameters><!--with the sample below--></Parameters>', 'W', 'N', 'M3G');
COMMIT;
```

The following parameters must be specified for each host that must run the Library Extract job managed by the Multiple Library Extract job:

- HOST_num
- ACCOUNT_num
- PASSWORD_num
- PATH_num

In addition, the optional MAX_NUMBER_OF_THREADS parameter can be used to define the number of threads to work with. Its default value is as follows:

- For one machine, the number of threads is 1 (regardless of the number of requested threads).
- For M machines where M > 1, the minimum number of threads is 2 and the maximum number of threads is 10.
 - For $1 < M \leq 10$, the number of threads is the number of machines.
 - For $M > 10$, the number of threads is 10.

The following example shows how to define two hosts and two threads in the GN1_CONNECT_PARAMS table:

Example:

```
<Parameters>
<Parameter Name="MAX_NUMBER_OF_THREADS" Value="2" IsSensitive="false"/>
<Parameter Name="HOST_1" Value="server1" IsSensitive="false"/>
<Parameter Name="ACCOUNT_1" Value="user1" IsSensitive="false"/>
<Parameter Name="PASSWORD_1" Value="password1" IsSensitive="true"/>
<Parameter Name="PATH_1" Value="/full/path/to/lib/extract/dir1" IsSensitive="false"/>
<Parameter Name="HOST_2" Value=" server2" IsSensitive="false"/>
<Parameter Name="ACCOUNT_2" Value=" user2" IsSensitive="false"/>
<Parameter Name="PASSWORD_2" Value=" password2" IsSensitive="true"/>
<Parameter Name="PATH_2" Value="/full/path/to/lib/extract/dir2" IsSensitive="false"/>
</Parameters>
```

Configuration Files

The Multiple Library Extract job does not have configuration files of its own. For information about the configuration files of the Library Extract job, which is managed by the Multiple Library Extract job, see the “Configuration Files” section in the “UTL1LIBEXT – Library Extract” chapter.

Database Connections

The Multiple Library Extract job does not connect to database tables directly. For information about the database connections of the Library Extract job, which is managed by the Multiple Library Extract job, see the “Database Connections” section in the “UTL1LIBEXT – Library Extract” and “UTL1POSTLIBEX – Library Post-Extract” chapters.

Troubleshooting

The MPR1_CONSOLE.log file shows the status of the entire flow. It indicates which hosts failed and which hosts succeeded, and whether both the Library Extract job and the Library Post-Extract job ran. In addition, it contains the error messages, if any.

The status of each job per host is also recorded in the ADJ1_LIB_EXT_VER_CONTROL and ADJ1_LIB_EXTRACT_CONTROL tables. The status of the entire flow can be understood from the job return status.

Each host contains the logs of the Library Extract and Post Library Extract job (see the “Log Files” section in the “UTL1LIBEXT – Library Extract” and “UTL1POSTLIBEX – Library Post-Extract” chapters). Additional information about the reason for the failure of the job can be found in these logs.

The following are some common issues and their solutions:

- If the Multiple Library Extract job cannot connect to a given host, the Library Extract job is aborted. As a result, this host is not marked as failed in the ADJ1_LIB_EXTRACT_CONTROL table. However, the main task fails in this case, and the reason for the failure is indicated in the console log as an error message and by the return status of the Multiple Library Extract job.
- If the job finished successfully without performing any actions, the following environment variable must be defined:
`export UTL_IS_TO_RUN_MULTIPLE_LIB_EXTRACT=true`

Recovery Instructions

Rerun the job from the Release Distribution map.

63 UTL1LIBEXT – Library Extract

This chapter describes the Library Extract process.

Description

The Library Extract process extracts the libraries generated by the Implementation Compiler libraries from the ADJ1_MPR_PRODUCTS table in the REFWAIT database to a predefined location in the shared file system.

Process Type

Batch job

Run Frequency

The Library Extract job runs:

- At the end of the day, as part of the Release Distribution map
- By request (after a failure)

Participation in Generic Maps

The Library Extract job participates in the Release Distribution map, but only if there is a new Implementation Repository release.

Activation

The Library Extract job must be activated only as part of the Release Distribution map.



Caution: Only one Library Extract job can run per shared file system. If the job runs twice, the subsequent run overwrites the previous run.

Preceding Processes

The following job must run successfully before the Library Extract job is activated:

- *UTL1MULTILEXT* – Multiple Library Extract

For more information about this process, see *Amdocs Billing Utilities Run Book*.

Dependent Processes

The following process can run only after the successful completion of the Library Extract job:

- *UTL1POSTLIBEX* – Library Post-Extract

Affected Applications

The Library Extract job affects the Event Server. The runtime environment switches to work with the updated libraries only after the completion and validation of the extract process.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Java Foundation log files* – JF1.log
- *Utility log files* – UTL1.log
- *Log files* – MPR1_LibExtract.log
- *Console log files* – MPR1_Console.log

Contents

The log files contain the following:

- *Java Foundation log files* – Information and error messages of Java Foundation utilities
- *Utility log files* – Information and error messages of Revenue Management utilities
- *Log files* – Information and error messages of the Library Extract job
- *Console log files* – The class path and initialization information of the Library Extract job

Output Files

This section describes the output files of the process.

Name

libmpr<name>_<version_id>.so

Location

The location of the files in the file system is determined by the MPR_LIB_EXTRACT_DIR environment variable (see the “[Environment Variables](#)” section in this chapter).

Contents

Each file contains a library generated by the Implementation Compiler.

Flow

The Library Extract job performs the following flow:

1. Gets the release version from the ADJ1_RELEASE_INFO table.
2. Checks whether a new library is available, according to the last record in the ADJ1_LIB_EXT_VER_CONTROL and ADJ1_MPR_PRODUCTS tables.
 - If a library exists in the ADJ1_MPR_PRODUCTS table but not in the ADJ1_LIB_EXT_VER_CONTROL table, continues
 - If a library exists in both tables, but its status in the ADJ1_LIB_EXT_VER_CONTROL table is ‘F’ (failure), continues
 - If a library does not exist in either table, the libraries are not extracted and the job finishes with the status of success.
 - If the library exists in both tables, and its status in the ADJ1_LIB_EXT_VER_CONTROL table is ‘S’ (success), the libraries are not extracted and the job finishes with the status of success.
3. Sets the status of the file in the ADJ1_LIB_EXTRACT_CONTROL table, which contains the status of each library per host, to ‘R’ (ready to be extracted).
4. For the release version, extracts the generated libraries from the ADJ1_MPR_PRODUCTS table to the file system. The number of threads that perform the extract is defined in the property file (see the “Property Files” section in this chapter). Each thread performs the following:
 - a. Receives the name of the library to extract.
 - b. Updates the status of the file in the ADJ1_LIB_EXTRACT_CONTROL table to ‘I’ (in process).
 - c. Reads the contents of the library from the database and extracts it to the file system.
 - d. Unzips the library.
 - e. Calculates the checksum of the file in the file system, and compares it to the checksum of the file in the ADJ1_MPR_PRODUCTS table.
 - f. Updates the status of the file in the ADJ1_LIB_EXTRACT_CONTROL table to ‘S’ (success) or ‘F’ (failure):
 - *S* – The extraction process is finished, and the checksum for the library has been validated.
 - *F* – The extraction process has failed. In this case, the description of the problem is written to the ERROR_MESSAGE field of the ADJ1_LIB_EXTRACT_CONTROL table. The job fails if at least one library does not pass the checksum validation.

5. When all the threads have finished their work, performs additional validations. These validations are configurable via the MPR1_LibExtValidations.xml file (for more information, see the “Validation File” section in this chapter). The out-of-the-box file includes the following validations:
 - a. *Lib Checker* (for more information, see *Amdocs Billing Utilities Run Book*)
 - b. *Rater Init* (for more information, see the “ADJ1ICWFlow – Implementation Compiler Workflow” chapter)

The logs of the validations are located in the same directory as the logs of the job itself (see the “Log Files” section in this chapter).

If a validation is not successful, the job fails.

6. If the validations are successful, removes old libraries from the file system according to a SONAR template variable defined in the properties file. The clean-up is performed at the host level.

Environment Variables

The Library Extract job uses the following environment variables.

Variable Name	Description	Valid Values/ Default Value
MPR_BIN_EXTRACT_COMMON_DIR	The location of Implementation Compiler binaries in the storage.	~/abp_home/mapper/bin
MPR_EXTRACT_COMMON_DIR	The location of Implementation Compiler files in the storage.	~/abp_home/mapper/
MPR_LIB_EXTRACT_COMMON_DIR	The location of Implementation Compiler libraries in the storage. This location must be included in the LD_LIBRARY_PATH variable for each environment.	~/abp_home/mapper/lib
MPR_LIB_EXTRACT_DIR	The location in the shared file system to which the generated Implementation Compiler libraries must be extracted. This location must be included in the LD_LIBRARY_PATH variable for each environment.	~/pbin/mapper/lib

Configuration Files

This section describes the configuration files of the Library Extract job.

Property Files

The Library Extract job has the following property files:

- *MPR1JF.properties* – Defines the number of threads to run. This number must be greater than 0 but lower than the number of available processors. The following value is used out of the box:

```
amdocs.thread.executor.pool.size.MPR1LIBEXT=5
```

- *MPR1App.properties* – Defines the number of library versions to keep (including the current version) after a successful extraction to the file system. The following value is used out of the box:

```
amdocs.mapper.lib.extract.number.of.versions.to.keep=2
```

If the clean-up of old libraries is not required, the value of this property must be set to '0'.

Validation File

The MPR1_LibExtValidations.xml file defines the additional validations that the Library Extract must perform when all the threads are finished. Following is the out-of-the-box file:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <validations>
- <validation name="Rater Init" script_name="MPR1ICWFlow_Rater_Init.ksh" execute="true"
description="Run rater init with the compiled lib">
- <parameters>
<parameter name="conn" sensitive="true"
type="env">>${MPRREFW_USER}/${MPRREFW_PASSWORD}@${MPRREFW_INSTANCE}</parameter>
</parameters>
</validation>
- <validation name="Lib Checker" script_name="JF1LibChecker_Sh" execute="true" description="Run
lib checker utility with mapper module">
- <parameters>
<parameter name="r" sensitive="true" type="env" description="regular mode" />
<parameter name="m" sensitive="true" type="env" description="mapper module">mapper</parameter>
</parameters>
</validation>
</validations>
```

This file can be used to skip the out-of-the-box validations or to add custom validations. Custom scripts used for validations must return an answer.

Database Connections

The Library Extract job connects to the following tables.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	ADJ1_LIB_EXTRACT_CONTROL	Update and Insert
Application	ADJ1_LIB_EXT_VER_CONTROL	N/A

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Reference	ADJ1_MPR_RPRODUCTS	Select
Reference	ADJ1_RELEASE_INFO	Select

Recovery Instructions

Rerun the job from the Release Distribution map.

64 UTL1POSTLIBEX – Library Post-Extract

This chapter describes the Library Post-Extract process.

Description

The Library Post-Extract job runs per host after the Library Extract job has finished on all hosts and makes sure that the Library Extract job runs again only if there is a new Implementation Repository release.

Process Type

Batch job

Run Frequency

The Library Post-Extract job runs:

- At the end of the day, as part of the Release Distribution map
- By request (after a failure)

Participation in Generic Maps

The Library Extract job participates in the Release Distribution map.

Activation

The Library Post-Extract job must be activated only as part of the Release Distribution map.

Preceding Processes

The following job must run successfully before the Library Post-Extract job is activated:

- *UTL1LIBEXT* – Library Extract

Dependent Processes

The following process can run only after the successful completion of the Library Extract job:

- *UTL1SYNSWITCH* – Reference Table Distribution Switch

For more information about this process, see *Amdocs Billing Utilities Run Book*.

Affected Applications

The Library Post-Extract job affects the next run of the Library Extract job. For more information, see the “Flow” section of the “*UTL1MULTILEXT*– Multiple Library Extract” chapter.

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

The names of the log files are:

- *Java Foundation log files* – JF1.log
- *Utility log files* – UTL1.log
- *Log files* – MPR1_PostLibExtract.log
- *Console log files* – MPR1_Console.log

Location

Log files are located in the following directory:

\$ABP_UTL_ROOT/logs/<job name, such as mprPostLibext>/<date>

Contents

The log files contain the following:

- *Java Foundation log files* – Information and error messages of Java Foundation utilities
- *Utility log files* – Information and error messages of Revenue Management utilities
- *Log files* – Information and error messages of the Library Post-Extract job
- *Console log files* – The class path and initialization information of the Library Post-Extract job

Flow

The Library Post-Extract job performs the following flow:

1. Gets the release version from the ADJ1_RELEASE_INFO table.
2. For the release version, checks whether the status of all the rows in the ADJ1_LIB_EXTRACT_CONTROL table is ‘S’ (success).
 - If so, inserts a row in the ADJ1_LIB_EXT_VER_CONTROL table with the status of ‘S’
 - If not, inserts a row in the ADJ1_LIB_EXT_VER_CONTROL table with the status of ‘F’.

Database Connections

The Library Post-Extract job connects to the following tables.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Application	ADJ1_LIB_EXTRACT_CONTROL	Select
Application	ADJ1_LIB_EXT_VER_CONTROL	Delete, Insert
Reference	ADJ1_RELEASE_INFO	Select

Recovery Instructions

Rerun the job from the Release Distribution map.

65 membdbdump – AIMOS Dump

This chapter describes the AIMOS Dump utility that is used for dumping and loading the data stored in Amdocs In-Memory Object Storage (AIMOS).

Description

For performance reasons, the Event Server carries out all of its operations using its internal memory, AIMOS.

The AIMOS Dump utility enables extracting the contents of AIMOS to files and loading this data later into AIMOS for analysis or reuse.

Run Frequency

By request

Activation

The AIMOS Dump utility works in two modes: dump (-d) and load (-l).

- *Dump mode* – Executed on the same machine as the Event Server whose AIMOS data is to be stored.
- *Load mode* – Executed on the same machine as the Event Server into which AIMOS data is to be loaded.

The following prerequisites must be met before the utility can be run:

- Both active and shadow Event Servers must be down.
- Before loading the shared memory from the dump, the existing shared memory with the same key must be deleted.
- The size of the free RAM must be greater than the total size of the shared memory area to be created and populated with the data.
- For a full dump, the process must run from a directory with enough disk space for output files. The amount of space required depends on the amount of used arena space and the type of information that is stored in the arena. The recommended amount of space is at least a half of the arena size.

Command Line:	<ul style="list-style-type: none">■ <code>memdbdump -d { [-k <root_key>] [-p <procname>] }</code>■ <code>memdbdump -l <file name prefix> -s <shared memory size></code> <p> Note: Either the <code>-d</code> (dump) or <code>-l</code> (load) option must be specified.</p>
Amdocs Monitoring & Control:	No
Script Name:	No
Executable Name:	memdbdump

Examples:

- `memdbdump -d -k 119`

Dumps the shared memory of root arena key 119 into a series of dump files named `full_119.n`, where `n` is a number starting from 1

- `memdbdump -d -p ES100`
Dumps the shared memory of Event Server 100 into a series of dump files as described in the previous example
- `memdbdump -l full_119 -s 218103808`
Loads the dump from the *full_119.n* files into the shared memory of the total size set in the database or reported by the AIMOS Dump utility during the dump operation

For more information on the command line options, see the “Parameters” section in this chapter.

Shutdown

The AIMOS dump utility exits upon the completion of the requested task.

Log Files

No log files are created. Progress logs are written to the standard output device.

Input Files

The AIMOS Dump utility requires the following input files:

- *In dump mode (-d)* – None
- *In load mode (-l)* – The name of the input files is specified as one of the command line parameters (`-l <file name prefix>`).

Output Files

The AIMOS Dump utility creates the following output files:

- *In dump mode (-d)* – A series of dump files containing the dumped memory. These files use the following naming convention:
`full_<root arena key>.<file number>`
where:
 - `<root arena key>` is the decimal representation of the arena key reported by AIMOS MetaInfo (for more information, see the “[metainfo – AIMOS MetaInfo](#)” chapter).
 - `<file number>` is the serial number of the specific output file among the dump files that were created (a natural number starting from 1).
- *In load mode (-l)* – None.

Flow

To use the AIMOS Dump utility:

1. Dump the shared memory into a series of files.
2. (*Optional*) Copy the dump files to another machine or environment.
3. Load the shared memory from the dump files.

The AIMOS Dump utility executes the following flow:

1. Parses the command line options and ensures that the selected options are compatible.
2. If the dump (-d) option is specified, executes the dump flow:
 - a. If the process name (-p) is specified instead of the root key (-k), executes the *memdbkey* flow to retrieve the shared memory key (see the “Shared Memory Space Allocation” section in the “ADJ1EVENTSRV – Event Server” chapter).
 - b. Maps the header of the shared memory and gets the size of the entire memory.
 - c. Maps the entire memory.
 - d. Dumps the contents of the shared memory into files by blocks of about 2 GB, compressing the data using the gzip algorithm. Each block is written into a separate file.
3. If the load (-l) option is specified, executes the load flow:
 - a. Parses the given file name to gather the shared memory key
 - b. Takes the shared memory size from the command line parameter
 - c. Creates a shared memory of the combined size of all the files, along with the specified shared memory key
 - d. Loads all the files to the shared memory

Environment Variables

The following environment variables affect the process. Most of these environment variables have default values, which are set automatically according to the type of the running environment. Therefore, in most cases, they must not be modified or customized.

Name	Description	Default Value	Method of Changing the Variable
Environment variables related to the set-up of the working environment			
ADJ1_SHMEM_PATH	The location of the shared memory	N/A	N/A
HOME	The location of shared memory if the ADJ1_SHMEM_PATH variable is not defined	N/A	N/A

Name	Description	Default Value	Method of Changing the Variable
PROCESS_GROUP_ID	The identifier of the process group	N/A	N/A
PROCESS_TYPE_ID	The identifier of the process type	N/A	N/A
USER	The user name of the environment in which the script runs if the USERNAME variable is not defined	N/A	N/A
USERNAME	The user name for the environment in which the script runs	N/A	N/A

If the PROCESS_GROUP_ID or PROCESS_TYPE_ID variables are not defined, this data is taken from the GN1_SYS_PROC_INSTANCE_CFG table in the database. The following environment variables determine the database to which the utility can connect, to retrieve this data.

ADJCONFIG_DB_INSTANCE	The instance of the database that contains the relevant GN1_SYS_PROC_INSTANCE_CFG table. It must point to the APPC database account.	N/A	Local setenv script
ADJCONFIG_DB_PASSWORD	The password for the database that contains the relevant GN1_SYS_PROC_INSTANCE_CFG table. It must point to the APPC database account.	N/A	Local setenv script
ADJCONFIG_DB_USER	The user name for the database that contains the relevant GN1_SYS_PROC_INSTANCE_CFG table. It must point to the APPC database account.	N/A	Local setenv script

Parameters

The following command line options are available.

Option	Option Name	Purpose	Additional Parameters	Valid in Dump Mode	Valid in Load Mode
-h	Help	Display help information that explains how to run the utility.	N/A	✓	✓
-d	Dump	Dump the shared memory into a series of files.	N/A	✓	-
-l	Load	Load the shared memory from a specified series of files.	filename		✓

Option	Option Name	Purpose	Additional Parameters	Valid in Dump Mode	Valid in Load Mode
-k	Root Key	<p>Provide a shared memory key for the root arena to use instead of the default calculated key.</p> <p>On non-NT platforms, either -p or -k must be specified.</p> <p> Note: In Windows, the root key is RootArena-<ArenaNumber>, where ArenaNumber is determined by the ADJ1_SH_MEM_ID environment variable.</p> <p>For instance, if ADJ1_SH_MEM_ID is 100, the RootArena key is RootArena-100.</p>	root_key	✓	-
-a	Allocated Memory	Dump only the allocated part of the shared memory.	N/A	✓	-
-n	Directory Name	Select the directory for the dump files.	dirname	✓	-
-m	Maximum Size	Set the maximum size of a dump file.	maxsize	✓	-
-s	Shared Memory Size	<p>Set the total size of the shared memory as defined in the database or reported by the AIMOS Dump utility during the dump operation. This size can also be calculated using the following command:</p> <pre>for FILE in <file name> ; do gzip -c \$FILE wc -c ; done awk '{Total += \$1;} END {print Total;}'</pre> <p>For full load operations, this option is mandatory.</p>	loadsize	-	✓
-p	Process Name	<p>Set the process name (for example, ES200). The process name is used to calculate the shared memory key.</p> <p>On non-NT platforms, either -p or -k must be specified.</p>	procname	✓	-
-g	Process Group	Set the process group. The process group is used to calculate the shared memory key. If it is required and not provided, the process group is retrieved from the database.	procgroup	✓	-

Database Connections

The process connects to the following table during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)
Configuration	GN1_SYS_PROC_INSTANCE_CFG	Read-only

Recovery Instructions

After the problem is fixed, rerun the utility with the same configuration.

66 metainfo – AIMOS MetaInfo

This chapter describes the AIMOS MetaInfo tool used for viewing data stored in AIMOS.

Description

To achieve the highest level of performance, the Event Server carries out all of the operations using its internal memory, also referred to as AIMOS (Amdocs In-Memory Object Storage). Due to the nature of AIMOS, its data cannot be viewed by external tools. The AIMOS MetaInfo tool enables the user to view information regarding AIMOS as well as the information stored within AIMOS.

AIMOS MetaInfo consists of several parts that perform the following actions:

- *Retrieving development and test information* – Viewing the list and structure of the data types stored in AIMOS. This part of AIMOS MetaInfo works in an interactive mode, namely, enables the user to choose the requested information from a menu and writes the results into a local log file. The information can be retrieved using the `-q` or `-lm` option. For the full list of options, see the “[Parameters](#)” section in this chapter. To get help regarding these options, run `metainfo -ha` or `metainfo -hc`.
- *Retrieving last actions* – Viewing the last actions performed by the Guiding to Customer, Event Processing, and Session Expiration threads in the Event Server. The information can be retrieved using the `-q` option. For more information, see the “[Last Actions Report](#)” section in this chapter.
- *Modifying AIMOS using AIMOS Demo* – Creating and modifying memory content (for example, creating new arenas or allocating memory). Examples of AIMOS Demo actions are `--t1` and `--tc` options. Because these actions are used only for development in a demo environment and are not supported in a regular environment, they are not covered in this document.

In addition, a different tool – AIMOS Viewer – can be used to retrieve customer, subscriber, or generic reference table data in XML format.

Run Frequency

By request

Activation

The AIMOS MetaInfo tool is executed on the same platform as the Event Server that is to be investigated.

Activation Using the metainfo Executable

Command Line:	<ul style="list-style-type: none"> ■ Unix metainfo -<Option> [{<Parameters>}] ■ NT <ul style="list-style-type: none"> • metainfo -<Option> [{<Parameters>}], or • metainfoD -<Option> [{<Parameters>}]
Amdocs Monitoring & Control:	No
Script Name:	N/A
Executable Name:	<ul style="list-style-type: none"> ■ Unix – metainfo ■ NT – metainfo or metainfoD



Note: The MetaInfo tool cannot run and collect data more frequently than every 30 seconds.

Examples:

- Activation
 - metainfo
 - Displays help information on most used AIMOS MetaInfo actions.
- Development information retrieval options
 - metainfo -q 119
 - Displays global information regarding the arena with arena key 119, for example, free and allocated memory.
 - metainfo -df
 - Displays system default values used when allocating a new arena, such as the default arena size.
 - metainfo -v RootArena-119 RootArena-11235
 - Displays the version of the currently used AIMOS MetaInfo tool and the version used in arenas with the RootArena-119 and RootArena-11235 keys.

For more information about command line options, see the “Parameters of the metainfo Executable” section in this chapter.

Activation Using the ADJ1_AimosViewer_Sh Script

For a selected number of frequently used options for retrieving business information, the AIMOS Viewer tool can be activated using the ADJ1_AimosViewer_Sh script. This script is available in Unix only.

AIMOS Viewer is a tool for viewing and analyzing the contents of AIMOS (arena data) related to resources or subscribers. It provides access to the following data:

- A list of resources loaded into an arena
- Details of a given resource

- A list of subscribers loaded into an arena
- Details of a given subscriber
- Information that is not accessible using the metainfo executable or the ADJ1_extract_AIMOS_data_Sh script:
 - Complexes
 - Reservations for open sessions
 - A list of generic applicative tables (GATs)
 - The contents of a specific generic applicative table (GAT)
 - Information used for session-based guiding

This information is presented in XML files. The ABP_LOG environment variable holds the name of the directory used to store these XML files. If this variable is not set, the current working directory is used.

Command Line:	<ul style="list-style-type: none"> ■ ADJ1_AimosViewer_Sh <APPLICATION_ID> -<Option> [{<Parameters>}] ■ ADJ1_AimosViewer_Sh -key <ARENA_KEY> -<Option> [{<Parameters>}]
Amdocs Monitoring & Control:	No
Script Name:	ADJ1_AimosViewer_Sh
Executable Name:	AimosViewer



Note: Information on more than one subscriber, customer, or resource may be requested in a single command.

Examples:

- ADJ1_AimosViewer_Sh ES100 -subsc 123/456
Displays information regarding subscriber 123 of customer 456.
- ADJ1_AimosViewer_Sh ES100 -sessions 123/456
Displays the session data of subscriber 123 of customer 456.
- ADJ1_AimosViewer_Sh -key 0x4109d14e –subsclist
Displays a list of subscribers in memory arena 0x4109d14e.

For more information about command line options, see the “Parameters of the ADJ1_AimosViewer_Sh Script” section in this chapter.

Shutdown

When working in the batch mode (for business information retrieval), the tool finishes running upon completing the request.

When working in the interactive mode (for development information retrieval), the user can exit the tool by choosing the 0 (EXIT) option from the menu.

Preceding Processes

Because the AIMOS MetaInfo tool connects to an existing shared memory, it can be executed only after running an Event Server or loading an AIMOS dump from files. For more information, see the “memdbdump – AIMOS Dump” chapter.

Log Files

The log file name is *metainfo<option>.log*. For example, when running *metainfo* with the *-q* option, the created log file is *metainfo-q.log*.

Log files are created in the current directory. If there are no permissions to write to the current directory, files are written to the *\tmp* directory.

In addition to some basic data displayed in the log files, these files contain the names of created output files.

Output Files

This section describes the output files of the AIMOS MetaInfo tool.

Business Information

During business information retrieval, output files are written into *\${ABP_LOG}* . These files have unique names and are not overwritten by other output files of the tool. The output of AIMOS Viewer is an XML file. Other files get the extension of *.log*.

Development Information

During development information retrieval, logs and output are written to the same log file (see the “Log Files” section in this chapter).

Last Actions Report

The Last Actions Report is produced:

- Automatically, when an Event Server is activated (after a fresh start of an active Event Server or a transition from the shadow to the active state). In this case, the report appears in the Event Server application log and shows the actions that were in process when the Event Server failed or was shut down.
- Manually, by means of a user request using the AIMOS MetaInfo tool. In this case, the report appears in the console window and shows the actions at the moment of the request.

To manually request a Last Actions Report:

1. Run the following command:

```
metainfo -q <arena key>
```

A list of suboptions appears.

2. Enter the number associated with the STATUS Last Action suboption.

The Last Actions Report consists of three sections:

- Event Processing (reported as Rating) actions
- Session Expiration actions
- Guiding to Customer (reported as FR) actions

Each section contains records of the same type, as displayed in the following sample report extract.

Example:

Rating actions

Life	Thread	Version	State	Customer	Subscriber	Session	Message
ID	ID			ID	ID	ID	ID
00000004	48	1	Processed	0	0	0 (.....)	1019
00000004	62	1	Processed	0	0	0 (.....)	1019
00000001	54	1	Processed	1144	527	10000 (.....'..)	100

See 'switch(messageID)' statement in RaterBS::doProcess() for message IDs

SessionExpiration actions

Life	Thread	Version	State	Customer	Subscriber	Session	Session
ID	ID			ID	ID	ID	Status
00000004	41	1	In use	286	136	105 (.....i..)	0
00000003	41	1	Processed	737	332	1001 (.....ð..)	0
00000002	41	1	Processed	54	25	32 (..... .)	0
00000001	41	1	Processed	1144	527	10003 (.....'..)	0

See 'switch (status)' statement in SessionExpirationBS::handle_timeout() for session statuses

FR actions

Life	Thread	Version	State	Message	Resource	Resource
ID	ID			ID	Type	Value
00000003	44	1	Processed	0	1	001400222004
00000002	44	1	Processed	0	1	269714
00000001	44	1	Processed	0	1	765432

See 'flyData.m_messageType' statement in FRBS::doProcess() for message IDs

All sections have the following common columns:

- *Life ID* – A sequential number given to the current instance of the active/shadow pair that serves as the active instance. Every time a switch between the active and the shadow process occurs, the Life ID is incremented. The report contains the last actions of all Event Server activations in the shared memory that were not removed manually.



Note: The report that is automatically produced when an Event Server becomes active is exactly the same report as the one that would be produced by a manual request immediately before the last time that the Event Server became inactive.

- *Thread ID* – An internal ACE thread ID. The thread is recorded in the application log.

Example:

```
System thread id: [2665672], ACE Thread ID [34] is working for task [SessionExpiration]
```

- *Version* – The version of the record, used internally by the application.
- *State* – The current state of the action. There are two valid values:
 - *Processed* – The action was completed.
 - *In use* – The action is in progress.

Other, action-specific columns carry information describing the action.



Note: The Session ID is presented in two forms (as a number and as a string) because it is set to a number during development and testing, but it must be a string for the Diameter protocol.

Help File

The help file is created automatically when running the tool. The help file is *metainfo.help*.

The help file is created in the current directory. If there are no permissions to write to the current directory, the file is written to the *\tmp* directory.

Flow

When retrieving development information, the AIMOS MetaInfo tool works in the interactive mode. It suggests a number of options regarding the types of information to be retrieved from AIMOS. The options are displayed in a menu from which the user can select the requested option. In some cases, a submenu is displayed. After completing the execution of the requested option, the user can choose more options until the investigation of AIMOS is complete.

After an option has been executed, the menu is not displayed automatically again. Rather it is displayed upon request when choosing the relevant *-h* option (which stands for *help*).

To display the menu of AIMOS MetaInfo, run one of the following commands:

- `metainfo` or `metainfo -hot` – Displays the most useful actions. The actions displayed via this option are also available via other help options.

This information consists of the following four sections:

- *General information* – Displays such information as the AIMOS MetaInfo tool version, current machine, log directory, working directory, and more
- *Usage* – Specifies all the command line options and the parameters required to use them
- *Synopsis* – Provides a short description of each command line option
- *Example* – Contains the examples of usage for various options and their parameters

- `metainfo -h` – Displays a complete list of available actions.
- `metainfo -hh` – Displays information regarding the existing types of help.
- `metainfo -ha` – Displays the actions that show information regarding an arena.
- `metainfo -hc` – Displays the actions that show information that is common to all arenas managed by AIMOS (for example, the default values of the Memory Arena Allocator).
- `metainfo -hd` – Displays the demo options.

Environment Variables

AIMOS MetaInfo uses the following environment variables:

- *ABP_LOG* – Defines the location for log files
- *MAAINFO_DISABLE_LOGS* – Defines that no log files are to be created by AIMOS MetaInfo
- Environment variables that define the minimum severity level of log messages:
 - *MAA_LOG_LEVEL* – Defines the minimum severity level of messages to be printed to the log. The default value is ‘T’ (all messages are printed).
 - *MAA_CONSOLE_LOG_LEVEL* – Defines the minimum severity level of messages to be printed to the console log. The default value is ‘E’ (only Error and Fatal messages are printed).
 - *MAA_CONSOLE_STATUS_MSG_LEVEL* – Defines the minimum severity level of messages to be printed to the console status log (such as messages about the remaining free memory). The default value is ‘T’ (all messages are printed).

Valid values:

- *T* – Trace
- *D* – Debug
- *I* – Info

- *W* – Warning
- *E* – Error
- *F* – Fatal
- *N* – No messages
- *ADJ1_SHMEM_PATH* – Defines the location of the shared memory

Parameters

To view information regarding the arenas, an arena identifier (arena key) must be supplied as part of the parameters.

- In a Unix environment, the arena identifier type is a number. Its value can be obtained using the *-lm* option.
- In a Windows environment, the arena key is a string and its value is `RootArena-<ArenaNumber>`, where ArenaNumber is determined by the `ADJ1_SH_MEM_ID` environment variable.

Subsequent subsections specify the main available options of AIMOS MetaInfo.

Parameters of the metainfo Executable

The following table describes the main parameters of the metainfo executable.

Option	Parameters	Description	Information Type	Available in Unix	Available in Windows
-ls	-	Displays a list of segments currently found in the shared memory (AIMOS). Not all the segments displayed in the output tables are actual arenas. The Arena Version column can be used to identify actual arenas. Only those segments that are actual arenas have versions. Otherwise, this column states “Not an Arena”.	Development	✓	-
-lm	-	Displays a list of arenas currently found in the shared memory (AIMOS). The arena identifier is in the Arena Key column in the output table of this option.	Development	✓	-
-q	ArenaKey	Displays general information regarding the arena, for example, free and allocated memory, or active anchors.	Development	✓	✓
-qm	ArenaKey	Displays information regarding objects that are stored or can be stored in a given arena, including an XML file with the contents of these objects.	Development	✓	✓

Option	Parameters	Description	Information Type	Available in Unix	Available in Windows
-df	-	Displays system default values used when allocating a new arena, such as the default arena size.	Development	✓	✓
-ds	MemorySegmentKey	Displays the dump of a given shared memory segment (not necessarily an arena).	Development	✓	✓
-dm	ArenaKey	Displays the shared memory dump of a given arena.	Development	✓	✓
-v	ArenaKey	Displays the version of the currently used AIMOS MetaInfo tool and the version used in a given list of arenas.	Development	✓	✓

Parameters of the ADJ1_AimosViewer_Sh Script

The following table describes the options of the ADJ1_AimosViewer_Sh script.

Option	Parameters	Description	Information Type	Available in Unix	Available in Windows
-subsc	[<SUBSCRIBE_R_ID>/<CUSTOMER_ID>]	Displays information regarding the subscribers of a requested customer. If a subscriber is specified, only the information regarding that particular subscriber is displayed.	Business	✓	-
-subsclist	N/A	Displays a list of all subscribers in the given arena or Event Server. This is the only option that is available for a replication Event Server because its shared memory does not contain other data.	Business	✓	-
-res	<RESOURCE_VALUE>/<RESOURCE_TYPE>	Displays information regarding a requested resource.	Business	✓	-
-reslist	N/A	Displays a list of all resources in the given arena or Event Server.	Business	✓	-

Option	Parameters	Description	Information Type	Available in Unix	Available in Windows
-sessions	[<SUBSCRIBE_R_ID>/<CUSTOMER_ID>]	Displays information regarding the sessions (both primary sessions and service sessions) and reservations of a requested customer. If a subscriber is specified, only the sessions of that particular subscriber are displayed. If the events of a subscriber or customer were redirected to a shared allowance customer, related information is also displayed.	Business	✓	-
-sbg	<SESSION_ID>	Displays information used for session-based guiding (such as the subscription ID). <SESSION_ID> is either the network session ID or the shortened session ID that the Event Server inserts into the SESSION_ID_HASH column of the APR1_SESSION_GUIDING table.	Business	✓	-
-gattablelist	-	Displays the list of the custom generic applicative tables (GATs) that were loaded into the shared memory.	Business	✓	-
-gatablesdata	-	Displays the values from the custom generic applicative tables (GATs) that were loaded into the shared memory. The values to be displayed must be configured in the ADJ1_GAT_FIELDS table by setting the corresponding LOADED_INTO_MEMORY field to 'Y'.	Business	✓	-
-gattabledata	<TABLE_ID>	Displays the values from a requested custom generic applicative table (GAT) that was loaded into the shared memory. The values to be displayed must be configured in the ADJ1_GAT_FIELDS table by setting the corresponding LOADED_INTO_MEMORY field to 'Y'.	Business	✓	-
-d		Enables viewing the output (only used for debugging).	Development	✓	-

Recovery Instructions

After the problem is fixed, rerun the tool with the same configuration.

67 ERT – Error Entity Recovery Tool

This chapter describes the Error Entity Recovery Tool (ERT).

Description

The Error Entity Recovery Tool is a utility for recycling events and their products that originally failed to be persisted into the database.

Depending on a configuration parameter, the entire Persistence Writer transaction that failed or only the problematic entities within the transaction are written into the APE1_ERROR_ENTITIES and APE1_ERR_ENTITIES_DETAILS tables. From those tables, the tool retrieves the following details for use in recycling:

- Events
- Accumulators
- Notification and dispatching records
- Duplicate check keys
- Indication whether the entity was rejected

The Error Entity Recovery Tool works in the following modes:

- *Export* – Extracting the data from the APE1_ERROR_ENTITIES and APE1_ERR_ENTITIES_DETAILS tables into an XML file so that the operator can investigate the data and correct it. (For example, the operator might change the size or type of a value or correct a problem in the database table.)
- *Import* – Importing the corrections from the XML file into the usage tables.
- *Full* – Automatically exporting entities from the APE1_ERROR_ENTITIES and APE1_ERR_ENTITIES_DETAILS tables and then importing them into the usage tables without manual correction of the XML file. This mode is useful if all the entities in the table failed because of a database problem that has been fixed.

Process Type

Batch job

Run Frequency

By request.

When a record is inserted into the APE1_ERROR_ENTITIES table, the Event Server generates an alert to make the existence of the record known to the operator.

Activation

Command Line:	<ul style="list-style-type: none"> ▪ Export and Full modes: <code>-processname <process name> -cyclecode <cycle code> {<optional parameters>}</code> <i>Example:</i> <code>-processname ERT_EXPORT -cyclecode "1"</code> <code>-processname ERT_FULL -cyclecode "1" -eewhereclause "TRANSACTION_NUMBER=1048353368136113" -eedwhereclause "ENTITY like 'RTD%'"</code> ▪ Import mode: <code>-processname <process name> -importkey <location of input XML file> -cyclecode <cycle code></code> <i>Example:</i> <code>-processname ERT_IMPORT -importkey 1346599238745 -cyclecode "1"</code>
Amdocs Monitoring & Control:	Yes
Script Name:	<ul style="list-style-type: none"> ▪ ERT_Export_Run.sh ▪ ERT_Import_Run.sh ▪ ERT_Full_Run.sh
Executable Name:	amdocs.adjust.errent.main.Main

Log Files

It is recommended that the operator check the log file after each run to verify that the process has succeeded.

Name

ert_<date_time>.log

Location

\$ABP_ERT_ROOT/logs

Contents

The log files contain information about the job status (Info/Warning/Error).

Input Files

The Import mode of the Error Entity Recovery Tool requires the corrected version of the XML file created by the Export mode (the version that is the output of editing by the support person).

Name

The names of input files follow this naming convention:

- error_entity_<time_stamp>_0.xml
- error_entity_<time_stamp>_1.xml
- ... error_entity_<time_stamp>_n.xml

The number of files (named *<time_stamp>_0.xml, *<time_stamp>_1.xml, *<time_stamp>_2.xml, and so on) depends on the number of transactions to which the entities in the APE1_ERROR_ENTITIES table belong and on the MaxTransactionsInFile parameter.

The *error_entity* prefix can be overridden by setting the WorkingFilePrefix parameter in the ADJ1_CONF_SECTION_PARAM table to a different value.

Location and Format

- *Location* – \$ABP_ERT_ROOT/work/data/<folder name by time stamp>/
- *Format* – XML

Contents

As described in the “Output Files” section in this chapter

Output Files

The Export mode of the Error Entity Recovery Tool creates an output file from the data of the APE1_ERROR_ENTITIES and APE1_ERR_ENTITIES_DETAILS tables.

In addition, the Import mode generates error files containing any records that failed to be imported back into the usage tables.

Name

The names of output files from Export mode follow this naming convention:

- error_entity_<time_stamp>_0.xml
- error_entity_<time_stamp>_1.xml
- ... error_entity_<time_stamp>_n.xml

The number of files (named *<time_stamp>_0.xml, *<time_stamp>_1.xml, *<time_stamp>_2.xml, and so on) depends on the number of transactions to which the entities in the APE1_ERROR_ENTITIES table belong and on the MaxTransactionsInFile parameter.

The names of error files from Import mode follow the same format but include the .err extension after the .xml extension.

The *error_entity* prefix can be overridden by setting the WorkingFilePrefix parameter in the ADJ1_CONF_SECTION_PARAM table to a different value.

Location and Format

- *Location* – \$ABP_ERT_ROOT/work/data/<folder name by time stamp>/
- *Format* – XML

Contents

The output file consists of a header and body. The body consists of a number of groups of rows, one group for each transaction that included failed entities.

Header

The header provides the following information:

- Start and end time of the file
- Cycle affected
- The WHERE clauses for extracting data from the APE1_ERROR_ENTITIES and APE1_ERR_ENTITIES_DETAILS tables
- File sequence number
- Indication of whether the file has been fixed (true/false)

Body

The body contains one group of rows for the entities of each erred transaction.

Each such group provides the following information:

- *Error code* – The Oracle error code.
- *System creation date* – The date and time of the entity's insertion into the APE1_ERROR_ENTITIES and APE1_ERR_ENTITIES_DETAILS tables
- *Problematic Entity* – Identifier of the particular entity within the transaction that caused the error. These are the possibilities:
 - *ACC* – Accumulator
 - *RTD* – Rated event
 - *DSP* – Dispatching record
 - *NTF* – Notification
 - *REJ* – Rejected event
 - *DUPK* – Duplicate check key
 - *RREJ* – Event that was rejected in rerate and could not be marked as deleted in the APE1_RATED_EVENT table
 - *RRTD* – Event that failed to be inserted into the APE1_RATED_EVENT table after rerating
 - *RRAC* – Accumulator that failed to be inserted into the APE1_ACCUMULATORS table after rerating

- *Transaction Number* – The group ID shared by all the entities that represent the same Persistence Writer transaction.
- *Fixed (true/false)* – An indication of whether entity correction has been completed at the transaction level. This flag is controlled manually by the support person.

ErrorEntitiesDetails	<table border="1"> <tr><td>= end_time</td><td>2012-07-15+03:00</td></tr> <tr><td>= start_time</td><td>2012-07-15+03:00</td></tr> <tr><td>= working_cycle</td><td>1</td></tr> <tr><td>= error_entities_...</td><td></td></tr> <tr><td>= error_entities_...</td><td>TRANSACTION_NUMBER=367282850000000</td></tr> <tr><td>= file_seq</td><td>0</td></tr> <tr><td>= fixed</td><td>true</td></tr> <tr><td>= xmlns</td><td>errorentities.details</td></tr> </table>	= end_time	2012-07-15+03:00	= start_time	2012-07-15+03:00	= working_cycle	1	= error_entities_...		= error_entities_...	TRANSACTION_NUMBER=367282850000000	= file_seq	0	= fixed	true	= xmlns	errorentities.details																													
= end_time	2012-07-15+03:00																																													
= start_time	2012-07-15+03:00																																													
= working_cycle	1																																													
= error_entities_...																																														
= error_entities_...	TRANSACTION_NUMBER=367282850000000																																													
= file_seq	0																																													
= fixed	true																																													
= xmlns	errorentities.details																																													
ErrorEntitiesRow	<table border="1"> <tr><td>= error_code</td><td>1400</td></tr> <tr><td>= sys_creation_date</td><td>2012-07-15T10:10:11.000+03:00</td></tr> <tr><td>= problematic_entity</td><td>RTD</td></tr> <tr><td>= transaction_number</td><td>367282850000000</td></tr> <tr><td>= fixed</td><td>true</td></tr> </table>	= error_code	1400	= sys_creation_date	2012-07-15T10:10:11.000+03:00	= problematic_entity	RTD	= transaction_number	367282850000000	= fixed	true																																			
= error_code	1400																																													
= sys_creation_date	2012-07-15T10:10:11.000+03:00																																													
= problematic_entity	RTD																																													
= transaction_number	367282850000000																																													
= fixed	true																																													
	<table border="1"> <thead> <tr><th></th><th>(o) record_details</th><th>(o) entity_details</th><th>(o) data</th><th>(o) common</th></tr> </thead> <tbody> <tr><td>1</td><td>[checkbox]</td><td>[checkbox]</td><td>[checkbox] data (84)</td><td></td></tr> <tr><td>2</td><td>[checkbox]</td><td>[checkbox]</td><td>[checkbox] data (17)</td><td>[checkbox] common (6)</td></tr> <tr><td>3</td><td>[checkbox]</td><td>[checkbox]</td><td>[checkbox] data (12)</td><td></td></tr> <tr><td>4</td><td>[checkbox]</td><td>[checkbox]</td><td>[checkbox] data (12)</td><td></td></tr> <tr><td>5</td><td>[checkbox]</td><td>[checkbox]</td><td>[checkbox] data (12)</td><td></td></tr> <tr><td>6</td><td>[checkbox]</td><td>[checkbox]</td><td>[checkbox] data (9)</td><td></td></tr> <tr><td>7</td><td>[checkbox]</td><td>[checkbox]</td><td>[checkbox] data (14)</td><td></td></tr> <tr><td>8</td><td>[checkbox]</td><td>[checkbox]</td><td>[checkbox] data (57)</td><td></td></tr> </tbody> </table>		(o) record_details	(o) entity_details	(o) data	(o) common	1	[checkbox]	[checkbox]	[checkbox] data (84)		2	[checkbox]	[checkbox]	[checkbox] data (17)	[checkbox] common (6)	3	[checkbox]	[checkbox]	[checkbox] data (12)		4	[checkbox]	[checkbox]	[checkbox] data (12)		5	[checkbox]	[checkbox]	[checkbox] data (12)		6	[checkbox]	[checkbox]	[checkbox] data (9)		7	[checkbox]	[checkbox]	[checkbox] data (14)		8	[checkbox]	[checkbox]	[checkbox] data (57)	
	(o) record_details	(o) entity_details	(o) data	(o) common																																										
1	[checkbox]	[checkbox]	[checkbox] data (84)																																											
2	[checkbox]	[checkbox]	[checkbox] data (17)	[checkbox] common (6)																																										
3	[checkbox]	[checkbox]	[checkbox] data (12)																																											
4	[checkbox]	[checkbox]	[checkbox] data (12)																																											
5	[checkbox]	[checkbox]	[checkbox] data (12)																																											
6	[checkbox]	[checkbox]	[checkbox] data (9)																																											
7	[checkbox]	[checkbox]	[checkbox] data (14)																																											
8	[checkbox]	[checkbox]	[checkbox] data (57)																																											

- *Error Entities Details Rows* – One or more rows, each containing the following information about the entity:

- *Entity Details*

entity_details	<table border="1"> <tr><td>= commonMetadata...</td><td>0</td></tr> <tr><td>= metadata_id</td><td>845490082021377</td></tr> <tr><td>= entity_version</td><td>1</td></tr> </table>	= commonMetadata...	0	= metadata_id	845490082021377	= entity_version	1
= commonMetadata...	0						
= metadata_id	845490082021377						
= entity_version	1						

- *Common Metadata ID* – The ID for metadata that is relevant to all dimensions and is located in dimension 1. Common metadata is not mandatory and depends on the implementation.
- *Metadata ID* – The ID for metadata that (unlike the common metadata described above) is specific to a particular dimension. This ID is assigned by the Implementation Compiler and is a unique key identifying the entity.
- *Entity Version* – The version of the entity.

- *Record Details* – Further information about the entity, including customer, cycle, and more.

record_details	
= f2e_record_id	1
= f2e_file_id	395662195905200
= target_id	0
= dimension_id	1
= item_id	392411
= offer_instance	0
= owner_type	S
= acc_type_id	248
= cycle_year	2008
= start_date	1970-01-01+02:00
= entity_type_id	0
= customer_id	272
= owner_id	114
= customer_seg...	65
= cycle_instace	7
= cycle_code	1
= application_id	PW
= sys_creation_d...	2012-07-15T10:10:11.000+03:00
= seq_number	3
= entity	ACC
= transaction_nu...	367282850000000

- *Data* – Entity fields that the support person can correct and then, via the Import mode of the Error Entity Recovery Tool, insert into the usage tables.

	= type	= db_field_name	= name	= value
1	C	RERATE_TYPE	Rerate type	R
2	N	N_1	Number of free events	0
3	N	NUMBER_OF_EVENTS	Number of events	1
4	N	PCN	Pay channel	263
5	N	BA	Billing arrangement	113
6	D	FIRST_EVENT_DATE	First event date	Sat Jul 05 11:00:01 IDT 2008
7	D	LAST_EVENT_DATE	Last event date	Sat Jul 05 11:00:01 IDT 2008
8	C	TAX_CHANGE_DATE	Tax change date	1960-01-01 19:50:00
9	C	BASIC_SERVICE_TYPE	Basic service type	V
10	N	N_2	Business entity	32156
11	C	X512_1	Number of events per period	24#1#0#T;1#=1;
12	C		Accumulated charge per period	32#1#0#T;1#=-1.49;
13	C	X512_3	Number of free events per period	
14	N	N_3	Number of credit events	0
15	F	F_1	Accumulated credit amount	0.00
16	C	X512_4	Accumulated duration per period	32#1#0#T;1#=-1.49;
17	C	X512_5	Accumulated free duration per period	32#1#0#T;1#=0.00;

Flow

The Error Entity Recovery Tool has three execution modes:

- Export
- Import
- Full

Export

The Error Entity Recovery Tool in Export mode can be activated from the command line (as ERT_EXPORT) or from Amdocs Monitoring & Control.

The Export mode of the Error Entity Recovery Tool (ERT_EXPORT) performs the following steps:

1. Configuration
 - a. Parses the input arguments and validates their values.
 - b. Creates a connection to the operational database using the OP_ORA_USER, OP_ORA_PASS, and OP_ORA_INS environment variables.

- c. Creates a connection to the reference database.
 - d. Loads the Metadata Repository for the entities attributes descriptor.

The entities attributes descriptor provides access to the types and the sequence of the attributes according to their metadata ID, so that they can be read from the entity buffer.
 - e. Loads the configuration parameters of the Error Entity Recovery Tool from the ADJ1_CONF_SECTION_PARAM table by process name.
 - f. Creates a connection to the Usage database.
 - g. Loads the Implementation Repository for the entity attributes.
 - h. Initializes the Data Access Object layer.
2. Data preparation
 - a. Extracts all the transaction IDs, or transaction IDs as filtered by the eewhereclause and eedwhereclause parameters
 - b. Extracts each relevant entry from the APE1_ERROR_ENTITIES table and related entries from the APE1_ERR_ENTITIES_DETAILS table
 - c. For each failed transaction, creates an entity group that consists of all the entities within the transaction that could not be written into the usage tables
 - d. Creates an ErrorEntityRow element of the output XML file for each transaction
 3. Creation of output
 - a. Merges the buffer in the APE1_ERR_ENTITIES_DETAILS table. There are three parts, each a BLOB of 2000 bytes, to be merged.
 - b. Gets the parser and descriptor from the pool, according to the metadata ID of the entity.
 - c. Parses the BLOB.
 - d. Creates an ErrorEntitiesDetailsRow element of the output XML file for each entry.
 - e. Starts a new output file whenever the number of transactions in the latest output file reaches the value of the MaxTransactionInFile configuration parameter and more transactions remain.

Making Corrections

After running the Error Entity Recovery Tool in Export mode and before running it in Import mode, the support person performs the following:

1. Corrects the exported XML file as necessary
2. In the file header and in each ErrorEntitiesDetailsRow as appropriate, changes the value of the Fixed attribute to ‘true’

The “Important Remarks” section of this chapter provides some correction scenarios.

Import

The Import mode of the Error Entity Recovery Tool can be activated from the command line (as ERT_IMPORT) or from Amdocs Monitoring & Control.

Sequence of Steps

The Error Entity Recovery Tool in Import mode (ERT_IMPORT) performs the following steps:

1. Configuration
 - a. Parses the input arguments and validates their values.
 - b. Creates a connection to the operational database using the OP_ORA_USER, OP_ORA_PASS, and OP_ORA_INS environment variables.
 - c. Creates a connection to the reference database.
 - d. Loads the Metadata Repository for the entities attributes descriptor.

The entities attributes descriptor provides access to the types and the sequence of the attributes according to their metadata ID, so that they can be read from the entity buffer.
 - e. Loads the configuration parameters of the Error Entity Recovery Tool from the ADJ1_CONF_SECTION_PARAM table by process name.
 - f. Creates a connection to the Usage database.
 - g. Loads the Implementation Repository for the entity attributes.
 - h. Loads the XML import files from the import folder according to the import key, and sorts the XML files by file name.
 - i. Initializes the Data Access Object layer.
 2. Data preparation
 - a. Checks whether the value of the Fixed attribute is ‘true’ – first in the header, and if so, in the particular transaction.
 - b. For a ‘true’ transaction, gets an entity from the APE1_ERR_ENTITIES_DETAILS table and checks whether the PROCESSING_STATUS for that entity in the table is RI or ER (see “Processing Status” in this chapter).
- If it is, step c is performed. Otherwise, the next entity is tried.

- c. Prior to importing the corrected entity into the relevant usage table, performs the following validations according to the entity type.

Entity Type	Validation
RTD	<ul style="list-style-type: none"> ▪ <i>Duplicate events</i> – The duplicate check key is used, according to the relevant event type, to determine whether the event is a duplicate. Duplicate events are not inserted back into the usage tables, and they are marked in the APE1_ERR_ENTITIES_DETAILS table as duplicates (see “Processing Status” in this chapter). ▪ <i>Rerateable events</i> – If the event is rerateable, it is inserted into the database, and the subscriber is marked for rerate.
ACC	If the value of the SYS_UPDATE_DATE field for the accumulator in the APE1_ACCUMULATORS table is later than the value of the SYS_CREATION_DATE field of the accumulator in the APE1_ERR_ENTITIES_DETAILS table, and if the accumulator is rerateable, the subscriber is marked for rerate, and the process continues to the next entry without updating the accumulator. Rerating fixes the mismatch between the accumulators in memory and in the database that was created as a result of the original failure to write the accumulator into the database.
DUPK	If a duplicate check key already exists in the APE1_EVENT_DUP_KEYS table, it is not inserted again, and it is marked in the APE1_ERR_ENTITIES_DETAILS table as duplicate.

3. Data import

- a. Gets the parser and descriptor from the pool, according to the metadata ID of the entity in the XML file
- b. Create a “prepared statement” query for the entity type, mapping the entity into the database according to descriptor fields and core fields
- c. Commits each entity to the relevant table in the Usage database and updates its status in the APE1_ERR_ENTITIES_DETAILS table (see “Processing Status” in this chapter)
- d. Creates a file with all the entities that failed to be imported into the relevant usage tables

Processing Status

The Error Entity Recovery Tool in Import mode accesses and updates the processing status of each entity in the APE1_ERR_ENTITIES_DETAILS table. The processing statuses are as follows.

Processing Status	Meaning
RI	<i>Ready for Investigation</i> – The Persistence Writer has inserted the erred entity into the APE1_ERROR_ENTITIES and APE1_ERR_ENTITIES_DETAILS tables.
DU	<i>Duplicate</i> – The event already exists in the APE1_RATED_EVENT table, or the duplicate check key already exists in the APE1_EVENT_DUP_KEYS table. In this case, the Error Entity Recovery Tool in Import mode does not insert the event or duplicate check key into the usage tables and marks it in the APE1_ERR_ENTITIES_DETAILS as duplicate.
UD	<i>Up to Date</i> – The value of the SYS_UPDATE_DATE field of the accumulator in the APE1_ACCUMULATORS table is later than that of the entry inserted into the APE1_ERROR_ENTITIES table. In this case, the Error Entity Recovery Tool does not insert the older entity into the usage tables and marks it in the APE1_ERR_ENTITIES_DETAILS table as ‘UD’ (meaning that an up-to-date entity already exists in the usage tables).
ER	<i>Error</i> – The Error Entity Recovery Tool in Import mode failed to write the entity into the usage tables.
CO	<i>Processed</i> – The Error Entity Recovery Tool in Import mode successfully wrote the entity into the usage tables.
EX	<i>Expired</i> – The creation date of the entry is earlier than allowed by the MaxTrxTimeLimitationInDays parameter, so the entity is marked as Expired.

Full

The Full mode of the Error Entity Recovery Tool (ERT_FULL) resembles a concatenation of Export mode and Import mode except that it imports all the data without checking whether the Fixed attribute is ‘true’ or ‘false’.

It is useful for importing large quantities of data that were left unwritten because of some technical problem with the Usage database that has since been solved.

The Error Entity Recovery Tool in Full mode from can be activated from the command line (as ERT_FULL) or from Amdocs Monitoring & Control.

Environment Variables

The following environment variables affect the tool.

Name	Description	Default Value
ABP_ERT_ROOT	The name of the directory containing the XML files, outputs, and logs of the tool	\$HOME/var/m3g/ert

Name	Description	Default Value
OP_ORA_USER	The user name for the Operational database	N/A
OP_ORA_PASS	The password to the Operational database	N/A
OP_ORA_INST	The instance of the Operational database	N/A
NTF_DAY_PARTITION_NUM	The number of partitions in the Notification tables	6

Parameters

The following parameters must be set for the tool.

Export

These are the arguments for the Export mode.

Argument	Description	Mandatory?	Example(s)
processname	The process name as defined in the ADJ1_CONF_HIERARCHY table	Mandatory	<ul style="list-style-type: none"> ▪ ERT_EXPORT ▪ ERT_EXPORT941
cyclecode	The cycle code of the transaction	Mandatory	1
eewhereclause	The WHERE clause to use in extracting from the APE1_ERROR_ENTITIES table	Optional	TRANSACTION_NUMBER =123456789
eedwhereclause	The WHERE clause to use in extracting from the APE1_ERR_ENTITIES_DETAILS table	Optional	ENTITY like ACC%

Import

These are the arguments for the Import mode.

Argument	Description	Mandatory?	Example(s)
processname	The process name as defined in the ADJ1_CONF_HIERARCHY table	Mandatory	ERT_IMPORT ERT_IMPORT942
cyclecode	The cycle code of the transaction	Mandatory	1
importkey	The folder name under \$ABP_ERT_ROOT/work/data in which the corrected XML file is located	Mandatory	1344253516178

Full

These are the arguments for the Full mode.

Argument	Description	Mandatory?	Example(s)
processname	The process name as defined in the ADJ1_CONF_HIERARCHY table	Mandatory	ERT_FULL ERT_FULL943
cyclecode	The cycle code of the transaction	Mandatory	1
eewhereclause	The WHERE clause to use in extracting from the APE1_ERROR_ENTITIES table	Optional	TRANSACTION_NUMBER =123456789
eedwhereclause	The WHERE clause to use in extracting from the APE1_ERR_ENTITIES_DETAILS table	Optional	ENTITY like ACC%

Configuration Parameters

The configuration parameters of the Error Entity Recovery Tool are defined in the database.

The following table shows the performance tuning (configuration) parameters of the Error Entity Recovery Tool. These parameters are defined in the ADJ1_CONF_SECTION_PARAM table using the Turbo Charging Configurator (see *Turbo Charging Configurator User Guide*).

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ERT	ert.config	AppId	ERT	Application ID.
ERT	ert.config	ByteOrder	LITTLE_ENDIAN	Order of the bytes in the buffer <ul style="list-style-type: none"> ■ Windows or Linux on x86 – LITTLE_ENDIAN ■ Unix – BIG_ENDIAN
ERT	ert.config	MaxBufferSize	2000	Maximum buffer size for an entity in the APE1_ERR_ENTITIES_DETAILS table.
ERT	ert.config	MaxTrxTimeLimitationInDays	30	Maximum number of days a transaction can remain in the APE1_ERROR_ENTITIES table before expiring.
ERT	ert.config	NumOfThreads	10	Number of threads. The modes of the Error Entity Recovery Tool support multi-threading as follows: <ul style="list-style-type: none"> ■ <i>Export</i> – Each thread works on a bulk of transactions. ■ <i>Import</i> – Each thread works on an input file. ■ <i>Full</i> – Multi-threaded export is followed by multi-threaded import.
ERT	ert.config	ODBCDriver	oracle.jdbc.driver.OracleDriver	Driver for Oracle.
ERT	ert.config	RacMode	true	Whether the Oracle RAC mode is in effect.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ERT	ert.config	RerateMode	Both	Rerateability mode for the entities: <ul style="list-style-type: none"> ■ <i>Rerateable</i> – All entities are automatically treated as rerateable. ■ <i>NotRerateable</i> – All entities are automatically treated as not rerateable. ■ <i>Both</i> – The tool checks whether each entity is rerateable or not.
ERT	ert.config	WorkingFilePrefix	error_entity	Output file prefix.
ERT_EXPORT	ert.config	MaxTransactions	250	Maximum number of transactions that can be exported by a single run.
ERT_EXPORT	ert.config	MaxTransactionsInFile	250	Maximum number of transactions that a file can contain before a new file is opened.
ERT_EXPORT	ert.config	Mode	Export	Basic mode for the process name specified in the PARAM_CLASS field. Values are: <ul style="list-style-type: none"> ■ Export ■ Import ■ Full
ERT_EXPORT	ert.config	OutputExportFolder	\${ABP_ERT_ROOT}/work/data/	Location of output files.
ERT_FULL	ert.config	Mode	Full	Basic mode for the process name specified in the PARAM_CLASS field. Values are: <ul style="list-style-type: none"> ■ Export ■ Import ■ Full
ERT_IMPORT	ert.config	IOTMode	false	Whether an index-organized table (IOT) is being used for duplicate check. In this case, duplicate check is performed against the APE1_EVENT_DUP_KEYS table and not against the APE1_RATED_EVENT table.
ERT_IMPORT	ert.config	InputImportFolder	\${ABP_ERT_ROOT}/work/data/	Location of input files.

PARAM_CLASS	SECTION_NAME	PARAM_NAME	PARAM_VALUE (Default)	Description
ERT_IMPORT	ert.config	Mode	Import	Basic mode for the process name specified in the PARAM_CLASS field. Values are: <ul style="list-style-type: none"> ■ Export ■ Import ■ Full
ERT_IMPORT	ert.config.rerate	MarkType	4	Rerate Parameter: Rerate level (for example, 4 for subscriber level).
ERT_IMPORT	ert.config.rerate	RerateSource	ERT_RECYCLING	Application ID of the application that marks the subscriber or customer for rerate.

Database Connections

The tool connects to the following tables during processing.

Database Type	Table Name	Connection Type (Update, Insert, and so on)	Comments
Usage	APE1_ACCUMULATORS	Update, Insert	Update or insert accumulators
Usage	APE1_ERR_ENTITIES_DETAILS	Update	Update PROCESSING_STATUS
Usage	APE1_ERROR_ENTITIES	Read	Read transaction attributes
Usage	APE1_EVENT_DUP_KEYS	Insert	Insert duplicate check keys
Usage	APE1_NOTIFICATIONS	Insert	Insert notifications
Usage	APE1_RATED_EVENT	Insert	Insert new rated events
Usage	APE1_REJECTED_EVENT	Insert	Insert new rejected events
Usage	APE1_SUBSCRIBER_RERATE	Insert	Mark subscribers or customers for rerate
Usage	APR1_DISPATCHING_RECORDS	Insert	Insert dispatching records
Reference	ADJ1_CONF_HIERARCHY	Read	Read configuration
Reference	ADJ1_CONF_SECTION_PARAM	Read	Read configuration
Reference	ADJ1_METADATA	Read	Read Implementation Compiler metadata
Reference	ADJ1_METADATA_ATTRIBUTES	Read	Read Implementation Compiler metadata
Reference	APE1_XML_DISTRIB	Read	Read Implementation Repository

Screen Messages

The following messages may be sent to the operator while the tool is running.

Error Messages

The following error messages may be sent to the operator.

Configuration

The following error messages are related to the configuration of the tool.

Short Description	Long Description	Troubleshooting
Mandatory Arguments are missing	Mandatory argument [process_name] is missing	Check the command line. A mandatory argument is missing.
Argument with no value	Some parameters are missing, please check [key, value]	Check the command line. The command line includes a parameter name with no value or vice versa.

Export

The following error messages may be sent when the tool is running in Export mode.

Short Description	Long Description	Troubleshooting
(ADJ1-042002) Error {0}, Serializer, Message:{1}	Error to read Buffer and create xml, this error occur in export mode	Possible reasons for the error are: <ul style="list-style-type: none"> ■ The buffer is corrupted. In this case, consult the Event Server team. ■ The metadata in the database is corrupted. In this case, consult the implementation team.
(ADJ1-042003) No error entities on cycle {0}	Error entities table is empty for cycle	There are no entities to export for the specific cycle. Check the cycle and run the tool on the correct cycle.
(ADJ1-042001) Error fetch error entities details, please check EEDWhereClause parameter	Error in execute query fetch from error entities maybe EEDWhereClause  Note: <i>The eedwhereclause parameter corrupted the query. The SQL is not valid.</i>	Data could not be fetched from the APE1_ERR_ENTITIES_DETAIL S table. Check that the WHERE clause in the eedwhereclause parameter is written correctly.

Import

The following error messages may be sent when the tool is running in Import mode.

Short Description	Long Description	Troubleshooting
(ADJ1-040010) A problem to marshller file {0}	Error to open import file	There was a problem in opening the import file. Check whether the file is a valid XML file and not corrupted.
(ADJ1-043001) Notification descriptor name {0} does not exist	Descriptor does not exist for notification metadataId ,maybe mismatch between entity and Implementation Compiler metadata	There is no notification descriptor. Consult the implementation team.
(ADJ1-043004) Import file {0} marked with fixed=false, ignore file	Import file not marked as fixed ,the process skip file	The file is not marked as fixed. Check whether the Fixed status of all the transactions is still ‘false’. If for one or more transactions it is ‘true’, change the status of the file to Fixed.

Short Description	Long Description	Troubleshooting
(ADJ1-043005) Error in Import, Date limitation passed trx date {0} Days limitation {1}	The entity in database old the max limitation property (MaxTrxTimeLimitationInDays parameter in adj1_conf_section_param table ,Default 30 days)	The file has aged past its expiration date. If you want files to survive longer before expiration, consult the person in charge of the business parameters.
Row does not exist in APE1_ERR_ENTITIES_DETAILS , continue to next row	Entry that we are trying to import does not exist in APE1_ERR_ENTITIES_DETAILS table	The entry was deleted from the APE1_ERR_ENTITIES_DETAILS table. Check whether the deletion is intentional or whether there is a problem of missing data.

Success Messages

The following success message may be sent to the operator in Export mode.

Short Description	Long Description
Export Error Entities Details to file: <filename> <i>Example:</i> Export Error Entities Details to file: var/m3g/projs/ert/work/dataerror_entity_1342533522470_0.xml	Close the export file and write it to the file system.

Troubleshooting

If the job does not succeed, the reason may be a database problem. These are possibilities:

- Locked tables. In this case, consult a database administrator.
- Missing Implementation Repository record in the PC1_XML_DISTRIB table. In this case, consult the implementation team.

Recovery Instructions

Rerun the job after fixing the problems that caused it to fail.

Important Remarks

Records may be inserted into the Error Entities tables because of various problems:

- A unique constraint violated for an event or an IOT table – for example, a duplicate event record listed in the IOT duplicate check table
- An overly long column
 - Field arrived from the network too long and no validation from the Event Server or from the Implementation Compiler failed (not likely to happen).
 - Field was populated by implementation and is too long.

- A complex attribute was populated with many entries, and the size defined for an entry in the Rating Logic Configurator was not correct.

There are other possible reasons as well. Each failure should be investigated, and then appropriate action should be taken. Actions may vary from one situation to the other:

- Correcting the event or accumulator
- Correcting the network element and the event
- Correcting the implementation
- Enlarging the column size in the database (when user-assigned)

The following remarks apply if the corresponding problems have been noticed and confirmed.

Corrupted Accumulators in Amdocs In-Memory Object Storage (AIMOS)

In an accumulator successfully committed into Amdocs In-Memory Object Storage (AIMOS), it is possible for an attribute to be too long for insertion into the database. In such a situation, all subsequent events will have their accumulator data inserted into the Error Entities tables, rather than into the Accumulators table, until the problem is fixed.

- If the problem is fixed by enlarging the column in the database, further events succeed with no additional effort.
- If the records are fixed by the Error Entity Recovery Tool and inserted into the database in their corrected form, the underlying problem – the accumulator in AIMOS – remains. In this case:
 - a. Use the Reconciliation tool to request a refresh of the accumulator from the database.
 - b. Because an additional event may, in the meantime, have had an impact on the memory, use the Reconciliation tool a second time, to request another refresh of the accumulator from the database.

Unrecognized Duplicate Event

After an event has been inserted into the Error Entities tables, it is possible that the same event may enter the system a second time. In this case, the event is not recognized as a duplicate (because the real check is against the APE1_RATED_EVENT table, where the record does not exist), and it is processed again.

The Error Entity Recovery Tool recognizes such a situation and does not insert the event a second time into the Rated Event table. In the Error Entities tables, the tool marks the event as duplicate. However, if the event is not rerateable, it affects the accumulators and the balance twice. Then it is the operator's responsibility to correct the accumulators.

If the same event arrives at the system within the duplicate check window in memory after the original event has been corrected and inserted into the APE1_RATED_EVENT table, it is recognized as a duplicate because the suspected key of the original event is in AIMOS. If a duplicate event arrives outside the duplicate check window, it is still recognized as one when the check is performed against the database.

68 High Availability

The Availability Manager monitors all Turbo Charging processes (jobs and daemons). It is responsible for:

- Starting the daemons through the Availability Manager plug-in to Amdocs Monitoring & Control
- Monitoring the health of each process
- Moving processes (and their segments, in the case of the Event Server) to another host at the time of failover
- Sending admin messages to processes

This chapter explains how to enable the Availability Manager for different types of processes, lists the high availability type of each Turbo Charging process, and describes how to configure the system for high availability.

For more information on running the Availability Manager, see *Amdocs Billing Availability Manager Run Book*.

Enabling the Availability Manager

To enable process activation through the Availability Manager:

- For C++ processes, define the following entry in the APR1_CONF_SECTION_PARAM table:

```
('APR', 'config', 'AVM_ENABLED', SYSDATE, SYSDATE),
    NULL, 'FMAPR1', 'v750', 0, 'Y',
    0, 'FALSE');
```
- For Java processes, define the following entry in the ADJ1_CONF_SECTION_PARAM table:

```
('FND', 'fnd.avm', 'useAvm', SYSDATE, SYSDATE),
    NULL, 'FMADJ1', 'v750', 0, 'Y',
    1, 'FALSE');
```

High Availability Types of Turbo Charging Processes

The following high availability types are used by Turbo Charging processes:

- *Bundle* – This is a bundle of one active and one shadow processes.
 - Upon process failure, the shadow is activated.
 - Upon host failure, the replicated process group is activated.
- *One up – one active* – The back-up process is down and is only started if the active process fails.
If a process fails, a configurable number of attempts to restart it are made, and then processing is switched to a back-up machine (after making sure that the process is down on the primary machine).
- *Many up – many active* – All process within the process group are active (all serving the same population), and a back-up process group serves as a standby.
 - Upon process failure, it is restarted by the External Watchdog and functions as part of the group.
 - Upon process group or host failure, the client receives a command to redirect messages to the standby process group.
- *Many up – one active* – The back-up process is up but not activated. When the active process fails, the back-up becomes activated.

- *Monitored only* – The Availability Manager monitors and controls the process but does not ensure its high availability. An external cluster management system, such as Veritas, is responsible for executing the failover of such processes in the case of host failure and restarting them in the case of process failure. Any process except for the Event Server can be defined with this high availability type. However, it must be used for File2E (or any other process that requires access to files) if a shared file system is not used.

The following table shows the type of high availability used by each Turbo Charging process.

Process Type	High Availability Type	Blocked on Change Route Command
Bill on Demand Extract	One up – one active	N/A
Compress Usage	Monitored only	N
Copy Cycle Usage	Monitored only	N
Copy Cycle Usage Clean-up	Monitored only	N
Copy Rolling Accumulators	Monitored only	N
Cycle Maintenance	Monitored only	Y
Cycle Maintenance EOD	Monitored only	Y
Cycle Memory Clean-up	Monitored only	N
Cycle State	Monitored only	N
DB2E	One up – one active	Y
DB2E Error	One up – one active	Y
Dispatcher	Many up – many active	N
Dispatcher for End of Cycle	Many up – many active	N
ELA Collector	One up – one active	N
Error Entity Recovery tool (Import, Export, Full)	Monitored only	N/A
Event Server	Bundle	Y
Extract Clean-up	Monitored only	N
File2E	<ul style="list-style-type: none"> ■ Many up – many active (if a shared file system is used) ■ Monitored only (if there is no shared file system) 	Y
GAT2E	One up – one active	Y
Move Usage	Monitored only	N
Notification	One up – one active	N
Outcollect Prepare	Many up – many active	Y
Read-Only Rater Client	Monitored only	N
Rejected Event Recycler	One up – one active	N
Rerate Error Report	Monitored only	Y
Rerate Population Report	Monitored only	Y

Process Type	High Availability Type	Blocked on Change Route Command
Rerate Prepare	Many up – many active	Y
Rerate Prepare for End of Cycle	Monitored only	Y
Rerate Prepare for End of Cycle Finish Notification	Monitored only	Y
Roll Accumulators	Monitored only	Y
Truncate Usage	Monitored only	N
Update Handler in CUG mode	Monitored only	N
Update Handler in Full mode	Monitored only	N
Update Handler in Incremental mode	One up – one active	N
Update Handler in Mark for Rerate mode	One up – one active	N
Update Handler in Mini-Full mode	Monitored only	N
Update Handler in Synchronous mode	Many up – many active	N
Update Handler in Reconciliation mode	One up – one active	N
Usage Extracts	Monitored only	N
Usage Query Server	Many up – many active	N

To change the high availability type of a process:

1. In the GN1_SYS_PROC_INST_GRP_CFG, change the value of the PROCESS_GROUP_TYPE_ID field for the functional unit to the desired value. (The value must be taken from the PROC_GROUP_TYPE_ID column of the AVM1_PROC_GRP_TYPE_REF table.)

Example:

To change the high availability type of File2E to ‘monitored only’, change the value of the PROCESS_GROUP_TYPE_ID field for F2E_FU to ‘10’. This value corresponds to the MONITORED_ONLY_FU_FAILOVER_TYPE failover schema defined in the AVM1_PROC_GRP_TYPE_REF table.

2. Change the value of the HA_POLICY field in the GN1_SYS_PROC_TYPE_CFG table. Valid values are:
 - ONE_UP_ONE_ACTIVE
 - MANY_UP_MANY_ACTIVE
 - MANY_UP_ONE_ACTIVE
 - BUNDLE
 - MONITORED_ONLY

Configuring High Availability

This section explains how to configure the Turbo Charging processes as highly available.

AVM1_CONFIG

For the Availability Manager to work with Turbo Charging processes, the following entries must be defined in the Availability Manager Configuration table:

- (1, 'AVMSRV_BASE', 'FOVR', 'AVM_MODE', 'FULL_SEGMENT_TOPOLOGY_MAP', SYSDATE, NULL, NULL, 'DIAVM1', 'v750', NULL, 'FALSE', NULL);
- (1, 'BASE', 'FOVR', 'ENABLE_ONE_UP_ONE_ACTIVE_EWD_FAILOVER_MECHANISM', Y, SYSDATE, NULL, NULL, 'DIAVM1', 'v750', NULL, 'FALSE', NULL);

GN1_SYS_SITES_NETWRKS_CFG

The System Sites Networks configuration table details the various networks on each site. There are six types of networks (for more information, see *Amdocs Billing Availability Manager Data Model*). Of those, at least one network must be configured for each of the following types:

- *Admin network* – For admin and Availability Manager messages
- *Internal network* – For connections among various Event Servers
- *External network* – For connections between Event Servers and other processes

GN1_SYS_IP_ADDRESSES_CFG

The System IP Addresses configuration table provides details about the IP addresses used (for more information, see *Amdocs Billing Availability Manager Data Model*).

Each IP address can be associated with several networks. The table must contain at least one IP address associated with the Admin network type.

GN1_SYS_PROCESS_TYPE_CFG

The System Process Types configuration table provides details of all system process types (for more information, see *Amdocs Billing Availability Manager Data Model*).

The following process types must be configured in this table:

- *All high availability system process types:*
 - Quorum
 - AVM 1
 - AVM 2
 - Agent
- *Bundle processes* – As critical
- *Single instances in a process group* – As critical

GN1_SYS_PROC_INSTANCE_CFG

The System Processes configuration table provides details about the process configurations (for more information, see *Amdocs Billing Availability Manager Data Model*).

All high availability system process instances must be configured in this table:

- *Quorum, AVM 1, AVM 2.*
- *Agent* – One process instance per host.
- ‘*One up – one active*’, ‘*many up – many active*’, and ‘*monitored only*’ types – Each process instance in one process group.
- *Bundle type* – Each process group contains three process instances:
 - An Event Server defined as *active* (priority within the group = 0)
 - An Event Server defined as *shadow* (priority within the group = 1)
 - A Rerate Prepare (RRP) process (priority within the group = 99)

GN1_SYS_CONN_PAIRS_CFG

The System Connection Pairs configuration table describes all of the connections among the processes in the system by defining the client-server relationships (for more information, see *Amdocs Billing Availability Manager Data Model*).

The Turbo Charging system considers the Event Server as the server and all other processes as clients. Therefore, this table must remain empty.

GN1_SYS_SEG_PROC_CFG

The System Segment Processes configuration table provides details about the various segment groups in the system and the process groups or the processes that handle them (for more information, see *Amdocs Billing Availability Manager Data Model*).

The following are valid process roles for segment groups in this table:

- *0* – Master role
- *1* – Replication role
- *10 and up* – Client role

Back-ups must be configured for the following:

- ‘*One up – one active*’ daemon processes (jobs must be configured *without* a back-up)
- Bundle processes with responsibility for a customer or resource segment group

The following are valid segment group types:

- *C* – Customer segment group
- *R* – Resource segment group
- *D* – Dummy segment group

- *A* – Availability Manager segment group
- *S* – Dispatcher segment group

The configuration for each of these segment groups is described in subsequent sections.

Customer Segment Groups

For each customer segment group, six entries must be configured in this table: two for the Event Server, two for DB2E, and two for DB2E_ERR:

- Event Server:
 - The master process group (Event Server) for this segment group (backup_order = 0)
 - The replication process group (Event Server) for this segment group (backup_order = 1)
- DB2E (with the client role):
 - The primary DB2E process group responsible for this segment group (backup_order = 0).
 - Another process group on a different host (backup_order = 1). This process group will be used when the Availability Manager performs a failover for the primary DB2E process group.
 - Groups of special cycle codes (for example, rejected events' cycle code, outcollect cycle code, or dropped events' cycle code) must not be defined at all for DB2E.
- DB2E_ERR (with the client role):
 - The primary DB2E_ERR process group responsible for this segment group (backup_order = 0).
 - Another process group on a different host (backup_order = 1). This process group will be used when the Availability Manager performs a failover for the primary DB2E_ERR process group.

For ‘one up – one active’ processes that handle customer groups (for example, DB2E and DB2E_ERR), the back-up process *must be the same* for all segment groups that belong to same process.

Resource Segment Groups

For each resource segment group, six entries must be configured in this table: two for the Event Server, two for DB2E, and two for DB2E_ERR:

- Event Server:
 - The master process group (Event Server) for this segment group (backup_order = 0)
 - The replication process group (Event Server) for this segment group (backup_order = 1)

- DB2E (with the client role):
 - The primary DB2E process group responsible for this segment group (backup_order = 0).
 - Another process group on a different host (backup_order = 1). This process group will be used when the Availability Manager performs a failover for the primary DB2E process group.
- DB2E_ERR (with the client role):
 - The primary DB2E_ERR process group responsible for this segment group (backup_order = 0).
 - Another process group on a different host (backup_order = 1). This process group will be used when the Availability Manager performs a failover for the primary DB2E_ERR process group.

For ‘one up – one active’ processes that handle resource groups (for example, DB2E and DB2E_ERR), the back-up process *must be the same* for all segment groups that belong to same process.

Dummy Segment Groups

Dummy segment groups must be configured for the following processes:

- Servers with no responsibility for segment groups, for example, Event Servers functioning as the Event Interface Module (CR) only, End-of-Cycle servers, or read-only Event Servers
- Note:  *In an integrated environment with Amdocs Service Platform, Event Servers functioning as the Event Interface Module must be configured with master responsibility for a dummy segment group. The Availability Manager uses the dummy segment group to identify the Event Servers that function as the Event Interface Module and sends this information to Amdocs Service Platform.*
- ‘One up – one active’ processes with no responsibility for segment groups that require back-up

Each such process group must have one dummy segment group in this table, with backup_order = 0.

For processes that require back-up, such segment groups also require an entry under another process group (that contains this process type), with backup_order=1. The second process group must be configured on another host. It is used when the Availability Manager performs a failover for the first process group. Both the process and its back-up must be configured with the client role.

Availability Manager Segment Groups

The table must contain one entry for each instance of an Availability Manager system process (Quorum, AVM master, AVM slave, and AVM agents).

Dispatcher Segment Groups

For information, see the “Configuring Relationships between Event Servers and Dispatcher Process” section in the “ADJ1DISPSRV – Dispatcher” chapter.

Handling Admin Commands

Each Turbo Charging process can receive admin commands (for example, Graceful Shutdown, Suspend, or Resume) from two sources:

- A dedicated UDP admin port, configured under the relevant instance in the GN1_SYS_IN_CONNS_CFG table with PORT_ROLE ‘A’
- Amdocs Monitoring & Control, through the CallBack APIs of Availability Manager

To send an admin command through a UDP port, use the ADJ1_Send_Admin_Command_Sh script in the following way:

**ADJ1_Send_Admin_Command_Sh <Target machine name or IP>
\${PROCESS_TYPE} \${PROCESS_INSTANCE_NUMBER} <Command name>
<Command parameters>**

Example:

```
ADJ1_Send_Admin_Command_Sh hpx418 ES 100
REMOVE_MEMORY_COMMAND --cycleInstance "5" --cycleCode "2" --cycleYear
"2008"
```

Command parameters must follow these standards:

- The value of each parameter must be enclosed in double quotation marks (“”).
- Each parameter must be preceded with a double hyphen (--). It must begin with a lowercase letter (a capital letter may follow).

Appendix A Reconciliation Requests

This appendix contains examples of SQL script for creating reconciliation (compare and update) requests directly in the Oracle database.

The following rules apply:

- *REQUEST_ID* – Each request must have a different value.
- *EXECUTION_ID* – Each instance of the Reconciliation tool gets all messages with the input execution ID. The execution ID that a particular instance of the Reconciliation tool handles is defined by the -executionId <*Execution ID*> parameter in the command line.
- *EFFECTIVE_DATE* – If this field is left empty, the request is effective immediately; otherwise, it becomes effective only when this date arrives.

Sample Comparison Requests

This section contains sample SQL scripts for inserting comparison requests into the database.

Sample Resource Comparison Request

Following is a sample SQL script for a resource comparison request:

```
Insert into ADJ1_REQUESTS
(REQUEST_ID, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID, APPLICATION_ID,
DL_SERVICE_CODE, DL_UPDATE_STAMP, EXECUTION_ID, REQUEST_NAME, EFFECTIVE_DATE,
STATUS, ERROR_CODE, ERROR_MSG, DAY_IN_MONTH, REQUEST_CREATOR)
Values
(4, TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), TO_DATE('06/06/2010 16:54:48',
'MM/DD/YYYY HH24:MI:SS'), 7, 'tt', 'rr', 1, 22, 'CMP_RESOURCE', TO_DATE('03/14/2010
00:00:00', 'MM/DD/YYYY HH24:MI:SS'), 'RD', NULL, NULL, 2, NULL);
```

Resource requests have two input parameters: Resource Value and Resource Type.
Therefore, the corresponding rows must be inserted into the
ADJ1_REQUESTS_PARAMS table, as follows:

```
Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(4, 'RESOURCE_VALUE', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL, NULL, NULL, NULL, '217227',
NULL, NULL);
Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(4, 'RESOURCE_TYPE', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL, NULL, NULL, NULL, NULL, 2,
NULL);
```

Sample Subscriber Comparison Request

Following is a sample SQL script for a subscriber comparison request:

```
Insert into ADJ1_REQUESTS
(REQUEST_ID, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID, APPLICATION_ID,
DL_SERVICE_CODE, DL_UPDATE_STAMP, EXECUTION_ID, REQUEST_NAME, EFFECTIVE_DATE,
STATUS, ERROR_CODE, ERROR_MSG, DAY_IN_MONTH, REQUEST_CREATOR)
Values
(2, TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), TO_DATE('06/13/2010 16:23:11',
'MM/DD/YYYY HH24:MI:SS'), 7, 'tt', 'rr', 1, 22, 'CMP_SUBSCR', TO_DATE('03/14/2010 00:00:00',
'MM/DD/YYYY HH24:MI:SS'), 'RD', NULL, NULL, 2, NULL);
```

Subscriber requests have three input parameters: Subscriber ID, Cycle Code, and Customer ID. Therefore, the corresponding rows must be inserted into the ADJ1_REQUESTS_PARAMS table, as follows:

```

Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(2, 'SUBSCRIBER_ID', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL, NULL, NULL, NULL, 598,
NULL);

Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(2, 'CYCLE_CODE', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 1, NULL);

Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(2, 'CUSTOMER_ID', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 4, NULL);

```

Sample Accumulator Comparison Request for a Specific Subscriber

Following is a sample SQL script for an accumulator comparison request (compare all accumulators for a specific subscriber):



Note: This request compares the accumulators of the specified subscriber and the accumulators at the customer level.

```

Insert into ADJ1_REQUESTS
(REQUEST_ID, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID, APPLICATION_ID,
DL_SERVICE_CODE, DL_UPDATE_STAMP, EXECUTION_ID, REQUEST_NAME, EFFECTIVE_DATE,
STATUS, ERROR_CODE, ERROR_MSG, DAY_IN_MONTH, REQUEST_CREATOR)
Values
(10, TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), TO_DATE('05/26/2010 13:00:50',
'MM/DD/YYYY HH24:MI:SS'), 7, 'tt', 'rr', 1, 22, 'CMP_SUBSCR_ACCUMS', TO_DATE('03/14/2010
00:00:00', 'MM/DD/YYYY HH24:MI:SS'), 'RD', NULL, NULL, 2, NULL);

```

Accumulator requests have the following mandatory input parameters: Owner ID, Cycle Year, Cycle Instance, Cycle Code, and Customer ID. Therefore, the corresponding rows must be inserted into the ADJ1_REQUESTS_PARAMS table, as follows:

```

Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(10, 'OWNER_ID', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), TO_DATE('03/15/2010
00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL, NULL, NULL, NULL, 4, NULL);

Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(10, 'CYCLE_YEAR', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 2008, NULL);

Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(10, 'CYCLE_INSTANCE', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 12, NULL);

Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(10, 'CYCLE_CODE', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 1, NULL);

Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(10, 'CUSTOMER_ID', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 4, NULL);

```

Sample Accumulator Comparison Request for a Specific Customer

Following is a sample SQL script for an accumulator comparison request (compare all accumulators for a specific customer):

```
Insert into ADJ1_REQUESTS
(REQUEST_ID, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID, APPLICATION_ID,
DL_SERVICE_CODE, DL_UPDATE_STAMP, EXECUTION_ID, REQUEST_NAME, EFFECTIVE_DATE,
STATUS, ERROR_CODE, ERROR_MSG, DAY_IN_MONTH, REQUEST_CREATOR)
Values
(7, TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), TO_DATE('06/21/2010 08:20:13',
'MM/DD/YYYY HH24:MI:SS'), 7, 'tt', 'rr', 1, 22, 'CMP_CUST_ACCUMS', TO_DATE('03/14/2010
00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
'CO', NULL, NULL, 2, NULL);
```

Accumulator requests have two mandatory input parameters: Cycle Code and Customer ID. Therefore, the corresponding rows must be inserted into the ADJ1_REQUESTS_PARAMS table, as follows:

```
Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(7, 'CYCLE_CODE', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 1,
NULL);
Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(7, 'CUSTOMER_ID', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 4,
NULL);
```

Sample Cycle History Comparison Request

Following is a sample SQL script for a cycle history comparison request:

```
Insert into ADJ1_REQUESTS
(REQUEST_ID, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID, APPLICATION_ID,
DL_SERVICE_CODE, DL_UPDATE_STAMP, EXECUTION_ID, REQUEST_NAME, EFFECTIVE_DATE,
STATUS, ERROR_CODE, ERROR_MSG, DAY_IN_MONTH, REQUEST_CREATOR)
Values
(6, TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), TO_DATE('05/27/2010 13:14:34',
'MM/DD/YYYY HH24:MI:SS'), 7, 'tt', 'rr', 1, 22, 'CMP_CUST_CYC_HIST', TO_DATE('03/14/2010
00:00:00', 'MM/DD/YYYY HH24:MI:SS'), 'RD', NULL, NULL, 2, NULL);
```

Cycle history requests have two input parameters: Cycle Code and Customer ID. Therefore, the corresponding rows must be inserted into the ADJ1_REQUESTS_PARAMS table, as follows:

```

Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
 APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
 DATE_VALUE)
Values
(6, 'CYCLE_CODE', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
 TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
 NULL, NULL, NULL, 1, NULL);
Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
 APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
 DATE_VALUE)
Values
(6, 'CUSTOMER_ID', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
 TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
 NULL, NULL, NULL, 342, NULL);

```

Sample Removal Requests

This section contains a sample SQL script for removing accumulators from AIMOS.

Sample Remove Compared Accumulator from Memory Request

Following is a sample SQL script for removing an accumulator from the shared memory.



Note: This request removes an accumulator whose status as the result of a previous comparison request is DIFFER.

```

Insert into ADJ1_REQUESTS
REQUEST_ID, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID, APPLICATION_ID, DL_SERVICE_CODE,
DL_UPDATE_STAMP, EXECUTION_ID, REQUEST_NAME, EFFECTIVE_DATE, STATUS, DAY_IN_MONTH,
REQUEST_CREATOR)
Values
(3, TO_DATE('01/18/2013 12:52:30', 'MM/DD/YYYY HH24:MI:SS'), TO_DATE('01/18/2013 16:07:51',
'MM/DD/YYYY HH24:MI:SS'), 0, 'RCNUI ',
've750 ', 0, 22, 'RMV_CMP_ACCUMS_MEM', TO_DATE('01/17/2013 00:00:00', 'MM/DD/YYYY
HH24:MI:SS'),
'CO', 18, 'RCNUser');

```

Remove requests have the following mandatory input parameters:

- *REQUEST_ID* – The request ID of the root compare request. Remove Accumulator requests run on the data in DIFFER status that was written into the ADJ1_CMPR_DETAIL_RSLT table as a result of a comparison request. If the value of the RMV_STATUS field of an entry is ‘RD’, the remove request takes the entry and all accumulator keys and sends this data to the Event Server so that it removes the accumulator from AIMOS.

- *CYCLE_CODE* – The cycle code of the compare request that is the root of the remove request.
- *REMOVE_ACCUM_OPEN_SESSION* – Determines whether to remove the accumulator in an open session (valid values: Y/N). If the parameter is set to ‘Y’, the Event Server closes all the sessions that belong to the customer and removes the accumulator. If the accumulator to be removed belongs to an open session, and if the value is ‘N’, the status of the entry in the ADJ1_CMPR_DETAIL_RS LT table is changed to ‘ER’. The operator can change the status to ‘RD’ and rerun the request.
- *REMOVE_ALL* – Determines whether to remove all the accumulators at the current accumulator level (valid values: Y/N). If the accumulator is at the customer or agreement level, and if the value is ‘Y’, the Event Server removes all the accumulators at the customer and subscriber level. If the accumulator is at the subscriber level, the Event Server removes all the accumulators at the subscriber level.

Therefore, the corresponding rows must be inserted into the ADJ1_REQUESTS_PARAMS table, as follows:

```

Insert into ADJ1_REQUESTS_PARAMS
  REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, OPERATOR_ID, APPLICATION_ID, DL_SERVICE_CODE,
  DL_UPDATE_STAMP, NUMBER_VALUE)
Values
(3, 'CYCLE_CODE', TO_DATE('01/18/2013 12:52:32', 'MM/DD/YYYY HH24:MI:SS'), 0,
 'RCNUI ', 'v750 ', 0, 1);

Insert into ADJ1_REQUESTS_PARAMS
  REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, OPERATOR_ID, APPLICATION_ID, DL_SERVICE_CODE,
  DL_UPDATE_STAMP, VARCHAR_VALUE)
Values
(3, 'REMOVE_ACCUM_OPEN_SESSION', TO_DATE('01/18/2013 12:52:32', 'MM/DD/YYYY HH24:MI:SS'), 0,
 'RCNUI ', 'v750 ', 0, 'N');

Insert into ADJ1_REQUESTS_PARAMS
  REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, OPERATOR_ID, APPLICATION_ID, DL_SERVICE_CODE,
  DL_UPDATE_STAMP, VARCHAR_VALUE)
Values
(3, 'REMOVE_ALL', TO_DATE('01/18/2013 12:52:32', 'MM/DD/YYYY HH24:MI:SS'), 0,
 'RCNUI ', 'v750 ', 0, 'N');

Insert into ADJ1_REQUESTS_PARAMS
  REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, OPERATOR_ID, APPLICATION_ID, DL_SERVICE_CODE,
  DL_UPDATE_STAMP, NUMBER_VALUE)
Values
(3, 'REQUEST_ID', TO_DATE('01/18/2013 12:52:32', 'MM/DD/YYYY HH24:MI:SS'), 0,
 'RCNUI ', 'v750 ', 0, 1);

```

Sample Update Requests

This section contains sample SQL scripts for inserting update requests into the database.

Sample Update Resource Request

Following is a sample SQL script for a resource update request:

```
Insert into ADJ1_REQUESTS
(REQUEST_ID, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID, APPLICATION_ID,
DL_SERVICE_CODE, DL_UPDATE_STAMP, EXECUTION_ID, REQUEST_NAME, EFFECTIVE_DATE,
STATUS, ERROR_CODE, ERROR_MSG, DAY_IN_MONTH, REQUEST_CREATOR)
Values
(8, TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), TO_DATE('05/27/2010 14:11:05',
'MM/DD/YYYY HH24:MI:SS'), 7, 'tt', 'rr', 1, 22, 'UPDT_RESOURCE_IN_MEM', TO_DATE('03/14/2010
00:00:00', 'MM/DD/YYYY HH24:MI:SS'), 'RD', NULL, NULL, 2, NULL);
```

Update resource requests have two input parameters: Resource Value and Resource Type. Therefore, the corresponding rows must be inserted into the ADJ1_REQUESTS_PARAMS table, as follows:

```
Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(8, 'RESOURCE_VALUE', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, '5346546', NULL, NULL);
Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(8, 'RESOURCE_TYPE', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 3, NULL);
```

Sample Update Subscriber Request

Following is a sample SQL script for a subscriber update request:

```
Insert into ADJ1_REQUESTS
(REQUEST_ID, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID, APPLICATION_ID,
DL_SERVICE_CODE, DL_UPDATE_STAMP, EXECUTION_ID, REQUEST_NAME, EFFECTIVE_DATE,
STATUS, ERROR_CODE, ERROR_MSG, DAY_IN_MONTH, REQUEST_CREATOR)
Values
(11, TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), TO_DATE('06/03/2010 11:48:18',
'MM/DD/YYYY HH24:MI:SS'), 7, 'tt', 'rr', 1, 22, 'UPDT_SUBSCR_IN_MEM', TO_DATE('03/14/2010
00:00:00', 'MM/DD/YYYY HH24:MI:SS'), 'RD', NULL, NULL, 2, NULL);
```

Update subscriber requests have three input parameters: Subscriber ID, Cycle Code, and Customer ID. Therefore, the corresponding rows must be inserted into the ADJ1_REQUESTS_PARAMS table, as follows:

```

Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(11, 'SUBSCRIBER_ID', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 6, NULL);
Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(11, 'CYCLE_CODE', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 1, NULL);
Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(11, 'CUSTOMER_ID', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 4, NULL);

```

Sample Update Accumulators in Database Request

Following is a sample SQL script for a request to update accumulators in the database:

```

Insert into ADJ1_REQUESTS
(REQUEST_ID, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID, APPLICATION_ID,
DL_SERVICE_CODE, DL_UPDATE_STAMP, EXECUTION_ID, REQUEST_NAME, EFFECTIVE_DATE,
STATUS, ERROR_CODE, ERROR_MSG, DAY_IN_MONTH, REQUEST_CREATOR)
Values
(14, TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), TO_DATE('06/13/2010 16:21:48',
'MM/DD/YYYY HH24:MI:SS'), 7, 'tt', 'rr', 1, 22, 'UPDT_ACCUM_IN_DB', TO_DATE('03/14/2010
00:00:00', 'MM/DD/YYYY HH24:MI:SS'), 'RD', NULL, NULL, 2, NULL);

```

Requests to update accumulators in the database have the following input parameters: Owner Type, Owner ID, Offer Instance, Item ID, Cycle Year, Cycle Instance, Cycle Code, Customer ID, and Accumulator Type ID. Therefore, the corresponding rows must be inserted into the ADJ1_REQUESTS_PARAMS table, as follows:

```

Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(14, 'OWNER_TYPE', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, 'S', NULL, NULL);

```

```

Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(14, 'OWNER_ID', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), TO_DATE('03/15/2010
00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL, NULL, NULL, NULL, 4, NULL);
Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(14, 'OFFER_INSTANCE', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 56, NULL);
Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(14, 'ITEM_ID', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), TO_DATE('03/15/2010
00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL, NULL, NULL, NULL, 29806, NULL);
Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(14, 'CYCLE_YEAR', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 2008, NULL);
Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(14, 'CYCLE_INSTANCE', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 12, NULL);
Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(14, 'CYCLE_CODE', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 1, NULL);
Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(14, 'CUSTOMER_ID', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 4, NULL);

```

```
Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(14, 'ACCUM_TYPE_ID', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 1122, NULL);
```

Sample Update Accumulators in Memory Request



Note: It is recommended that users create requests to update accumulators in memory via the Reconciliation Tool GUI rather than directly in the database.

Following is a sample SQL script for a request to update accumulators in memory:



Note: Such requests do not update the key fields and core attributes of accumulators. For more information, see the “Update Requests” section in the “ADJ1RCN – Update Handler in Reconciliation Mode” chapter.

```
Insert into ADJ1_REQUESTS
(REQUEST_ID, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID, APPLICATION_ID,
DL_SERVICE_CODE, DL_UPDATE_STAMP, EXECUTION_ID, REQUEST_NAME, EFFECTIVE_DATE,
STATUS, ERROR_CODE, ERROR_MSG, DAY_IN_MONTH, REQUEST_CREATOR)
Values
(6, TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), TO_DATE('06/01/2010 13:39:10',
'MM/DD/YYYY HH24:MI:SS'), 7, 'tt', 'rr', 1, 11, 'UPDT_ACCUM_IN_MEM', TO_DATE('03/14/2010
00:00:00', 'MM/DD/YYYY HH24:MI:SS'), 'RD', NULL, NULL, 2, NULL);
```

Requests to update accumulators in memory have the following input parameters:
Version ID, Owner Type, Owner ID, Offer Instance, Item ID, Dimension ID, Cycle Year,
Cycle Instance, Cycle Code, Customer ID, Accumulator Type ID, and Accumulator
Attributes.



Note: The value of the ACCUMULATOR_ATTRIBUTES parameter must consist of the following elements:

- *Attribute name*
- *The operation required*
- *The value with which the attribute is to be updated*

These elements must be separated by ;@:.

For example, the value of 'PI status; ;@:+= ;@;2' as in the sample script below means that the existing value of the PI status attribute must be increased by 2.

The parameter can contain more than one attribute to be updated, separated by %@% (for example, 'PI status;@:==;@;4%@%record number;@:==;@;7').

The ACCUMULATOR_ATTRIBUTES parameter is not validated. Even if the same attributes appear a number of times, the request does not fail although it is not valid.

Therefore, the corresponding rows must be inserted into the ADJ1_REQUESTS_PARAMS table, as follows:

```

Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(6, 'VERSION_ID', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 1, NULL);

Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(6, 'UPDATE_ID', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), TO_DATE('03/15/2010
00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL, NULL, NULL, NULL, 1, NULL);

Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(6, 'OWNER_TYPE', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL, NULL, NULL, NULL, 'S', NULL,
NULL);

Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(6, 'OWNER_ID', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), TO_DATE('03/15/2010
00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL, NULL, NULL, NULL, 4, NULL);

Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(6, 'OFFER_INSTANCE', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 0, NULL);

Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(6, 'ITEM_ID', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), TO_DATE('03/15/2010
00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL, NULL, NULL, NULL, 32258,
NULL);

```

```

Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(6, 'DIMENSION_ID', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL, NULL, NULL, NULL, 1,
NULL);
Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(6, 'CYCLE_YEAR', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL, NULL, NULL, NULL, NULL, 2008,
NULL);
Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(6, 'CYCLE_INSTANCE', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 12, NULL);
Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(6, 'CYCLE_CODE', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 1, NULL);
Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(6, 'CUSTOMER_ID', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 4, NULL);
Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(6, 'ACCUM_TYPE_ID', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 245, NULL);

```

```
Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(6, 'ACCUMULATOR_ATTRIBUTES', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 'PI status; ;@;+= ;@;2', NULL, NULL);
```

Sample Update Customer Cycle History Request

Following is a sample SQL script for a request to update a customer's cycle history:

```
Insert into ADJ1_REQUESTS
(REQUEST_ID, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID, APPLICATION_ID,
DL_SERVICE_CODE, DL_UPDATE_STAMP, EXECUTION_ID, REQUEST_NAME, EFFECTIVE_DATE,
STATUS, ERROR_CODE, ERROR_MSG, DAY_IN_MONTH, REQUEST_CREATOR)
Values
(15, TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), TO_DATE('05/27/2010 13:14:34',
'MM/DD/YYYY HH24:MI:SS'), 7, 'tt', 'rr', 1, 22, 'UPDT_CUST_CYC_HIST_IN_MEM',
TO_DATE('03/14/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), 'RD', NULL, NULL, 2, NULL);
```

Update customer's cycle history requests have two input parameters: Cycle Code and Customer ID. Therefore, the corresponding rows must be inserted into the ADJ1_REQUESTS_PARAMS table, as follows:

```
Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(15, 'CYCLE_CODE', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 1, NULL);
Insert into ADJ1_REQUESTS_PARAMS
(REQUEST_ID, PARAM_NAME, SYS_CREATION_DATE, SYS_UPDATE_DATE, OPERATOR_ID,
APPLICATION_ID, DL_SERVICE_CODE, DL_UPDATE_STAMP, VARCHAR_VALUE, NUMBER_VALUE,
DATE_VALUE)
Values
(15, 'CUSTOMER_ID', TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'),
TO_DATE('03/15/2010 00:00:00', 'MM/DD/YYYY HH24:MI:SS'), NULL,
NULL, NULL, NULL, 342, NULL);
```

Appendix B Configuration Parameter Mechanism

Each process has a set of configuration parameters, which have values that Turbo Charging requires to function properly. Some of the parameters are process-driven (defined for a specific application or process), and some of the parameters are foundation-based (for example, Audit & Control parameters).

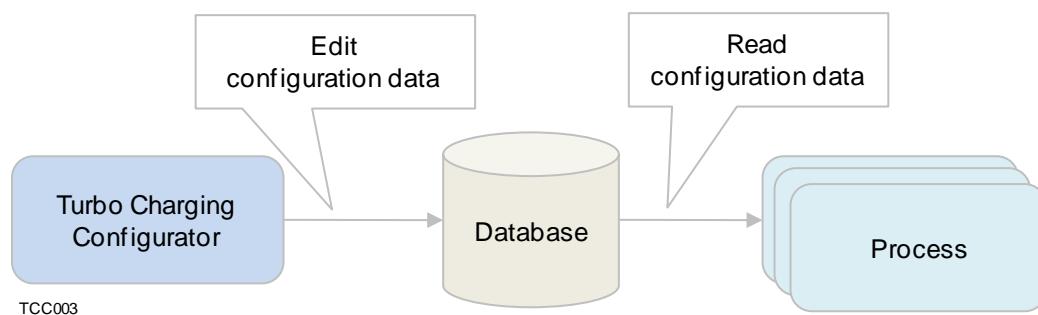
The configuration parameters of several Turbo Charging processes are stored in the database, and not in a properties file. This appendix describes the configuration parameter mechanism in Turbo Charging.

Overview

In Turbo Charging, processes of several process types, such as Event Server, DB2E, Update Handler, and Usage Query Server, can run simultaneously. The behavior of all processes is determined based on configuration data.

Figure B.1 shows the high-level architecture of the configuration parameter mechanism. Configuration parameters are created and updated using the Turbo Charging Configurator. Configuration data in the form of parameter names and values is loaded from a database. At runtime, this data is available to processes as a container of parameter name-value pairs that is searchable by the parameter name.

Figure B.1 High-Level Architecture



The configuration parameter mechanism enables applications to get process configuration data that the components of an application require without duplicating configuration tables.

This mechanism enables loading and finding configuration parameters at runtime and is made of two major parts:

- An in-memory repository for named parameters, organized in a hierarchy
- A utility class that enables to easily populate process property objects with reference data that resides in the database

The basic capabilities of the configuration parameter mechanism are:

- Provide processes with parameter values that are organized in each section in a pair of parameter name and value
- Provide a hierarchy that allows one process to inherit parameters from another process

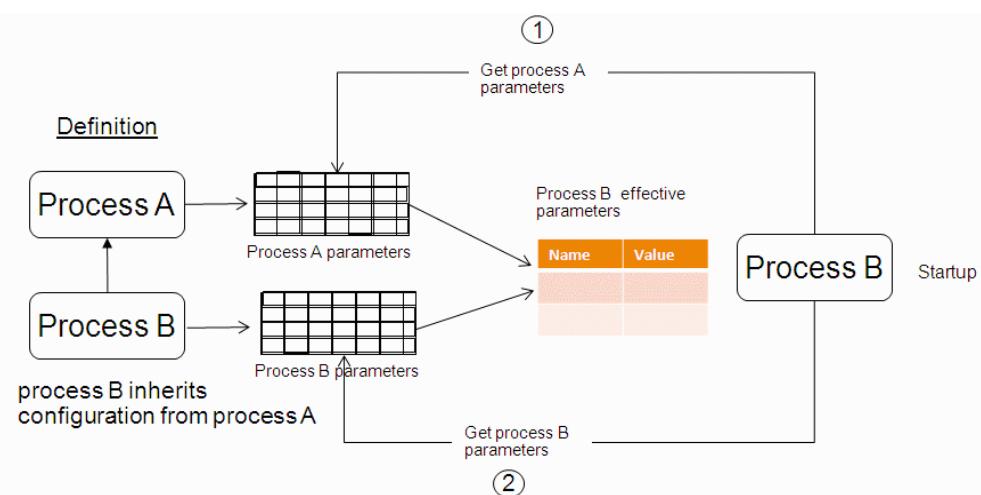
Parameter Inheritance

Process hierarchy is defined in the Configuration Hierarchy table of the specific component.

For example, process B can define that it is dependent on (or inherits from) process A. The configuration parameter mechanism finds and loads the parameters of both process A and B when process B starts. Furthermore, process B can override the values of parameters defined by process A by specifying a parameter with the same name that has a different value.

Figure B.2 describes how configuration parameters are inherited and loaded.

Figure B.2 Configuration Parameter Mechanism Architecture



Process A and process B both define parameters in database tables. Process B is also configured to inherit the parameters of process A. When process B starts, the parameter loading mechanism first loads the parameters of process A into a container in memory and then loads the parameters of process B into the same container. Because the parameter name is used as the key, if process B defines parameters using the same name that is used in process A, the parameters defined in process B override the parameters defined in process A.

Thus, process B uses the following configuration parameters:

- *Parameters that are defined in process A* – Uses the values from process A
- *Parameters that are defined in both process A and B* – Uses the values from process B
- *Parameters that are defined in process B*

This mechanism supports any number of inheritance levels and allows each process to inherit from more than one process.

Configuration Parameters

This section describes foundation-based and process-driven parameters.

Foundation-Based Parameters

All process types inherit generic configuration parameters of the relevant foundation component. For example, a process that handles files, and logs its messages and errors, uses configuration parameters defined for the Audit & Control (files) and Logger (messages) layers.

The ADJ1_COMPONENTS_REF table defines all foundation components relevant for Turbo Charging and their respective configuration tables. This information is used to show and update the configuration parameters of these foundation components.

The foundations and applications that are used by a specific *process type* are defined in the Turbo Charging's Hierarchy Configuration tables (ADJ1_CONF_HIERARCHY, APE1_CONF_HIERARCHY, APR1_CONF_HIERARCHY, and AGD1_CONF_HIERARCHY). For more information on these tables, see *Turbo Charging Data Model*.

Process-Driven Parameters

There are different types of parameters:

- *Parameters that are the same for all instances of the process.*
In this case, the parameters *and their values* are defined at the process type level. Each process instance inherits the values from its parent process type.
- *Parameters that have a default value for all instances of the process, but this value can be overwritten.*
In this case, the parameters *and their default values* are defined at the process type level. Each process instance inherits the values from its parent process type. If the user wants to overwrite the value, the user must create an instance of the parameter at the process instance level.
- *Parameters that have specific values per process instance.*
In this case, the parameters are defined at the process type level. No default value is stored. Once the parameter is defined at the process type level, the process instance must define the value of the parameter.

Defining Configuration Parameters

A component or foundation may have several parameters. Even though not all parameters are necessarily used by a specific process directly, these parameters may require values.

To provide smart editing capabilities for parameters in Turbo Charging Configurator, each parameter has metadata associated with it. This metadata enables the Turbo Charging Configurator to perform validation on parameter values, display choice lists with valid values, apply rules when editing parameters, and so on. All parameters in the system with their metadata are defined in the Configuration Parameters (ADJ1_CFG_PARAMS) table. When a new parameter is created for one of the processes in the system, it is created based on one of the parameters previously defined in the ADJ1_CFG_PARAMS table and persisted into the Section Parameter Configuration (*nnn1_CONF_SECTION_PARAM*) table of the relevant process (where *nnn* is the abbreviated name of the process). It is also possible for different applications to share the same table and use different class names and parameter names to determine which parameter belongs to which application. For more information on these tables, see *Turbo Charging Data Model*.

For each configuration parameter, the following values must be set.

Parameter Class

The parameter class is used to define the type of process that uses the parameter. The class can represent foundation components (for example, C++ Foundation), a specific process type (for example, Event Server or DB2E), or a specific instance of one of the process types (for example, DB2E instance number 830).

When parameters are loaded at runtime, each process looks for parameters of its class (type).



Note: In some Turbo Charging processes, the parameter class is defined as a numeric value because it is numeric in the process code. It is still possible to check which process instance the setting is for and what its process type is by joining with the GN1_SYS_PROC_INSTANCE_CFG and GN1_SYS_PROCESS_TYPE_CFG tables. The parameter class must match one of the process instance IDs in the GN1_SYS_PROC_INSTANCE_CFG table.

Section Name

The section name is used to divide parameters of the same class into different logical groups, for example, logging settings, database settings, caching settings, and more.

Sections are named entities. Section names are case-sensitive and they follow the hierarchical naming rule. A section is said to be an ancestor of another section if its name followed by a dot is a prefix of the descendant section name. A section is said to be a parent of a child section if there are no ancestors between this section and the descendant section. For example, the section named logger.output is a parent of the section named logger.output.file.

The following example describes how section names are used.

Example:

Assume that configuration data is as follows.

Section Name	Parameter Name	Value
UH	CONNECTION_TIMEOUT	30
UH	RETRY_INTERVAL	10
UH.Database	USE_ORACLE_RAC	True
UH.Database.ConnectionPool	NUM_OF_CONNECTION	5

- *Scenario 1* – The application must get the value of the NUM_OF_CONNECTION parameter in the UH.Database.ConnectionPool section. In this case, the application gets the value of the parameter directly in this section.
- *Scenario 2* – The application must get the value of the CONNECTION_TIMEOUT parameter in the UH.Database.ConnectionPool section. The application will not find the value in this section and will try again in the parent section, UH.Database. Again, it will fail and will try in the next section up the hierarchy, UH. This time, the value 30 will be found.

Parameter Name

The parameter name is the name that is used in the name-value pair that is loaded into the properties container at runtime. The application looks for the parameter value by its name. Parameter names are not case-sensitive and must not contain spaces.

Parameter Value

Parameters values are stored in the database as strings. It is the responsibility of the application to convert the value to a different type (numeric, date, or Boolean) and perform the appropriate validations.

Final Parameters

One parameter name can be defined multiple times under different section names. As described above, section names are hierarchical. The parameter value is searched up the hierarchy from child to parent, and the first match found is used.

In some cases, a parameter that is defined at an upper level must not be overridden by any child levels. In this case, the parameter is defined as final, and the properties loader ignores any values at child levels and loads only the value that is defined at the level marked as final.

Customization Level

The core product may be customized by accounts, and this customization may require a change to some properties defined by the core components. To change a value of a core property without modifying the core property itself, the same parameter name, class, and section can be defined with a higher customization level. When loading properties, the higher value is used, and the others are ignored.

Defining Parameter Inheritance

Configuration parameter hierarchy is defined at two levels:

- Section hierarchy, which enables defining parameters and finding them from the most specific (for example, UH.Database.ConnectionPool) to the less specific(for example, UH) section
- Inheritance, which enables one process to inherit parameters defined by another process

Inheritance of parameters between different processes or applications is defined in the Configuration Hierarchy table (*nnn1_CONF_HIERARCHY*, where *nnn* is the abbreviated name of the application that uses this table). For more information about this table, see *Turbo Charging Data Model*.

Process Groups

The Process Group column of this table contains the information about the configuration parameter (property) hierarchy that is relevant for the process name defined in the same record.

The value of this field must comply with the following convention:

Parent name[[:target table name [:process name in target table]], [Parent name...]

where:

- *Parent name* – The name of the parent process to inherit parameters from.
- *Target table name (optional)* –The name of the table containing the parent process configuration parameters. If no target table is defined, the parameters are searched for in the current table.
- *Process name in target table (optional)* – This name cannot appear if the target table value is not specified.

The string in the process group column may contain multiple values separated by a comma. The search order is from the first value to the last.

Example:

DB2E,APR,ADJ:ADJ1_CONF_SECTION_PARAM: ADJ1_CONF_HIERARCHY

First look at class name DB2E in the same table, then look for the hierarchy definitions of ADJ in the ADJ1_CONF_HIERARCHY table, and then look for class name ADJ in the ADJ1_CONF_SECTION_PARAM table.

Cross-Application Configuration

For cross-application configuration sharing, one application must know where to find the parameters of another application whose configuration it inherits. For example, if the Event Server inherits configuration from Audit & Control, it needs the name of the CONF_SECTION_PARAM table used by Audit & Control to store its parameters.

To achieve that, the same *nnn1_CONF_HIERARCHY* table is used. As described above, in addition to the section name, the entries in the Process Group field may contain the name of the target table to get the data from. The properties loader that parses the hierarchy string gets the table name and looks for the properties of the requested parameter class in that table.

Loading Configuration Parameters

The parameters defined in the database are loaded into memory and stored in a searchable container available to the application that needs them.

The C++ foundation provides a utility class that users can use to easily populate their process Properties object with database-resided data (reference). For Java-based processes, a Java properties loader and properties container are the foundation components that may be used by any application to load and search for parameters at runtime.