Oracle® Communications Billing and Revenue Management

Configuring Pipeline Rating and Discounting

Release 7.5

E16710-16

December 2016

ORACLE

Oracle Communications Billing and Revenue Management Configuring Pipeline Rating and Discounting, Release 7.5

E16710-16

Copyright © 2011, 2016, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

2 Configuring Pipeline Rating

3 Configuring EDR Input Processing

4 Configuring EDR Output Processing

5 Configuring EDR Preprocessing

6 Setting Up EDR Enrichment

7 Setting Up Pipeline Aggregation

8 Migrating Pipeline Manager Data between Test and Production Systems

9 Transferring Data Between Pipeline Manager Databases

Part II Configuring Pipeline Discounting

10 About Discounts

11 About Implementing Discounts

12 Configuring Discounting Modules and Components

13 Discount Sharing Configuration Example

19 Using Standard Recycling to Recycle Suspended EDRs

20 About Suspense Manager

21 Installing Suspense Manager

22 Configuring Suspense Manager

23 Using Suspense Manager

24 Suspense Reasons

25 About Suspense Manager Opcodes

26 Suspense Management Utilities

27 Recycling EDRs in Pipeline-Only Systems

Part IV Twin Talk Enabler

28 About Twin Talk Enabler

29 Sample Twin Talk Enabler Configuration Procedures

37 Pipeline Manager Data Modules

38 Pipeline Manager iRules

39 Pipeline Manager iScripts

40 Pipeline Manager Input and Output Modules

41 Pipeline Manager Framework Modules

42 Pipeline Manager Utilities

Preface

The book describes Oracle Communications Billing and Revenue Management pipeline rating.

Audience

This book is intended for system administrators and developers.

Accessing Oracle Communications Documentation

BRM documentation and additional Oracle documentation; such as Oracle Database documentation, is available from Oracle Help Center:

http://docs.oracle.com

Additional Oracle Communications documentation is available from the Oracle software delivery Web site:

https://edelivery.oracle.com

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Document Revision History

The following table lists the revision history for this book.

Version	Date	Description
E16710-01	November 2011	Initial release.

Version	Date	Description
E16710-02	May 2012	Documentation updates for BRM 7.5 Patch Set 1. ▪ Made minor formatting and text changes. ▪ Added documentation for the Rated Event Loader daemon: About Running the RE Loader Daemon Configuring the RE Loader Infranet.properties File Configuring the start_rel_daemon Script for an Oracle IMDB Cache System Running the RE Loader Daemon
E16710-03	August 2012	Documentation updates for BRM 7.5 Patch Set 2. ▪ Added documentation about the **ReloadCreditThresholdParam** semaphore entry for the following pipeline modules: DAT_BalanceBatch FCT_ApplyBalance ▪ Added documentation about the Oracle 11*g*R2 installation requirement to the "Configuring Oracle Libraries for RE Loader" section. ▪ Added documentation for the following iScripts: ISC_RAP_0105_InMap UpdateTapInfo_StopRapout UpdateTapInfo_Tapin ▪ Added documentation for the "StopRapGen" utility. ▪ Updated documentation for "Output Controller".
E16710-04	December 2012	Documentation updates for BRM 7.5 Patch Set 3. ▪ Made minor formatting and text changes. ▪ Updated and renamed the following sections: ISC_TAP_0312_Include ISC_TAP_0312_InMap ISC_TAP_0312_Validations ISC_TapDetailValidation_v3_12 ISC_TapHeaderTrailerValidation_v3_12 ▪ Added the "Installing RE Loader on Non-IMDB Cache Enabled Systems" topic. ▪ Added documentation for policy-driven charging to the following sections: About Discounting Additional Discount Configuration
E16710-05	March 2013	Documentation updates for BRM 7.5 Patch Set 4. ▪ Made minor formatting and text changes. ▪ Added new balance impact EDR fields in Table 36–23. ▪ Added documentation for the "RoamingConfigGen64" utility. ▪ Added documentation for the "Instances" module. ▪ Replaced multidatabase information with multischema information.

Version	Date	Description
E16710-06	July 2013	Documentation updates for BRM 7.5 Patch Set 5. ■ Made minor formatting and text changes.
E16710-07	August 2013	On HP-UX IA64, BRM 7.5 is certified as of BRM 7.5 Patch Set 5. Documentation added for HP-UX IA64.
E16710-08	February 2014	Documentation updates for BRM 7.5 Patch Set 7. ■ Made minor formatting and text changes.
E16710-09	May 2014	Documentation updates for BRM 7.5 Patch Set 8. ■ Added documentation for the "load_edr_field_mapping" utility.
E16710-10	October 2014	Documentation updates for BRM 7.5 Patch Set 10. ■ Updated the documentation about "DAT_PriceModel" and "DAT_Rateplan". ■ Updated the "Starting and Using Suspense Management Center" section.
E16710-11	January 2015	Documentation updates for BRM 7.5 Patch Set 11. ■ Updated Table 37–3, " DAT_BalanceBatch Registry Entries".
E16710-12	June 2015	Documentation updates for BRM 7.5 Patch Set 12. ■ Made minor formatting and text changes. ■ Added new entries in Table 32–3 and Table 32–5. ■ Added the "Setting Up RE Loader to Load Preprocessed Rated Events from ECE into BRM" topic.
E16710-13	August 2015	Documentation updates for BRM 7.5 Maintenance Patch Set 1. ■ Replaced the phrase "BRM CDR format" with the phrase "Oracle CDR format". ■ Updated the following sections: Adding Custom Fields to Suspense Management Center Setting Up Logging of Debugging Information
E16710-14	December 2015	Documentation updates for BRM 7.5 Patch Set 14. ■ Updated the following rows in Table 1–1, " Functional Modules and Processing Dependencies": FCT_Discount and FCT_ApplyBalance. ■ Updated "FCT_ApplyBalance" dependencies. ■ Updated "FCT_Discount" dependencies. ■ Updated the list of suspense reason codes (see "Suspense Reasons"). ■ In Table 34–6, " Basic Detail Record Fields", changed the maximum value of NUMBER_OF_UNITS. ■ Updated the "FCT_Timer" section.
E16710-15	August 2016	Documentation updates for BRM 7.5 Patch Set 16. ■ Updated "ISC_ProfileLabel" dependencies. ■ Updated "pin_discount_cleanup" section.

Version	Date	Description
E16710-16	December 2016	Documentation updates for BRM 7.5 Patch Set 17. ■ Updated "DAT_ItemAssign" section.

Part I

Configuring Pipeline Rating

Part I describes how to configure pipeline rating in an Oracle Communications Billing and Revenue Management (BRM) system. It contains the following chapters:

1

About Pipeline Rating

This chapter provides an overview of how to use the Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager to rate usage events.

Before reading this document, you should be familiar with BRM concepts and architecture. See the following documents:

- Introducing BRM
- BRM system architecture

How Events Are Rated by Using Pipeline Manager

Pipeline Manager rates usage events as follows:

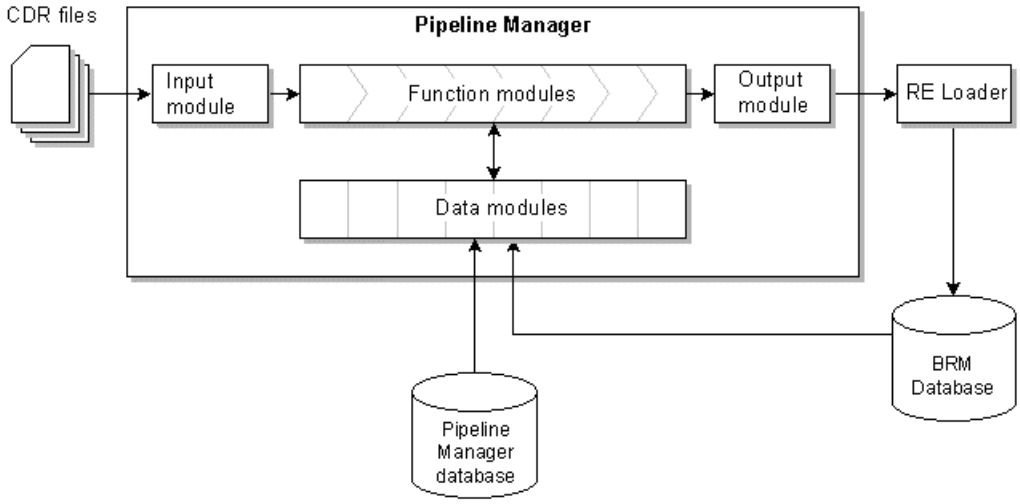
1. Call detail record (CDR) files are collected from network switches and processed by mediation software. The mediation software converts data into the Oracle CDR format so it can be read by a pipeline input module. See "About the Oracle CDR Format".

   Mediation can also remove records that are not needed for rating. For example, it can remove records for free text messages reminding customers of a missed call.

2. The mediation software places the CDR file in a predefined directory.
3. The pipeline input module reads the CDR file and begins processing it. The input module does the following:
   - Performs error checking on the input CDR.
   - Converts the data in the CDR file into event data records (EDRs). To do so, the input module normalizes the raw data that represents each event and formats it into a standard structure that can be processed by the pipeline modules.
4. Function modules, working with data modules, rate the events.
   - *Function modules* perform the preprocessing and rating operations. For example, they check for duplicated EDRs, determine the quantity to rate, the zone to apply, and the charge for the event.
   - *Data modules* supply data to the function modules by reading from the Pipeline Manager database or the BRM database.
5. The output module writes data to an output file.
6. Rated Event (RE) Loader reads the output file, loads the events into the BRM database, and updates the customer's account balance.

Figure 1–1 shows how usage events are rated:

Figure 1–1 Usage Events Rating



When you configure Pipeline Manager, you do the following:

- Use Pricing Center to configure rating, pricing, and zoning data.
- Configure the input module to accept and process your type of CDR file.
- Configure function and data modules to carry out normalization, zone mapping, and rating.
- Configure the output module to output the data.

In addition to configuring Pipeline Manager, you need to do the following:

- Configure the Account Synchronization Data Manager (DM) to transfer data from the BRM database to Pipeline Manager. See "Installing and Configuring the Account Synchronization DM" in *BRM Installation Guide*.
- Configure RE Loader to load rated events into the BRM database. See "Understanding Rated Event Loader".

How an EDR Is Processed in a Pipeline

When an EDR is sent through a pipeline, function modules perform the following types of operations:

- The input module processes the CDR and creates an event data record (EDR) for each event. Processing the CDR file includes:
  - Identifying each type of record in the file; for example, header records, event records, and trailer records.
  - Normalizing data and translating it into the internal EDR format.
- Preprocessing modules prepare EDRs for rating. For example:
  - The FCT_DuplicateCheck module discards duplicate EDRs.
  - The FCT_CallAssembling module assembles calls that have been split into multiple records.
  - The FCT_Reject module rejects EDRs with errors.

  Preprocessing modules typically run at the beginning of a pipeline.

- Enrichment modules normalize or add data that the rating modules need. For example:
  - The FCT_ServiceCodeMap module assigns an internal service code to identify which service generated the event.
  - The FCT_CliMapping module maps multiple phone numbers to a single phone number, so customers can be billed for all of their phones on one bill.

  Enrichment modules typically run before rating modules in a pipeline.
- Zoning modules calculate geographic or area-code-based zones for rating purposes.
- Rating modules perform rating.
- Discount modules perform discounting, after the EDR has been rated.
- The aggregation module collects data for reports.

About the Order of Modules in a Pipeline

Some modules need to be configured in a specific order. For example:

- You use the FCT_Discard module to discard EDRs that you don't want to rate. You need to discard them before the rating modules process them; otherwise you spend system resources on rating unwanted EDRs.
- To provide discounts, the discount module needs to work with events that have already been processed by the rating modules.

For information about the order of modules in a pipeline, see "Function Module Dependencies".

About the Oracle CDR Format

The Oracle CDR format is the standard CDR file format used by Pipeline Manager for processing CDRs. For example, the Oracle CDR format is used by pipelines to generate CDRs that are passed between pipelines for additional processing, such as between a preprocessing pipeline and a rating pipeline.

The Oracle CDR format can also be used as the input format for rating pipelines. Additionally, pipelines can be configured to translate other formats into the Oracle CDR format for rating.

For complete details about the Oracle CDR format structure, see "BRM Rating EDR Container Description".

For information about the input and output processes, see "Configuring EDR Input Processing" and "Configuring EDR Output Processing".

About EDRs

The data for each event is stored as an EDR. As an EDR is processed, function modules process data in it or add data. For example, the FCT_CustomerRating module adds the rate plan code. The FCT_MainRating module uses that code to calculate and add the charge amount.

The following sample shows a portion of a charge packet in an EDR. Each field stores a specific piece of data; for example, the RATEPLAN_CODE field stores the rate plan used for rating the event.

```
CHARGE_PACKET
RATEPLAN_CODE:                <SIMPLERATE>
RATEPLAN_TYPE:                <R>
ZONEMODEL_CODE:               <MT_ZM>
SERVICE_CODE_USED:            <SMS>
SERVICE_CLASS_USED:           <DEF>
IMPACT_CATEGORY:              <MT_SMS>
TIMEMODEL_CODE:               <TM_MTEL>
TIMEZONE_CODE:                <TZ_WKNDDT>
DAY_CODE:                     <WEEKEND>
TIME_INTERVAL_CODE:           <0800_2000>
PRICEMODEL_CODE:              <MT_SMS>
PRICEMODEL_TYPE:              <S>
RESOURCE:                     <YEN>
RUMGROUP:                     <EVENT>
RUM:                          <EVT>
CHARGE_TYPE:                  <N>
CHARGING_START_TIMESTAMP:     <20020901182200>
CHARGEABLE_QUANTITY_VALUE:    <1>
CHARGED_CURRENCY_TYPE:        <R>
CHARGED_AMOUNT_VALUE:         <50>
CHARGED_AMOUNT_CURRENCY:      <JPY>
CHARGED_TAX_TREATMENT:        <N>
CHARGED_TAX_RATE:             <0>
CHARGED_TAX_CODE:             <NORM>
USAGE_GL_ACCOUNT_CODE:        <50000>
```

About EDR Containers

EDR containers are temporary in-memory data structures for transporting EDRs through function modules. Each container stores data for a header record, a detail record, or a trailer record.

When a transaction starts:

1. The EDR Factory creates an EDR container according to the information in the container description file.
2. The input module writes data to relevant fields in the container. For more information, see "Configuring EDR Input Processing".
3. The function modules process data in or add data to particular fields. For example, the FCT_BillingRecord function module processes balance impact-related data.
4. The EDR Factory empties the container and releases the cache, which will be used for other containers.

About the EDR Container Description

The data in an EDR, the default values, and the way the data is organized are defined in a *container description*. A typical container description defines the following types of containers:

- **A header container type.** The header container type contains information stored in a header record; for example, the country of origin, originating network, and creation time. It also includes sequence numbers for ensuring that EDRs are processed correctly. A separate EDR is created for the header record.
- **One or more basic detail container types.** A basic detail container includes the data in an event record; for example, a phone call. This record includes information such as the A number and the service that generated the event. An

EDR is created for each detail record. Each EDR can contain data for one service only; for example, GSM or GPRS.

Each detail record includes one or more associated records. These records include service-specific data, zoning data, and rating data. See "About Associated Records".

- **A trailer container type.** The trailer container type contains information stored in a trailer record; for example, the number of records. A separate EDR is created for the trailer record.

In almost all cases, you can use the BRM EDR container description. You can also customize the data that is included in an EDR by customizing the container description. See "Modifying and Loading the EDR Container Description" in *BRM Setting Up Pricing and Rating*.

Important: If you customize the container description, you need to make sure that your customizations do not affect existing module functionality. For example, many modules require data from a specific EDR field.

About the Container Description File

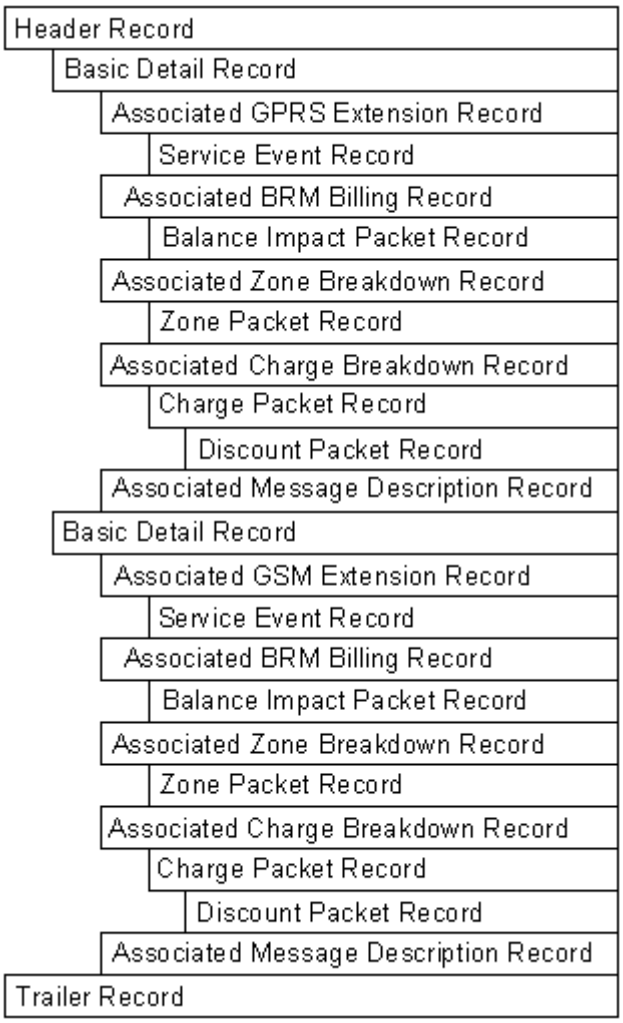
The container description file is an ASCII file that defines how to build EDR containers. You use the file to define the data in the EDR container, the default values, and the way the data is organized. For more information, see "About the EDR Container Description".

The default container description file is *Pipeline_Home*/**formatDesc/Portal/containerDesc.dsc**, where *Pipeline_Home* is the directory in which you installed Pipeline Manager. This file covers the needs of most input file formats, but you can customize it to meet your business requirements. For information, see "Modifying and Loading the EDR Container Description" in *BRM Setting Up Pricing and Rating*.

Note: If you alter or add fields to the container description file, you may also need to write custom iScripts to access these new fields.

Figure 1–2 shows the BRM EDR container organization:

Figure 1–2 BRM EDR Container Organization



About Associated Records

Associated records are added to EDRs by different modules; for example, the FCT_SegZoneNoCust module adds an associated zone breakdown record.

- **Associated service extension record**. This record stores specific information about the service that generated the event; for example, the originating switch and the device number.

  EDRs for GSM events can include one or more service event records. This record includes event information, such as equipment used and details about supplementary services.

- **Associated BRM billing record**. This record stores rated event data that is loaded into the BRM database. It includes the POID of the **/account** object and **/service** object and the POID of the item that receives the balance impact. This record is created by the FCT_BillingRecord module.

  If an event affects more than one customer balance, an associated BRM billing record is created for each balance.

An associated BRM billing record can contain one or more *balance impact packets*. These contain data about the event. Each balance impact packet includes data for one balance impact per resource, and, optionally, per G/L ID.

- **Associated charge breakdown record**. These records hold the charges for an event. For example, the FCT_CustomerRating module adds an associated charge breakdown record to the EDR to record the rating results.

  Each associated charge breakdown record includes one or more charge packets. Each charge packet includes a single charge; for example, the charge for a telephone call for a single time period.

  Each charge packet can include one or more *discount packets*. A discount packet includes information about the discount owner and rollover information.

  During rating, Pipeline Manager might generate several charge packets. For example:

  - When rating multiple resources, each resource has its own charge packet.
  - When using multiple ratable usage metrics (RUMs), each RUM has its own charge packet.
  - When splitting charges across time zones, each time zone has its own charge packet.
  - If multiple currencies are used, each currency has its own charge packet.

- **Associated zone breakdown record**. This record contains data for zoning. It is created by the FCT_SegZoneNoCust module. This is used for comparative analysis of different zoning options.

  An associated zone breakdown record includes one or more *zone packet records*. Each of these records includes data about a single zone model.

- **Associated message description record**. This record holds information and error messages.

How an Input File Is Represented in EDR Containers

A typical conversion from an input file to EDR containers creates the following EDRs:

- A set of control EDRs that the pipeline uses for managing transactions. For example, this control EDR specifies to start a transaction:

```
= = = = =  S T A R T   O F   C O N T A I N E R  = = = = =
Container type = <SERVICE>
Content type   = <BEGIN_TRANSACTION>
Originator     = <ifw.Pipelines.W_SAMPLE.Input>
Stream Number  = <0>
IsValidDetail  = <false>
Record number  = <1>
has Errors     = <0>
.
.
.
= = = = = =  E N D   O F   C O N T A I N E R  = = = = = =
```

- A header record that includes information about the input file:

```
= = = = =  S T A R T   O F   C O N T A I N E R  = = = = =
Container type = <DATA>
Content type   = <HEADER>
Originator     = <>
Stream Number  = <3>
```

```
IsValidDetail  = <false>
Record number  = <0>
has Errors     = <0>

HEADER
------

RECORD_LENGTH:                    <0>
RECORD_TYPE:                      <>
RECORD_NUMBER:                    <0>
.
.
.
= = = = = =  E N D   O F   C O N T A I N E R  = = = = = =
```

- Multiple detail records, each containing the data for one event:

```
= = = = =  S T A R T   O F   C O N T A I N E R  = = = = =
Container type = <DATA>
Content type   = <DETAIL>
Originator     = <>
Stream Number  = <3>
IsValidDetail  = <true>
Record number  = <1>
has Errors     = <0>

DETAIL
------

RECORD_LENGTH:                    <0>
RECORD_TYPE:                      <020>
RECORD_NUMBER:                    <0>
DISCARDING:                       <0>
TYPE_OF_A_IDENTIFICATION:         <S>
A_MODIFICATION_INDICATOR:         <00>
A_TYPE_OF_NUMBER:                 <0>
A_NUMBERING_PLAN:                 <0>
A_NUMBER:                         <008190115551212>
.
.
.
= = = = = =  E N D   O F   C O N T A I N E R  = = = = = =
```

- A trailer record, which includes information about the records in the file:

```
= = = = =  S T A R T   O F   C O N T A I N E R  = = = = =
Container type = <DATA>
Content type   = <TRAILER>
Originator     = <>
Stream Number  = <3>
IsValidDetail  = <false>
Record number  = <7>
has Errors     = <0>

TRAILER
-------

RECORD_LENGTH:                    <0>
RECORD_TYPE:                      <090>
RECORD_NUMBER:                    <14>
.
.
.
```

```
= = = = = =  E N D   O F   C O N T A I N E R  = = = = = =
```

- A set of control EDRs. This EDR specifies the end of a transaction:

```
= = = = =  S T A R T   O F   C O N T A I N E R  = = = = =
Container type = <SERVICE>
Content type   = <END_TRANSACTION>
Originator     = <ifw.Pipelines.W_SAMPLE.Input>
Stream Number  = <0>
IsValidDetail  = <false>
Record number  = <8>
has Errors     = <0>
.
.
.
= = = = = =  E N D   O F   C O N T A I N E R  = = = = = =
```

How EDRs Are Used for Managing Transactions

Transactions are managed by using the EDR content type:

- When an EDR with the BEGIN_TRANSACTION content type is processed, a new transaction is started. This usually occurs at the beginning of each input file.
- When an EDR with the END_TRANSACTION content type is processed, the transaction is ended. This usually occurs at the end of each input file.
- When an EDR with the STOP content type is processed, this indicates that the pipeline is shutting down gracefully and gives the modules a chance to save their state and prepare for the shutdown.

For more information, see "About Pipeline Manager Transactions" in *BRM System Administrator's Guide*.

About Mapping EDR Field Names and Alias Names

The EDR container description defines the data in the EDRs and the name of each field. See "About the EDR Container Description".

In the pipeline modules, EDR fields can optionally be represented using two names:

- The EDR container field name; for example, DETAIL.RECORD_LENGTH. This is the input format of the field.
- An alias name, to which the container field name is mapped; for example, BDR_RECORD_LENGTH.

The sample below shows EDR container field names on the left and internal alias names on the right:

```
DETAIL.RECORD_LENGTH     ->  BDR_RECORD_LENGTH
DETAIL.RECORD_TYPE       ->  BDR_RECORD_TYPE
DETAIL.RECORD_NUMBER     ->  BDR_RECORD_NUMBER
DETAIL.DISCARDING        ->  DISCARDING
DETAIL.CHAIN_REFERENCE   ->  CHAIN_REFERENCE
```

Pipeline Manager function modules and iScripts use the alias name for manipulating data. When you write a custom iRule or an iScript, you must create an alias name for the EDR container fields that your module uses. Using alias names facilitates customization because you can use the container fields with different names and hierarchies without the need to change the module source code. See "Viewing and Creating Alias Mapping for an EDR Field".

> **Note:** The same EDR container field can be mapped to different alias names in different modules.

The list of BRM-defined alias names is in *Pipeline_Home*/**formatDesc/Portal/AliasFieldList.dsc**. You can view all the existing BRM-defined and custom alias names in the alias mapping table in Pricing Center.

Viewing and Creating Alias Mapping for an EDR Field

Use **Pipeline Setup Toolbox -EDR- EDR Container Description - Alias Mapping** in Pricing Center to view a list of existing alias mappings and to add custom alias mappings.

See Pricing Center Help.

The Alias Mapping table lists the existing alias names including the BRM-defined alias names. Each entry includes the following mapping information:

- **EDR Container Description**, which specifies the EDR container to which the EDR field belongs.
- **Reference**, which is the pipeline module that uses the alias name. Its value is the pipeline module reference in the module block of the pipeline registry file; for example, **PipelineSplit** in the following registry entry:

```
PipelineSplit
{
   ModuleName = FCT_IRules
   Module
  {
    …
   }
}
```

> **Note:** If the value of **Reference** is Account_CustA, Account_CustB, UniData_CustA, or UniData_CustB, the alias mapping table entry defines the EDR field from which to get the account identifier. If the value is RUM, the alias mapping defines the EDR field from which to get the rateable usage metrics (RUM) quantity. If the value is UOM, the alias mapping defines the EDR field from which to get the unit of measure for a RUM

- **Key**, which is the alias name for the EDR container field.

> **Note:** For Account_CustA, Account_CustB, UniData_CustA, and UniData_CustB, this is the service code.

- **Type**, which is **Internal** for the alias mappings used by the default pipeline modules and **Plugin** for the modules used by custom modules.
- **Field ID**, which is the name of the field in the EDR container.
- **History**, which specifies when the alias mapping was created or modified.

About Function Modules

Function modules perform all the rating tasks in a pipeline. In addition, they perform EDR management tasks, such as ensuring that duplicate EDRs aren't processed.

There are different categories of function modules. They generally run in the following order:

- Preprocessing modules prepare EDRs for rating. See "About Preprocessing Modules".
- Enrichment modules add data to EDRs. See "About Enrichment Modules".
- Service mapping modules map external service codes to internal service codes. See "About Service Mapping Modules".
- Zoning modules calculate zone data to use for rating. See "About Zoning Modules".
- Rating modules rate the events. See "About Rating Modules".
- Discount modules adjust the charges. See "About Discounting Modules".
- Roaming modules are specialized zoning and rating modules that handle roaming events. See "About Roaming Modules".

About Preprocessing Modules

Use the preprocessing modules to handle the following tasks:

- Use the "FCT_CallAssembling" module to assemble calls that have been split into multiple EDRs. See "Assembling EDRs".
- Use the "FCT_Reject", "FCT_PreSuspense", and "FCT_Suspense" modules to handle EDRs that contain errors. Rejected EDRs are sent to a separate output stream. You can then fix the problem that caused them to be rejected and re-process them. See "Suspending and Recycling EDRs".
- Use the "FCT_DuplicateCheck" module to discard duplicate EDRs. See "Handling Duplicate EDRs".
- Use the "FCT_Discard" module to discard EDRs that you don't want to process. See "Discarding and Skipping EDRs".
- Use the "FCT_EnhancedSplitting" module to send EDRs to different output streams based on rules that you define. For example, you can split EDRs from roaming outcollects and incollects into different streams. See "Using Rules to Send EDRs to Different Output Streams".
- Use the "IRL_EventTypeSplitting" iRule to send EDRs to different output streams based on the service. See "Sending EDRs to Pipeline Output Streams".
- Use the "FCT_AccountRouter" to send EDRs to the correct pipeline in a multischema system. See "Using Pipeline Manager with Multiple Database Schemas".

You set up EDR preprocessing by configuring modules in a pipeline. In some cases, you use Pricing Center to configure how a module works; for example, to set up discard rules. For information on configuring preprocessing modules, see "Configuring EDR Preprocessing".

About Enrichment Modules

When you enrich EDR data, you add or change data for further processing.

- Use the "FCT_CliMapping" module to specify that multiple phone numbers owned by one customer are charged for on one bill. See "Mapping Multiple Phone Numbers to a Single Number".
- Use the "FCT_SocialNo" module to identify "social numbers" that are not displayed on an invoice. See "Setting Up Social Numbers".
- Use the "FCT_NOSP" module to identify a network operator and service provider when performing segment rating. See "Identifying the Network Operator/Service Provider".
- Use the "FCT_NumberPortability" module to process events correctly when a customer changes network provider but keeps the phone number. See "Setting Up Number Portability".
- Use the "FCT_PrefixDesc" module to specify how to identify the destination of a call based on the number prefix. See "Creating Call Destination Descriptions".

You set up EDR enrichment by configuring modules in a pipeline. For more information about enrichment modules, see "Setting Up EDR Enrichment".

About Service Mapping Modules

Incoming EDRs from multiple switches often use different codes to represent the same service or supplementary service. To process EDRs, you need to normalize that data by mapping external service codes to internal service codes, service classes, and usage classes. You use the following modules:

- Use the "FCT_ServiceCodeMap" module to map external service codes to internal service codes. For example, CDRs might use different codes for different types of telephony services. You can map those codes to a single code; for example, TEL. The service is typically a bearer or primary service.
- Use the "FCT_UsageClassMap" module to map external codes for supplementary services, such as call forwarding, to internal usage classes. Usage classes are typically used for rating based on quality of service, sub-services, or service-level agreement values.
- Use the "IRL_UsageType" iScript to assign usage types to EDRs. A usage type is an internal code that represents an attribute of a customer account; for example, a friends and family discount.
- Use the "FCT_UoM_Map" module to convert the unit of measurement (UoM) of an incoming EDR to a UoM needed for rating a particular service. For example, an EDR might include the usage amount in seconds, but the service is configured to charge by minutes. The FCT_UoM_Map module converts seconds to minutes, using rounding rules.

You configure service mapping by setting up mapping rules in Pricing Center and configuring mapping modules in a pipeline. For more information about service mapping, see "Setting up Pipeline Price List Data" in *BRM Setting Up Pricing and Rating*.

About Zoning Modules

Zoning function modules prepare EDRs for rating by identifying geographic or logical zones that are used for rating.

You can identify zones as follows:

- Using logical source and destination; for example, area codes, the service used, or retail or wholesale usage.
- Using geographic distance.

Zoning is performed by the following function modules:

- Use the "FCT_PreRating" module to calculate zones and impact categories.
- Use the "FCT_APN_Map" module to process APN data for zoning.
- Use the "FCT_USC_Map" module to refine impact categories based on service attributes.
- Use the "FCT_Zone" module to compute zones when you use a pipeline only for zoning.
- Use the "FCT_SegRateNoCust" module to find the segment using the source network information instead of using the customer information.

You set up zoning by creating zone models in Pricing Center and configuring zoning modules in a pipeline. For more information about zoning, see "Setting up Zones for Batch Pipeline Rating" in *BRM Setting Up Pricing and Rating*.

About Rating Modules

Rating modules rate EDRs. Each module performs a specific function; for example, the FCT_Dayrate module determines the time of day used for rating. Because each module adds or modifies data, the order of modules is important. For example, the FCT_RateAdjust module must process EDRs after the FCT_MainRating module. For more information, see "Function Module Dependencies".

- Use the "FCT_GlobalRating" module to apply a global rate plan to all EDRs.
- Use the "FCT_CustomerRating" module to provide customer data for other rating modules.
- Use the "FCT_SegRateNoCust" module to perform business analysis.
- Use the "FCT_RSC_Map" module to perform rating based on the quality of service when you set up service-level agreements (SLAs).
- Use the "FCT_RateAdjust" module to adjust charges after rating.
- Use the "FCT_BillingRecord" module to consolidate billing information for loading into the BRM database.

You set up rating by configuring rate plans in Pricing Center and configuring rating modules in a pipeline. For information about each module, and how pipeline rating works, see "Configuring Pipeline Rating".

About Discounting Modules

Discounting modules run after rating modules. Pipeline discounting uses the following processes:

- The module checks for the conditions that allow a discount; for example, using 100 minutes per month.
- The module grants the discount; for example, reducing the charge by 10%.

You set up discounting by configuring discount models in Pricing Center and configuring the following discounting modules in a pipeline:

- Use the "FCT_DiscountAnalysis" module to analyze EDRs before discounting.

- Use the "FCT_Discount" module to calculate and apply discounts.

About Roaming Modules

Roaming modules rate roaming usage. You set up roaming rates by using Pricing Center and configuring the following modules in a pipeline:

- Use the "FCT_CarrierIcRating" module to supply the interconnect rate plan for the FCT_MainRating module.

  For more information, see "About Rating Roaming Events" in *BRM Configuring Roaming in Pipeline Manager*.

About iScripts and iRules

Use iScripts and iRules to create custom pipeline functionality. For example, you can perform additional processing on data or add data to EDRs.

Use iScripts to perform the same operation on every EDR. Use iRules when you need to run an operation only under certain conditions.

Fore more information, see "Creating iScripts and iRules" in *BRM Developer's Guide*.

How Pipeline Manager Uses BRM Data

Pipeline Manager needs to get data from the BRM database to rate each account. For example, Pipeline Manager gets the rate plan to use from the services and products owned by an account. Pipeline Manager also gets historical data; for example, if a customer changes a phone number, Pipeline Manager needs the old number to rate calls made using it.

Pipeline Manager also needs to get data that is not required for rating but is required by the BRM event objects that Rated Event (RE) Loader loads into the BRM database. For example, every event requires an item, so Pipeline Manager needs to get the correct item for the event. When the rated event is loaded, the correct item is already recorded in the event.

Pipeline Manager also supplies additional data needed for rating or for creating a valid event. For example, Pipeline Manager supplies the G/L ID for the event (although the G/L IDs must match in the pipeline pricing data and real-time rating pricing data).

How Pipeline Manager Identifies Accounts

When Pipeline Manager rates usage events, there is no information in the original CDR that specifies which BRM account was responsible for the event. Pipeline Manager needs to know the account to rate the event and to apply discounts.

To find the account:

1. In the EDR, Pipeline Manager finds the phone number identified as the calling number.
2. In the data retrieved from BRM, this number is stored in the alias list in a service object. Once the service object is found, it has a pointer to the account object, from which Pipeline Manager identifies the account.

For information about configuring how to identify accounts in Pipeline Manager, see "Specifying Which Data is Used for Identifying Accounts" in *BRM Setting Up Pricing and Rating*.

How Pipeline Manager Chooses a Rate Plan

Pipeline Manager needs a rate plan for determining usage charges. There are three ways that Pipeline Manager can find out which rate plan to use:

- **Product priority**. By default, Pipeline Manager uses the rate plan associated with the highest-priority product. That is, it searches through all purchased products, from highest priority to lowest priority, until it finds one that matches the event's rating criteria, such as the zone model and time model type. It then selects the rate plan associated with that product. You assign priorities to products in your pipeline rate plan. See "About Creating a Price List" in *BRM Setting Up Pricing and Rating*.
- **Lowest charge**. When configured for least cost rating, Pipeline Manager finds the product that generates the lowest overall charge to the customer and then uses the rate plan associated with the product. You configure a pipeline for least cost rating by using the IRL_LeastCostPerEDR and ISC_LeastCost modules. See "About Least Cost Rating".
- **ERA**. If a service or an account is associated with an extended rating attribute (ERA), Pipeline Manager uses the rate plan you configure and prioritize for the ERA in Customer Center. See "About Extended Rating Attributes for Telco Services" in *BRM Telco Integration*.

> **Note:** If a subscription service and member service both own a service-level ERA, the member service's ERA has priority and is used for selecting the rate plan. See "Managing Customers' Subscription-Level Services" in *BRM Managing Customers*.

How Pipeline Manager Assigns Delayed Events to Items

When Pipeline Manager outputs events to RE Loader, the events must include all mandatory event data. Most of that data comes from the incoming CDR; for example, the call origin and destination. Some of the data must come from BRM, including which bill item stores the balance impact of the event.

In BRM, every event object is associated with a bill item. The incoming CDR does not have any information about bill items, so Pipeline Manager gets that information from BRM.

1. In the EDR, Pipeline Manager finds the phone number identified as the calling number.
2. In the data retrieved from BRM, this number is stored in the alias list in a service object. Once the service object is found, Pipeline Manager uses the information in the item POID list to determine which bill item the event applies to.

BRM creates service usage items for the next accounting cycle on four occasions:

- When the service is purchased by an account.
- When billing is run.
- When the Bill Now feature is used in Customer Center.
- When a usage event occurs.

However, BRM does not create a usage item when Pipeline Manager processes an event. If no usage item exists, Pipeline Manager rejects the event. Therefore, BRM pre-creates usage items for Pipeline Manager to use.

BRM pre-creates items in the following cases:

- When an account is created.
- When you run billing.
- When a CSR uses Bill Now.
- At the end of the accounting cycle when you use delayed billing.

> **Important:** To maintain correct discount balances, and to minimize rejected EDRs, always use delayed billing when you use pipeline rating. See "Setting up Delayed Billing" in *BRM Configuring and Running Billing*.

For information about setting up precreated items, see "Setting Up Service-Level Bill Items" in *BRM Installation Guide*.

To choose the correct bill item, Pipeline Manager can do one of the following:

- Assign the event to the current open bill item.
- Assign the event to the next open bill item.
- Reject the event because it belongs to an item that has already been included in a bill.

To decide which item to apply the bill to, Pipeline Manager takes into account the following dates:

- The date when the call occurred (the EDR date).
- The current system date.
- The date when the current accounting cycle ends. This is called the *next accounting cycle* date.
- The number of days after the current accounting cycle ends when delayed billing runs. This number is called the *delayed billing offset*. (For more information, see "Setting up Delayed Billing" in *BRM Configuring and Running Billing*.)

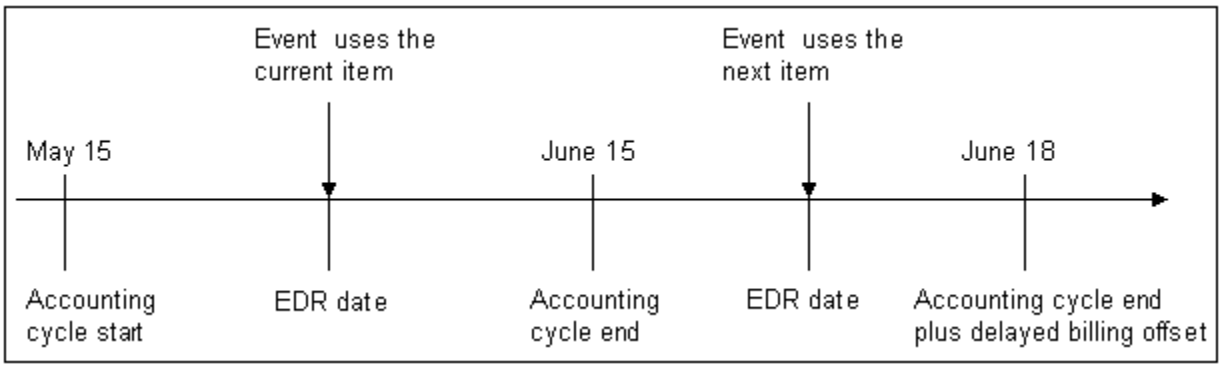
> **Note:** You can set up an accounting cycle delay period to manage which items delayed events are assigned to. This is useful when your billing cycle spans multiple accounting cycles. See "About Accounting Cycle Delay Periods".

To assign the event to an item:

- If the EDR date falls before the next accounting date, the event is assigned to the current item.
- If the EDR date falls after the next accounting date, the event is assigned to the next item. This can happen because the event might occur after the close of the accounting cycle but before the delayed billing offset date.

Figure 1–3 shows how events are assigned:

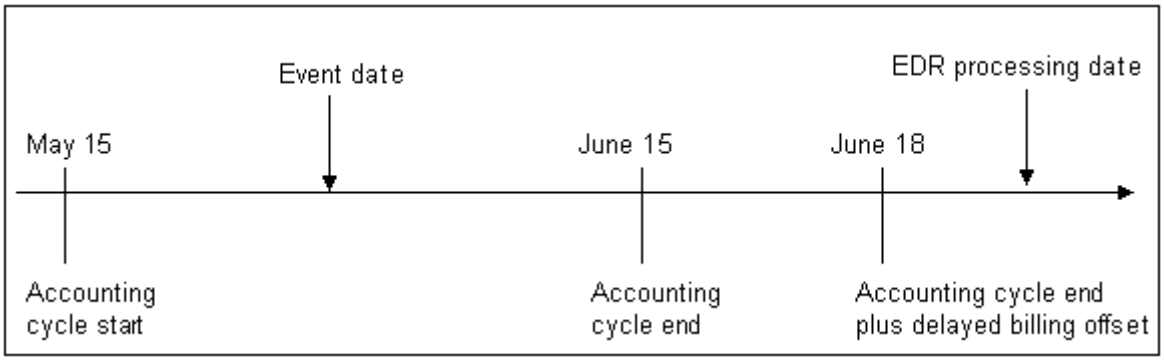
Figure 1–3 Event Assignment



Note: The customer billing date is not relevant when choosing which item to use for the event. There might be multiple accounting cycles in one billing cycle. New items are created for each accounting cycle.

If Pipeline Manager needs to assign an event to the current item, but billing for that item has already occurred, Pipeline Manager includes the event and assigns it to the current item. For example, if the event is rated on May 20 and loaded after the account cycle ends, it is still included in the current item as shown in Figure 1–4.

Figure 1–4 Current Event Assignment After Billing



About Accounting Cycle Delay Periods

In the batch pipeline, events that occur in one cycle can sometimes be rated and loaded into the BRM database after that cycle is completed. To assign delayed events to items of the billing cycle in which they occurred, you configure delayed billing. To assign delayed events to items of the accounting cycle in which they occurred, you configure an accounting cycle delay period.

When a billing cycle spans multiple accounting cycles, the items for those accounting cycles are not closed until billing is run. If you run a general ledger (G/L) report at the end of an accounting cycle for which billing has not yet been run, the status of the revenue in the G/L can change if additional events are rated and loaded for that cycle before the accounts are billed.

If you require that G/L data not change after the G/L report is run, you can configure an accounting cycle delay period after which events are no longer assigned to items of

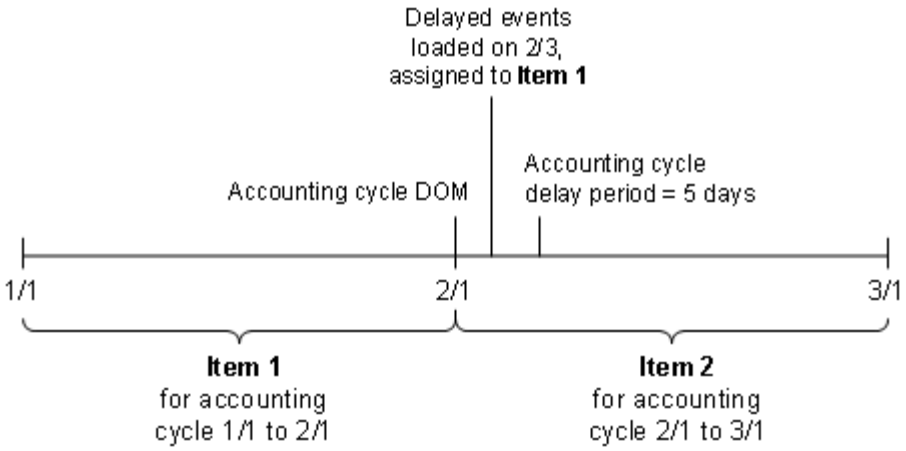
that accounting cycle, even if those items are not closed. You then run G/L reports after the accounting cycle delay period has ended. This ensures that the revenue reported in the G/L is accurately represented and that the state of the revenue (earned, unearned, billed, and unbilled) does not change after the G/L report is run.

Important: To use an accounting cycle delay period, you must also configure delayed billing. See "Setting up Delayed Billing" in *BRM Configuring and Running Billing*.

When you configure an accounting cycle delay period, BRM assigns delayed events to items based on when the accounting cycle delay period ends. BRM assigns events to items when RE Loader loads the events into the database:

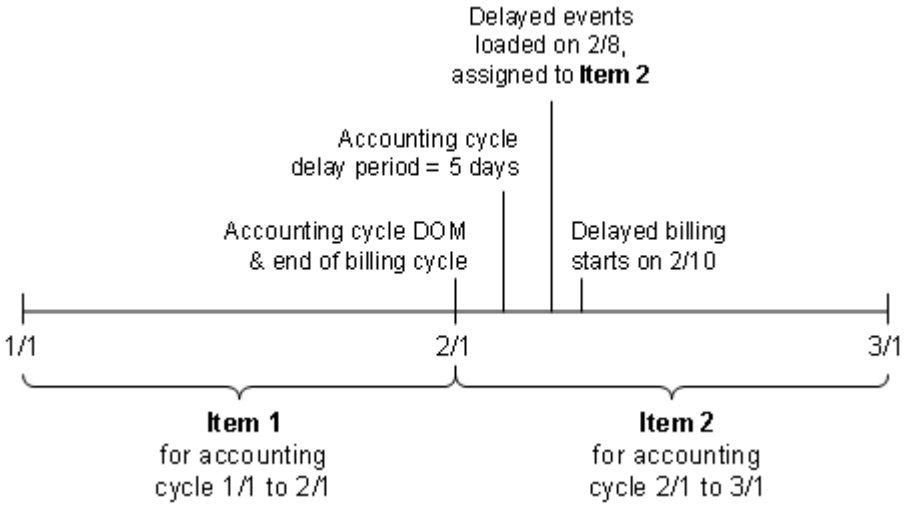
- When delayed events (events that occurred in the previous cycle) are loaded after the accounting day of month (DOM), but before the end of the accounting cycle delay period, those events are posted to the item for which the DOM has just passed as shown in Figure 1–5:

Figure 1–5 Delayed Events Arriving During Cycle Delay Period



- When the billing cycle has ended and delayed events are loaded after the end of the accounting cycle delay period, but before delayed billing is run, those events are posted to the item for the next (current) accounting cycle, even though the previous cycle has not been billed and its items are still pending. This is shown in Figure 1–6:

Figure 1–6 Delayed Events Arriving After Cycle Delay Period



- After the account is billed, items for the billed cycle are closed so delayed events are posted to the item for the following (the current) cycle.

> **Note:** If the accounting cycle delay period is longer than the delayed billing period, the accounting cycle delay period is ignored after billing is run. After billing is run, if any remaining events that occurred in the previous cycle are rated and loaded in the current cycle, they are assigned to the current cycle's item.

You specify the accounting cycle delay period by modifying a business parameter configuration (**/config/business_params** object). RE Loader checks the accounting cycle delay period in the **/config/business_params** object before loading events. See "Configuring an Accounting Cycle Delay Period".

Configuring an Accounting Cycle Delay Period

By default, an accounting cycle delay period is disabled. You can enable this feature by modifying a field in the **billing** instance of the **/config/business_params** object.

You modify the **/config/business_params** object by using the **pin_bus_params** utility. For information on this utility, see "pin_bus_params" in *BRM Developer's Guide*.

To configure an accounting cycle delay period:

1. Use the following command to create an editable XML file from the **billing** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```

   This command creates the XML file named **bus_params_billing.xml.out** in your working directory. The file contains the current billing configuration values in the **/config/business_params** object in the BRM database. If you don't want this file in your working directory, specify the path as part of the file name.

2. Open **bus_params_billing.xml.out** file and search for the following line:

   ```
   <AcctCycleDelayPeriod>-1</AcctCycleDelayPeriod>
   ```

3. Change **-1** to the number of days in the accounting cycle delay period. The number of days must be a positive value. (A value of **-1** indicates that there is no accounting cycle delay period.)

   For example, if the accounting cycle delay period is 3 days and the accounting cycle ends at midnight on March 31, the delay period ends at midnight on April 3.

   Caution: BRM uses the XML in this file to overwrite the existing **billing** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM billing configuration.

   Important: Do not set the accounting cycle delay period to be longer than the delayed billing period.

4. Save and close the **bus_params_billing.xml.out** file and rename the file to **bus_params_billing.xml**.
5. Use the following command to load this change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_billing.xml
   ```

   You should execute this command from the *BRM_Home*/**sys/data/config** directory, which includes support files used by the utility. *BRM_Home* is the directory where you installed BRM components. To execute it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.
6. Read the object by using the **testnap** utility or Object Browser to verify that all fields are correct:
   - For general instructions on using **testnap**, see "Using Testnap" in *BRM Developer's Guide*.
   - For information on how to use Object Browser, see "Reading objects by using Object Browser" in *BRM Developer's Guide*.
7. Stop and restart the Connection Manager (CM). See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

About G/L IDs

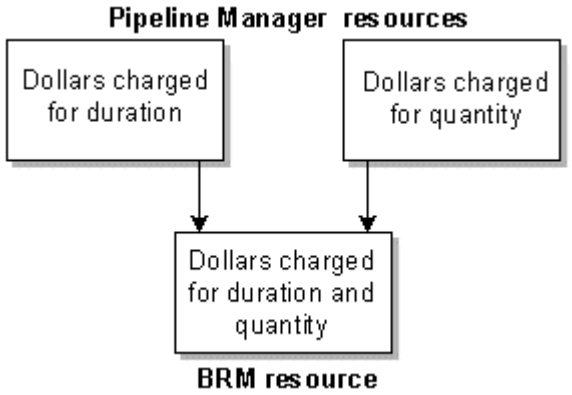
Pipeline Manager uses G/L IDs defined in the Pipeline Manager database. However, those G/L IDs must match the G/L IDs in the BRM database. You should define G/L IDs in the BRM database first and then in Pipeline Manager.

About Mapping Resources between the Pipeline Manager Database and the BRM Database

Resource IDs must match between the Pipeline Manager database and the BRM database. In addition, you can configure how Pipeline Manager resources map to BRM resources. For example, Pipeline Manager can separate resource amounts in a variety of ways, but you might want to combine resource amounts when defining BRM balance impacts.

Figure 1–7 shows how the charges for a GPRS event can be mapped between the Pipeline Manager database and the BRM database:

Figure 1–7 GPRS Charge Mapping Between BRM and the Pipeline Manager



How Pipeline Manager Gets Historical Data

Because there is a gap of time between when a call occurs and when it is rated, information about the customer can change during that time. For example, a customer might change the phone number before a call is rated. Pipeline Manager needs to look up account data based on the old number.

To retrieve historical information, Pipeline Manager gets data from audited objects. By default, auditing in BRM is turned off for most objects. After you install the Account Synchronization DM, you must run the **object_auditing.pl** script to turn on auditing for the objects and fields that Pipeline Manager needs data about. For more information, see "Turning on Object Auditing" in *BRM Installation Guide*.

In addition, Pipeline Manager pricing configuration data includes validity dates that can be used to apply the correct rating to delayed events.

About Loading Pipeline-Rated Event Data

Pipeline Manager sends the results of rating to output files. BRM loads the data in these files into the BRM database.

You configure BRM Batch Controller to start a batch handler when rated event files are ready for loading.

You configure a batch handler to load the rated-event data into the BRM database. The batch handler runs the utilities that load the data.

BRM provides the following sample batch handlers that you can use to load pipeline-rated events:

- **SampleRelHandler**: This batch handler starts the Rated Event (RE) Loader utility (pin_rel). The utility loads data for events that are rated in batches.
- **OODHandler**: This batch handler starts the rerate-request loader utility (pin_load_rerate_jobs). The utility creates rerate jobs for events that were rated out of order.
- **SampleHandler**: This batch handler starts the Universal Event (UE) Loader utility (uel). The utility loads batches of events into the BRM database for rating by the rating opcodes. UE Loader is also used by Pipeline Manager to load various types

of data files that it generates during rating; for example, validity period data for products, discounts, and resources that start on first usage, roaming settlement data, and revenue assurance data.

- **ConfigurableValidityHandler**: This batch handler starts the **pin_rel** and **pin_load_rerate_jobs** utilities, and the instance of **uel** that loads first-usage validity data, thereby eliminating the need to configure separate instances of the handlers for these utilities. For more information, see "About Using a Single Batch Handler to Run Multiple Loading Utilities".

About Using a Single Batch Handler to Run Multiple Loading Utilities

If you use Pipeline Manager to rate products, discounts, or granted resources that start on first usage, you configure the ConfigurableValidityHandler sample batch handler to run the utilities for RE Loader (pin_rel), UE Loader (uel), and out-of-order rerating (pin_load_rerate_jobs).

For information about validity periods that start on first usage, see "About Effective Periods That Start on First Usage" and "About Balance Impacts That Become Valid on First Usage" in *BRM Setting Up Pricing and Rating*.

Note: If you do not wish to use a single batch handler to run these utilities, you can configure the individual sample batch handlers provided for RE Loader, UE Loader, and out-of-order rerating.

ConfigurableValidityHandler runs **pin_rel**, **uel**, and **pin_load_rerate_jobs** instances sequentially, waiting for one process to complete before starting the next. In a single sequence, ConfigurableValidityHandler processes files that were produced for the same CDR file or for the same pipeline transaction (if the pipeline is configured to produce an output file per transaction instead of per CDR file).

When pipeline rating outputs a file of rated events, the ConfigurableValidityHandler batch handler performs the following tasks:

1. Runs **pin_rel** to load the results of pipeline rating into the BRM database.

   For more information about RE Loader, see "Understanding Rated Event Loader".

2. Checks for first-usage validity files for products and discounts and, if one exists for the same CDR or transaction, runs **uel** to load the product and discount validity data.

   For more information about using UE Loader to load validity data, see "About Updating Validity Period Information in the BRM Database".

3. Checks for first-usage files for resources and, if one exists for the same CDR or transaction, runs **uel** again to load the resource validity data.

4. Checks for rerate-request files for events rated out of order and, if one or more exist for the same CDR or transaction, runs **pin_load_rerate_jobs** for each file.

   For more information about rerating out-of-order events, see "About Automatic Rerating of Out-Of-Order Events" in *BRM Setting Up Pricing and Rating*.

ConfigurableValidityHandler handles errors in the following ways:

- If **pin_rel** does not successfully load the pipeline batch-rated events, the failure is logged in the handler log file (**configurable_validity_handler.log**) and **uel** and **pin_load_rerate_jobs** are not run.

- If **uel** or **pin_load_rerate_jobs** fails, the entire process is recorded as failed in the handler log file.

If all processes are successful, the **configurable_validity_handler.log** file is deleted. If one or more processes are not successful, the **configurable_validity_handler.log** file is not deleted.

Along with processing information and status, the log files include the name of the input file that was loaded. If **pin_rel** loads a rated-event file and there were no associated first-usage validity files or rerate-request files, this is noted in the log file.

About Pipeline Rating and BRM Billing

When Pipeline Manager receives an EDR for the next billing cycle for an account that hasn't yet been billed, the EDR is suspended (not rated). Only when the account's billing process is complete can the new EDRs be rated.

The number of accounts being billed affects the time it takes to complete the billing process. The longer the processing time, the greater the chance EDRs might need suspending or rerating. If you process many EDRs and need accounts to be billed quickly so that their new usage can be rated, you can set up Pipeline Manager to trigger billing. When Pipeline Manager triggers billing for an account, it is billed in a separate billing process.

For more information, see "Setting up Pipeline-Triggered Billing" in *BRM Configuring and Running Billing*.

Function Module Dependencies

Table 1–1 provides guidelines for how to configure the order of function modules in a pipeline. The modules are listed in a typical order, but your configuration might vary. Some modules require that other modules be run first, whereas some modules can be located anywhere in the pipeline.

For more information, see the reference documentation for the module.

Table 1–1 Functional Modules and Processing Dependencies

Function module	Processing dependencies
FCT_PreSuspense	Must be the first preprocessing module in a pipeline.
ISC_SetAndValidateBatchInfo	This is the first iScript that must be used so that the batch ID gets inserted before any further processing of the mediation batches.
FCT_DuplicateCheck	Should run early in a pipeline to discard duplicate EDRs.
FCT_CallAssembling	Should run early in a pipeline to assemble EDRs. Must run before FCT_Discard.
FCT_ServiceCodeMap	Some modules require an internal service code, so this module should run near the front of a pipeline.
ISC_CiberInputValidation	Because erroneous CIBER records can be discarded, this module must run before the FCT_Discard module.
FCT_Discard	Because you can discard or split EDRs based on service codes, this module should run after the FCT_ ServiceCodeMap module. Should be early in the function pool, but must be run after FCT_ CallAssembling.

Table 1–1 (Cont.) Functional Modules and Processing Dependencies

Function module	Processing dependencies
iRuleValidation	Run this iRule before ISC_TapSplitting.
ISC_TapSplitting	Must run after the following modules: ■ FCT_DuplicateCheck ■ FCT_Discard
FCT_AccountRouter	For general use, this module must run after the FCT_ServiceCodeMap module and before the rating modules. For use with standard recycling or Suspense Manager using multiple database schemas, this module must run before OUT_GenericStream in a pre-recycling pipeline. This module sends output to a separate pipeline for each BRM database schema.
FCT_EnhancedSplitting	Because you can split EDRs based on service codes, this module should run after the FCT_ServiceCodeMap and FCT_UsageClassMap modules.
FCT_CliMapping	Must run before the rating modules.
FCT_UoM_Map	Must run after the FCT_ServiceCodeMap module and before the rating modules.
FCT_UsageClassMap	Must run before the zoning and rating modules.
FCT_NumberPortability	Must run before the zoning and rating modules.
ISC_MapNetworkOperatorInfo	Must run after the FCT_NumberPortability module and the ISC_PopulateOpcodeAndUtilBlock_Diameter iScript.
FCT_NOSP	Must run before segment rating is performed.
FCT_Account	Must run before the zoning and rating modules.
ISC_ProfileAnalyzer	Must run after FCT_Account and before any rating modules.
ISC_ProfileLabel	Must run after FCT_Account and before any rating modules.
IRL_UsageType	Must run after FCT_Account and before FCT_USC_Map.
FCT_CiberOcc	Must run after the FCT_DuplicateCheck module and before the FCT_CarrierIcRating module.
FCT_ItemAssign	Must run after the FCT_Account, rating, and discounting modules and before the FCT_BillingRecord module.
FCT_CustomerRating	Must run after FCT_Account.
FCT_Filter_Set	Must run after FCT_Account.
IRL_PromotionalSavingPerEDR	Must run before IRL_LeastCost and FCT_CustomerRating.
IRL_LeastCostPerEDR	Must run before FCT_CustomerRating and *after* "FCT_Filter_Set".
FCT_CustomerRating	Must run after FCT_Account.
ISC_LeastCost	Must run after FCT_CustomerRating.

Table 1–1 (Cont.) Functional Modules and Processing Dependencies

Function module	Processing dependencies
FCT_CiberOcc	Must run after the FCT_DuplicateCheck module and before the FCT_CarrierIcRating module.
FCT_CarrierIcRating	Must run after FCT_Account.
FCT_SegRateNoCust	Must run after FCT_Account.
FCT_DroppedCall	Must run after FCT_Account.
FCT_APN_Map	Can run before or after the zoning modules (FCT_Zone and FCT_PreRating). See "Setting up APN Mapping" in *BRM Setting Up Pricing and Rating*.
FCT_PreRating	Must run before the FCT_MainRating module.
FCT_SegZoneNoCust	Must run before the FCT_MainZoning module.
FCT_MainZoning	Must run after the FCT_SegZoneNoCust module.
FCT_Zone	Must run after FCT_Account.
FCT_USC_Map	Must run after the following: ■ FCT_UsageClassMap ■ ISC_UsageType ■ FCT_PreRating
FCT_TriggerBill	Must run before the FCT_MainRating module.
FCT_MainRating	Must run after at least one of the following modules: ■ FCT_GlobalRating ■ FCT_CustomerRating ■ FCT_SegRateNoCust ■ FCT_CarrierIcRating
FCT_RSC_Map	Must run after the FCT_MainRating module to adjust the rate.
FCT_Dayrate	Must run after the FCT_MainRating module to adjust the rate.
FCT_RateAdjust	Must run after the FCT_MainRating module to adjust the rate.
FCT_Rounding	Must run after the FCT_RateAdjust module if you want rating results to be rounded and after FCT_Discount module if you want discount results to be rounded. FCT_Rounding must come after each module for which rounding should occur. For batch rating, it must come before the FCT_ApplyBalance module.
FCT_ExchangeRate	Must run after the FCT_MainRating module.
ISC_FirstProductRealtime	Must run this iScript in the real-time rerating pipeline before the FCT_DiscountAnalysis module.
FCT_DiscountAnalysis	For pipeline rating, must run after the FCT_Account module and before the FCT_Discount module. For real-time rating, must run before the FCT_Discount module.

Table 1–1 (Cont.) Functional Modules and Processing Dependencies

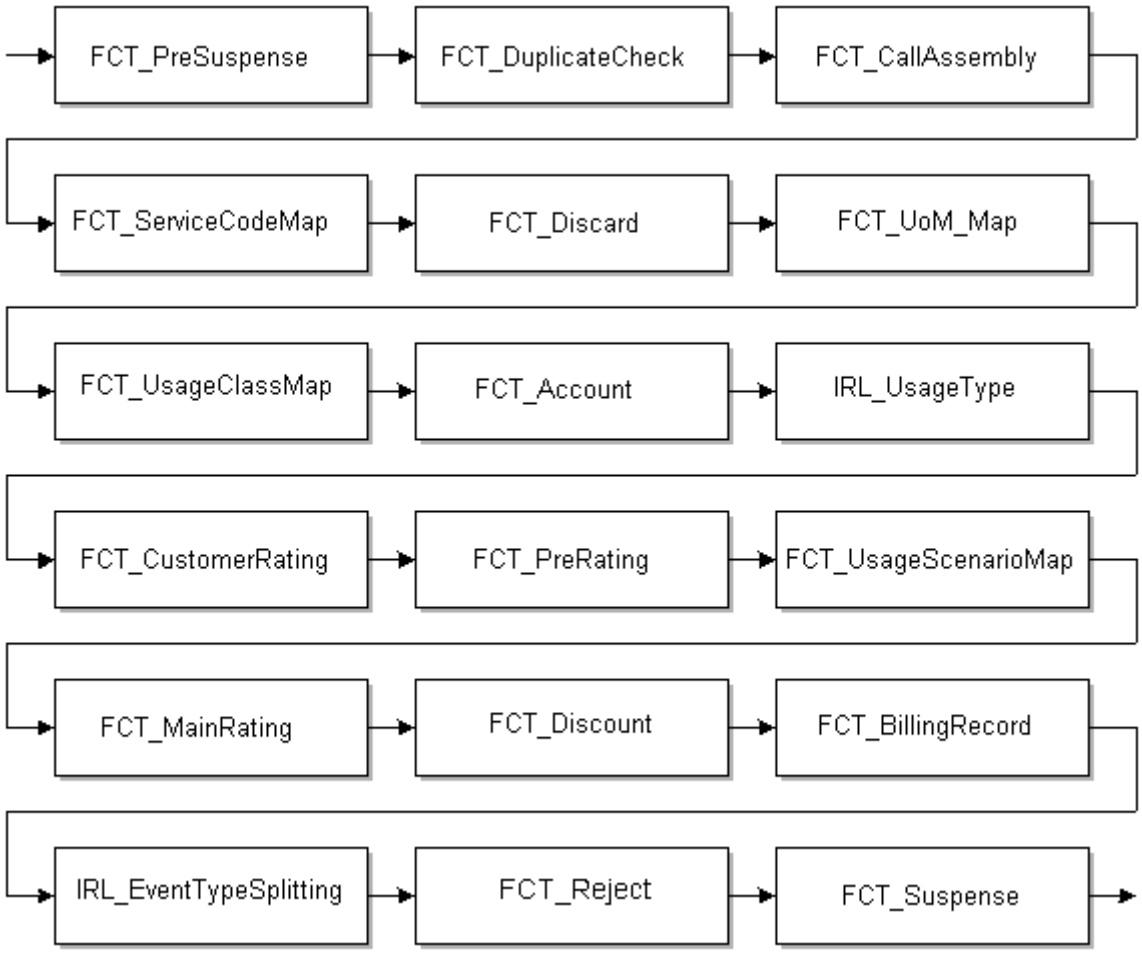
Function module	Processing dependencies
FCT_Discount	Must run after the FCT_MainRating module. For batch rating, this module should be in the same function pool as the FCT_ApplyBalance module and run before that module.
FCT_FirstUsageNotify	Must run this module before the FCT_ApplyBalance and FCT_Reject modules.
FCT_ApplyBalance	Must run after the FCT_Rounding module. This module should be in the same function pool as the FCT_Discount module and run after that module.
ISC_TaxCalc	Must Run this module before the FCT_BillingRecord module, but after the FCT_MainRating module.
FCT_BillingRecord	Must run after the FCT_MainRating and FCT_Discount modules.
FCT_EventOrder	Must run *after* the FCT_MainRating and FCT_Discount modules and *before* the FCT_Reject module.
ISC_CiberOutputMapping	Must run after the FCT_MainRating module and ISC_PostRating iScript.
IRL_EventTypeSplitting	Must run after FCT_ServiceCodeMap and before the FCT_Reject module. This is typically the last module before the FCT_Reject module.
FCT_Reject	Runs after the rating and discount modules.
ISC_SetOutputStream	Must run after FCT_Reject.
ISC_PostRating	Must run: ■ After rating modules FCT_CustomerRating, FCT_PreRating, and FCT_MainRating or ■ After the FCT_ExchangeRate module
ISC_SetEDRStatus	Must run before FCT_AggreGate.
ISC_SetRevenueFigures	Must run after rating and discounting and before FCT_AggreGate.
ISC_SetRevenueStream	Must run before FCT_AggreGate and after post rating (after the EDRs are rated).
ISC_SetEDRStatus	Must be run before the FCT_AggreGate scenario that collects audit data grouped on the EDRStatus field.
FCT_AggreGate	Runs after rating modules. This module usually runs in its own pipeline.
FCT_CancelTimer	Depends on the FCT_Timer in the Dispatcher pipeline for the TimerID and the timeout flag values.
FCT_Recycle	For pipeline-only recycling, must be the last module in the pipeline.
FCT_Suspense	For standard recycling and Suspense Manager, must be the last module in a pipeline.

The following modules do not have any placement dependencies and can be run anywhere in a pipeline or in a separate pipeline, depending on the data that is being processed:

- FCT_IRules
- FCT_IScript
- ISC_AddCBD
- IRL_PipelineSplitting
- FCT_Timer
- FCT_PreRecycle
- FCT_Opcode
- ISC_ApplyTax
- ISC_BACKOUTTypeSplitting
- ISC_CiberRejectReason
- ISC_ConsolidatedCP
- ISC_DupRAPRecords
- ISC_EDRToTAPOUTMap
- ISC_Migration
- ISC_MiscOutcollect
- ISC_ObjectCacheTypeOutputSplitter
- ISC_OverrideSuspenseParams
- ISC_RemoveASSCBD
- ISC_RollbackSettlement
- ISC_UsageClassSetting
- FCT_SocialNo
- FCT_PrefixDesc

Figure 1–8 shows the order of the most common function modules:

Figure 1–8 Order of Common Function Modules



Data Module Dependencies

This section provides guidelines for how to configure the order of data modules in the registry file. Some modules require that other modules be run first, whereas some modules can be located anywhere in the pipeline.

- "DAT_PortalConfig". Due to the dependency of other data modules on DAT_ PortalConfig, the DAT_PortalConfig registry entries must appear before all other data module entries in the registry file.

2

Configuring Pipeline Rating

This chapter describes how to configure the Oracle Communications Billing and Revenue Management (BRM) pricing data and modules necessary for batch rating with Pipeline Manager. It includes information about using Pricing Center to create rate plans and price models as well as information about the function modules used during rating.

Before you read this document, you should be familiar with these topics:

- Pipeline rating. See "About Pipeline Rating".
- Real-time rating. See "About Real-Time Rate Plans" in *BRM Setting Up Pricing and Rating*.

About Configuring Pipeline Rating

Configuring pipeline rating involves two sets of tasks:

- Creating rate plans, price models, and other data using Pricing Center. See "Creating Pipeline Rate Plans and Price Models" in *BRM Setting Up Pricing and Rating*.
- Configuring rating function modules. See "About Configuring Function Modules for Pipeline Rating".

About Configuring Function Modules for Pipeline Rating

Three groups of modules are involved in rating:

- The pre-rating modules (FCT_ GlobalRating, FCT_CustomerRating, FCT_ SegRateNoCust, and FCT_RSC_Map) gather information and prepare the event data record (EDR) for rating.
- The FCT_MainRating module applies rate plans and price models to the EDR, creating charge breakdown data, including charge packets with charges.
- The post-rating modules (FCT_RateAdjust and FCT_BillingRecord) make changes to the EDR after the charge data is included.

Multiple Pipeline Manager modules add and delete charge packet blocks to EDRs. Therefore, the value in the NUMBER_OF_CHARGE_PACKETS field does not reflect the actual number of charge packets in the charge breakdown record. You can get the correct number of charge packets by using the edrNumDatablocks function in a custom iScript module.

Depending on how your pipelines are set up, you need to configure some or all of these modules for rating.

About the Rating Data Modules

Configure the following data modules for rating:

- DAT_AccountBatch
- DAT_BalanceBatch
- DAT_Calendar
- DAT_Currency
- DAT_Dayrate
- DAT_ExchangeRate
- DAT_ItemAssign
- DAT_ModelSelector
- DAT_NOSP
- DAT_PriceModel
- DAT_Rateplan
- DAT_TimeModel

About Using Filter Sets to Apply System Products and Discounts

Use filter sets to apply system products and system discounts to a select group of customers. For example, you can provide reduced international call rates for all customers with great credit. You define the criteria to qualify for a filter set and the list of available products and discounts by using Pricing Center.

When configured to use filter sets, the pipeline:

1. Analyzes each EDR to determine whether it meets the criteria for a filter set.
2. Adds any applicable system products and system discounts, along with their priorities, to the EDR's list of purchased products.
3. Uses the product and/or discount when rating the EDR.

> **Note:** The actual product or discount the pipeline uses to rate the EDR depends on how your system is configured. See "How Pipeline Manager Chooses a Rate Plan" in *BRM Managing Customers*.

You use the following to configure your system for filter sets:

- **system_filterset_edr_field_values**. This file specifies which EDR fields and values can be used as filter criteria. You must load this file into the BRM database prior to creating your filter sets in Pricing Center.
- **IRL_UsageType**. This iRule assigns usage types to an EDR. This signals the pipeline that the EDR qualifies for the special consideration that a filter set contains.
- **FCT_Filter_Set**. This module determines whether an EDR qualifies for any filter sets, and, if it does, adds any applicable system products and system discounts to the EDR's list of purchased products.

To configure BRM to use filters sets, perform the following tasks:

1. Specify which EDR fields and values can be used as filter criteria and then load the data into the BRM database. See "Loading Filter Set Data into BRM".
2. Define your filter sets by using Pricing Center. See "Defining Your Filter Sets".
3. Configure the IRL_UsageType iRule to assign usage types to EDRs. See "Configuring the IRL_UsageType iRule for Filter Sets" in *BRM Setting Up Pricing and Rating*.
4. Configure FCT_Filter_Set to apply system products and discounts to specified market segments. See "FCT_Filter_Set".
5. Connect the FCT_DiscountAnalysis module to the FCT_Filter_Set module. Use the FCT_DiscountAnalysis module's **Filter_SetModule** registry entry. See "FCT_DiscountAnalysis".

Loading Filter Set Data into BRM

You can specify the EDR fields and values that you can use as filter criteria. To do so, you edit the **system_filterset_edr_field_values.en_US** sample file in the *BRM_Home*/**sys/msgs/system_filter_set** directory. *BRM_Home* is the directory where you installed BRM components. For example, to use location as filter criteria, add these entries to the file:

```
DETAIL.ASS_CAMEL_EXT.MSC_ADDRESS="London"
DETAIL.ASS_CAMEL_EXT.MSC_ADDRESS="Paris"
```

After defining the field values, you use the load_localized_strings utility to load the contents of the **system_filterset_edr_field_values.***locale* file into the **/strings** object. To run the **load_localized_strings** utility, use this command:

```
load_localized_strings system_filterset_edr_field_values.locale
```

Note: If you are loading a localized version of this file, use the correct file extension for your locale. For a list of file extensions, see "Locale Names" in *BRM Developer's Guide*.

When you finish loading the file, confirm that pipeline rating is enabled in Pricing Center. In the **C:\Program Files\Portal Software\PricingCenter\lib\custom.properties** file, make sure the following entry is set to **True**:

```
pricingcenter.pipeline.rating=True
```

For information on loading the **system_filterset_edr_field_values.***locale* file, see "Loading Localized or Customized Strings" in *BRM Developer's Guide*. For information on creating new strings for this file, see "Creating New Strings and Customizing Existing Strings" in *BRM Developer's Guide*.

Defining Your Filter Sets

You define your filter sets by using Pricing Center. BRM then stores data about each filter set in **/filter_set/product** objects in the BRM database.

Note: To create and manage filter sets by using a custom client application, configure your application to call the filter set opcodes. See "Managing Filter Sets" in *BRM Developer's Guide*.

To create a filter set, perform the following in Pricing Center:

1. Create your system products and system discounts and associate them with specific service and event types. For example:
   - Create a system product for **/service/telco/gsm/telephony** events that charges a flat rate of 5 cents per minute.
   - Create a system discount for **/service/telco/gsm/sms** events that provides a 20% discount for the first 10 minutes of usage.
2. Create your filter sets by mapping filter criteria to system products and system discounts. When creating filter sets, you specify:
   - The conditions required for an EDR to qualify for a system product or discount. That is, the list of required EDR fields and values for each filter set.
   - Applicable system products and discounts, along with their priority and validity dates.

   In Table 2–1, Filter_1 specifies that all GSM wireless calls made to Paris by customers with great credit qualify for a 20% discount.

Table 2–1 Example Filters

Filter name	Filter criteria	System product or system discount	Priority
Filter_1	DETAIL.ASS_GSMW_EXT.RECORD_TYPE="Great credit" DETAIL.ASS_CAMEL_EXT.MSC_ADDRESS="Paris"	20% Discount	3
Filter_2	DETAIL.ASS_GSMW_EXT.RECORD_TYPE="Great credit"	Flat rate product	6

For detailed information on how to create filter sets, see Pricing Center Help.

About Global Rating

Use global rating to rate every EDR by using the same set of rate plans. Global rating is performed by the FCT_ GlobalRating module. This module adds an associated charge breakdown record to the EDR. Each charge, or partial charge, from the global rate plans adds a charge packet.

Global rating is typically used for gathering data used for business planning. For example:

- You can use global rating to calculate an average wholesale charge. You can then compare the average charge with the actual charge to find profit margins for different customer types.
- You can use global rating to calculate an average retail charge. You can use this data to monitor unusual charge amounts that can indicate an error in a rating configuration.

To configure global rating, see "FCT_CustomerRating". Use the **EdrRatePlans** registry entry to specify the global rate plans. You can use this entry in a semaphore.

About Least Cost Rating

Use least cost rating to rate EDRs by using the product that produces the lowest charge to customers. In this configuration, the pipeline:

1. Rates EDRs by using all products and rate plans associated with an EDR and applies any discounts.
2. Finds the rate plan and discount that produces the lowest charge.
3. Applies the lowest charge to the customer's balance impact.

> **Note:** An EDR can qualify for *either* least cost rating or overlay promotion. It cannot qualify for both. See "About Overlay Promotions"

You use the following modules to find the lowest possible charge for an event:

- The IRL_LeastCostPerEDR iRule screens EDRs for least cost rating. It compares each EDR against conditions that you specify. When an EDR meets the criteria, the module flags the EDR for least cost rating.
- The FCT_CustomerRating module checks whether an EDR is flagged for least cost rating, and, if it is, generates breakdown records and associated charge packets for each product. All charge packets are listed in order of priority, highest priority first.
- The ISC_LeastCost iScript calculates the total charge for each product and discount and flags the rate plan that produces the lowest charge. When the lowest charge is generated by a promotional product, the module also calculates the total savings between the promotional product and the lowest priority (base) product.

You configure the modules in a discounting pipeline that includes the FCT_DiscountAnalysis module and the FCT_Discount module. These modules find any applicable discounts for each rate plan and calculate the discount for each charge packet.

Configuring Least Cost Rating

To set up least cost rating:

1. Specify your least cost rating criteria. See "Specifying the Rules to Qualify for Least Cost Rating". To do so, you edit and configure the IRL_LeastCostPerEDR iRule. See "IRL_LeastCostPerEDR".
2. Configure FCT_CustomerRating for least cost rating by using the **LeastCostRating** entry. See "FCT_CustomerRating".
3. Configure the ISC_LeastCost iScript to find the lowest charge for customers. See "ISC_LeastCost".

Specifying the Rules to Qualify for Least Cost Rating

To specify which EDRs are flagged for least cost rating:

1. Edit the *Pipeline_Home*/**iScriptLib/iScriptLib_Standard/IRL_LeastCostPerEDR.irl** file to include your rules for least cost rating. *Pipeline_Home* is the directory where you installed Pipeline Manager. The variables in the file correspond to positions in the **IRL_LeastCostPerEDR.data** file.

For example, this rule specifies that EDRs matching the conditions in position 1 will have their DETAIL.CUST_A.LEAST_COST EDR field set to the value in position 2 of the **IRL_LeastCostPerEDR.data** file:

```
CONDITION:

SetLeastCostDefault ();
${1};

RESULT:
edrLong(DETAIL.CUST_A.LEAST_COST) = ${2};
```

2. Edit the *Pipeline_Home*/**iScriptLib/iScriptLib_Standard/IRL_LeastCostPerEDR.data** file to include the conditions required for least cost rating. Use Boolean operators to specify the combinations of market segments and product priorities for testing whether EDRs qualify for least cost rating. You can create any number of entries.

   For example, this rule specifies that only EDRs that meet both of these criteria are flagged for least cost rating (DETAIL.CUST_A.LEAST_COST set to 2):

   - MARKET_SEGMENT EDR field = 1234
   - Product priority greater than 4

   ```
   segmentContains("1234") and priority()>4;2
   ```

3. Configure the FCT_IRules module to point to **IRL_LeastCostPerEDR.irl** by using the following registry entries:

   Important: If you configure a pipeline to use filter sets, make sure IRL_LeastCostPerEDR runs *after* FCT_Filter_Set.

   See "IRL_LeastCostPerEDR".

About Calculating the Promotional Savings

You can calculate how much money your customers save when an event is rated with a promotional product rather than a base product by using promotional savings. *Promotional savings* allows you to calculate the difference in charges between a promotional product and the base product, so you can advertise the savings to your customers.

When configured for promotional savings, the pipeline:

1. Rates EDRs by using the highest priority product and the lowest priority (base) product.
2. Calculates the difference between the charge for the promotional product and the charge for the base product.
3. Applies the savings amount to the EDR.

   Note: The pipeline applies the savings amount only when the promotional product generates the lowest charge.

You use the following modules to calculate promotional savings:

- The IRL_PromotionalSavingPerEDR iRule screens EDRs for promotional savings. It compares each EDR against your specified criteria. When an EDR meets the criteria, the module flags the EDR for promotional savings.
- The FCT_CustomerRating module checks whether an EDR is flagged for promotional savings, and, if it is, generates separate charge breakdown records and associated charge packets for the highest priority (promotional) and lowest priority (base) product.
- The ISC_LeastCost iScript calculates the charges for the highest and lowest priority products and calculates the total savings amount.

See "How Pipeline Modules Process Overlay Promotions" for information about how the processing of overlay promotions and promotional savings are related.

To set up promotional savings:

1. Specify your promotional savings criteria. See "Specifying the Rules to Qualify for Promotional Savings".
2. Configure FCT_CustomerRating for promotional savings by using the **PromotionalSaving** entry. See "FCT_CustomerRating".
3. Configure ISC_LeastCost to calculate the promotional savings amount. See "ISC_LeastCost".

Specifying the Rules to Qualify for Promotional Savings

To specify which EDRs are flagged for a promotional savings calculation:

1. Edit the *Pipeline_Home*/**iScriptLib/iScriptLib_Standard/IRL_PromotionalSavingPerEDR.irl** file to include your rules for qualifying for a promotional savings calculation. The variables in the file correspond to positions in the **IRL_PromotionalSavingPerEDR.data** file.

   For example, this rule specifies that EDRs matching the conditions in position 1 will have their DETAIL.CUST_A.PROMOTIONAL_SAVING EDR field set to the value in position 2:

   ```
   CONDITION:

   ${1};

   RESULT:
   edrLong( DETAIL.CUST_A.PROMOTIONAL_SAVING ) = ${2};
   ```

2. Edit the *Pipeline_Home*/**iScriptLib/iScriptLib_Standard/IRL_PromotionalSavingPerEDR.data** file to include your conditions for qualifying for a promotional savings calculation. Use Boolean operators to specify the required combination of filter sets and product priorities. You can create any number of entries.

   For example:

   ```
   productName("Standard GSM Telephony", "Highest") and priority("Highest") > 4
   and usageStartTimeGreaterThan("20040101000000", "Highest") and
   serviceType("/service/telco/gsm/telephony", "Highest"); 2

   productName("Standard GSM Telephony", "Base") and priority("Base") < 4 and
   usageStartTimeGreaterThan("20040101000000", "Base") and
   serviceType("/service/telco/gsm/telephony", "Base"); 2
   ```

3. Configure the FCT_IRules module to run the IRL_PromotionalSavingPerEDR iRule. Use the PromotionalSaving entry to specify the path to your IRL_PromotionalSavingPerEDR iRules file.

   For more information, see "IRL_PromotionalSavingPerEDR".

About Overlay Promotions

Overlay promotions allow you to quickly and easily replace your existing products and discounts with special products and discounts that take precedence. For example, you may want to offer a special 10% discount for three months to your existing customers who have paid all their bills on time. To do this, you add a product to your price list that offers the same services but costs10% less and has a higher priority than the original product.

You do not need to create new zone models for promotional products. Promotional products can use the same zone models as standard products. For example, you can create two promotional products, CALL_USA and CALL_INDIA. You can give CALL_USA priority 4 and CALL_INDIA priority 2, while your base product has priority 0. There is no need to create special zone models for the promotional products. If a customer buys the promotional products, BRM uses them to rate calls to the US and India. Calls to other countries will be rated using the base product.

You create overlay promotions by using Pricing Center to create new products associated with the same service and event types as other products, but with higher product priorities. For details on creating products, see Pricing Center Help.

The capability to use overlay promotions is built into a pipeline, which:

- Recognizes and uses product priorities.
- Allows you to associate multiple products with the same service and event.
- Allows you to rate calls using overlay promotions only.

> **Note:** The overlay promotions feature interacts with least cost rating and promotional savings. An EDR can qualify for *either* least cost rating or overlay promotion. It cannot qualify for both.

How Pipeline Modules Process Overlay Promotions

The overlay promotions feature requires no special configuration of pipeline modules. The following modules are involved in the processing of overlay promotions:

- **FCT_CustomerRating**. The module creates a list of associated charge breakdowns and corresponding charge packets ordered from highest to lowest priority. If two products have the same priority, the product with the earliest start time is given a higher priority.

  The module then determines whether an EDR qualifies for least cost rating or overlay promotions. If the EDR qualifies for overlay promotions, the module further checks whether the EDR qualifies for promotional savings.

> **Note:** An EDR can qualify for *either* least cost rating or overlay promotion. It cannot qualify for both. If the EDR qualifies for overlay promotion, it may also qualify for promotional savings.

If the EDR is eligible for least cost rating, FCT_CustomerRating performs these tasks:

- Enables least cost rating by setting the DETAIL.CUST_A.LEAST_COST field to **2**.
- Disables promotional savings and overlay promotion by setting the DETAIL.CUST_A.PROMOTIONAL_SAVING and DETAIL.CUST_A.PROMOTION fields to **1**.

If the EDR is eligible for overlay promotions and promotional savings, FCT_CustomerRating performs these tasks:

- Enables promotional savings by setting the DETAIL.CUST_A.PROMOTIONAL_SAVING field to **2**.
- Disables overlay promotions and least cost rating by setting the DETAIL.CUST_A.PROMOTION and DETAIL.CUST_A.LEAST_COST fields to **1**.

If the EDR qualifies for overlay promotions but not promotional savings, FCT_CustomerRating performs these tasks:

- Enables overlay promotions by setting the DETAIL.CUST_A.PROMOTION field to **2.**
- Disables least cost rating and promotional savings by setting the DETAIL.CUST_A.LEAST_COST and DETAIL.CUST_A.PROMOTIONAL_SAVING fields to **1**.

- **FCT_PreRating**. The module populates the zoning information for each charge breakdown record in the EDR:
  - If least cost rating is used, the module finds the zoning information for all charge breakdown records. See "About Least Cost Rating".
  - If promotional savings or overlay promotion is used, the module finds the zoning information for at least one charge breakdown record. See "About Overlay Promotions".
- **FCT_MainRating**. The module functions differently depending on whether the EDR is rated in least cost rating, overlay promotion, or promotional savings mode:
  - **Least cost mode**. The module rates all charge breakdown records. If it fails to rate any charge breakdown record, it returns an error. See "About Least Cost Rating".
  - **Overlay promotion mode**. The module tries to rate charge breakdown records starting with the highest priority record. If rating is successful for a charge breakdown record, the module selects the rate plan of that record and populates the INTERN_FOUND_PP_INDEX field with the index of that rate plan. The module then deletes all other charge breakdown records from the container.
  - **Promotional savings mode**. The module tries to rate charge breakdown records, starting with the highest priority record. If rating is successful for a charge breakdown record, the module selects the rate plan of that record and populates the INTERN_FOUND_PP_INDEX field with the index of that rate plan. The module continues rating charge breakdown records until it reaches the last record. It then checks for and selects the last successfully rated record. The module keeps the first and last successful charge breakdown records in the EDR container but deletes all others. See "About Calculating the Promotional Savings".

Creating an Overlay Promotion

You use Pricing Center to create an overlay promotion in the same way you create any other product, except that you assign it a higher priority than the product it is supplanting. You then add the promotional product to an add-on plan or deal.

When you make the new overlay promotion available to your customers and they purchase the promotion, it takes priority over your other products.

You can create multiple overlay promotion products. For example, you could create an overlay product for calls to the US and another for calls to India.

About Rating with Products and Discounts Whose Validity Starts on First Usage

The effective period of an account's products and discounts can start when the products and discounts are first used to rate a subscriber's usage.

For more information, see "About Effective Periods That Start on First Usage" in *BRM Setting Up Pricing and Rating*.

If products or discounts are configured to start on first usage, you set up pipeline rating to set their validity periods when first usage occurs.

> **Note:** Resources that are granted by products and discounts can also start on first usage. To set resource validity when first usage occurs, you configure the batch discounting pipeline. See "About settIng the Validity of Resources Impacted by Discounts". For information about resources that start on first usage, see "About Balance Impacts That Become Valid on First Usage" in *BRM Setting Up Pricing and Rating*.

When configured to set validity on first usage, the pipeline:

1. Suspends EDRs that use products or discounts configured to start on first usage until the product or discount validity periods are set.

   EDRs are not suspended when first-usage resource validity is set because the pipeline calculates and stores the validity period in memory. The pipeline uses the stored validity period if it needs to consume resources for any other events before the balance validity period has been set in the database.

2. Sends the product or discount information to a file in the output stream.

   BRM processes the file, sets the validity periods in the database, and charges any applicable purchase and cycle fees. While their validity periods are being set, the products and discounts are locked.

3. If the event is discounted and the discount balance impact consumes a resource balance that starts on first usage, the pipeline sends the resource information to a file in the output stream.

   BRM processes the file and sets the resource validity periods in the database. If the validity period of all first-usage resources are configured to be synchronized, BRM also sets the validity period of those resources.

4. Rates the events when the EDRs are recycled.

If the pipeline transaction is rolled back or canceled, the validity period of any product or discount that was set in the transaction is unset.

BRM uses the Account Synchronization Data Manager (DM) to synchronize product and discount validity periods in the BRM database and pipeline memory.

The following modules are involved in setting the effective periods of products and discounts:

- The FCT_FirstUsageNotify module. See "About Suspending EDRs for Products and Discounts that Start on First Usage".
- The DAT_AccountBatch module determines if an account's products and discounts start on first usage, determines the state of the validity period, and calculates the validity periods based on the EDR timestamp. It sends this information to the FCT_Account module.

  The state of a validity period is used for coordinating the validity-setting and EDR-suspension processes. The state can be one of the following:

  - **NOT_FirstUsage**: Indicates that the product or discount's validity period is already set in the database.
  - **NEW_FirstUsage**: Indicates that the product or discount is configured to start on first usage and that its validity period has not yet been initialized.
  - **INIT_FirstUsage**: Indicates that the product or discount's validity period has been initialized but has not yet been stored in the database.

  If the validity periods of all first-usage resources in the deal should be synchronized, DAT_AccountBatch retrieves a list of the balance groups that have first-usage validity and are associated with the products or discounts in the deal. For information about synchronizing first-usage validity of resources, see "About Synchronizing First-Usage Validity of Resources in Deals" in *BRM Setting Up Pricing and Rating*.
- The FCT_ApplyBalance module. See "About settIng the Validity of Resources Impacted by Discounts".
- The OUT_GenericStream module and related modules. See "Configuring Pipeline Output for First-Usage Products, Discounts, and Resources".

To set the validity for products and discounts that start on first usage, you must also perform these tasks:

- Configure BRM recycling to recycle suspended EDRs. For more information, see "Setting Up Recycling for Events whose Product or Discount Validity Starts on First Usage".
- Use Universal Event (UE) Loader to update the validity periods in the BRM database. See "About Updating Validity Period Information in the BRM Database".

About Suspending EDRs for Products and Discounts that Start on First Usage

Use the FCT_FirstUsageNotify module to suspend EDRs while BRM sets the validity periods of products and discounts that start on first usage. See "FCT_FirstUsageNotify".

Note: To recycle and rate EDRs that are suspended while validity is being set, you must configure BRM standard recycling. See "Configuring Standard Recycling".

If a call details record (CDR) uses both a product and a discount that start on first usage for the first time, the EDR is suspended and recycled twice: once to set the product validity period and again to set the discount validity period. This is because the product may grant resources that can be impacted by the discount, so the product's purchase and cycle events need to be processed before the discount is evaluated.

The FCT_FirstUsageNotify module performs the following tasks:

1. Checks whether the products and discounts used to rate an EDR have validity periods that start on first usage.
2. If a product or discount starts on first usage, flags the EDR for suspense and recycling by setting the ERR_FIRST_USAGE_VALIDITY_NEEDS_INITIALIZING error code and the **FirstUsageValidity** recycle key in the EDR.
3. Notifies the DAT_AccountBatch module that the validity period of the product or discount is being set.
4. Sends the product and discount validity information to a separate output stream.
5. Continues to flag for suspense and recycling subsequent EDRs that use the products or discounts with first-usage validity until the validity period is set.

FCT_FirstUsageNotify predetermines whether the FCT_Reject module will reject an EDR. FCT_FirstUsageNotify does not process EDRs that will be rejected because product, discount, or resource validity should not be set for EDRs that will be otherwise suspended.

Configuring Pipeline Output for First-Usage Products, Discounts, and Resources

In the batch rating pipeline, you configure two output streams: one for files containing products and discounts that start on first usage, and another for files containing resources that start on first usage.

To create the first-usage output streams, configure the following:

- The OUT_GenericStream module. See "Configuring First-Usage Output Streams".
- The DataDescription section of the registry. See "Specifying the First-Usage Format and Mapping Files in the DataDescription Registry".

Configuring First-Usage Output Streams

Configure the OUT_GenericStream module to write the first-usage products, discounts, and resources to an output file.

The default output grammar description files are:

- For products and discounts, **FirstUsageNotify_OutGrammar.dsc**.
- For resources, **FirstUsageResource_OutGrammar.dsc**.

In the EXT_OutFileManager section of the registry, specify the temporary file specifications as follows:

- For first-usage products and discounts:

```
OutputPath = ./data/out/firstUsage/prod_disc
OutputPrefix = test_PROD
OutputSuffix = .out_1
TempPrefix = .
TempDataPath    = ./data/out/firstUsage/prod_disc
TempDataPrefix  = prod.tmp.
TempDataSuffix  = .data_1
```

- For first-usage resources:

```
OutputPath = ./data/out/firstUsage/resources
OutputPrefix = test_RES
OutputSuffix = .out_1
TempPrefix = .
TempDataPath    = ./data/out/firstUsage/resources
TempDataPrefix  = res.tmp.
TempDataSuffix  = .data_1
```

  See "OUT_GenericStream".

Specifying the First-Usage Format and Mapping Files in the DataDescription Registry

Configure the first-usage stream formats, input mapping, and output mapping in the batch pipeline DataDescription registry section as follows:

```
DataDescription
{
   Standard
   {
   ModuleName = Standard
   Module
   {
   StreamFormats
   {
      FIRST_USAGE_NOTIFY_OUTPUT=./formatDesc/Formats/FirstUsageNotify/FirstUsageNotify.dsc
      FIRST_USAGE_RESOURCES=./formatDesc/Formats/FirstUsageNotify/FirstUsageResource.dsc
   }
   . . .
   OutputMapping
   {
      FIRST_USAGE_PROD_DISC=./formatDesc/Formats/FirstUsageNotify/FirstUsageNotify_OutMap.dsc
      FIRST_USAGE_RESOURCES=./formatDesc/Formats/FirstUsageNotify/FirstUsageResource_OutMap.dsc

   }
   }
   }
}
```

For more information, see "Configuring Output for Rated Events and AAA Responses".

About Updating Validity Period Information in the BRM Database

You use UE Loader to update validity periods in the BRM database when first usage occurs.

You use the ConfigurableValidityHandler batch handler to run the UE Loader utility (**uel**).

Perform the following tasks to update validity periods for first-usage products, discounts, and resources:

- Load the first-usage validity Universal Event Mapper templates. See "Loading the First-Usage Validity Templates".
- Configure the ConfigurableValidityHandler batch handler. See "Configuring the ConfigurableValidityHandler Batch Handler".

- Configure Batch Controller. See "Configuring Batch Controller to Start the ConfigurableValidityHandler Batch Handler".

Loading the First-Usage Validity Templates

BRM provides two Universal Event (UE) Mapper event import templates that are used by UE Loader to process the first-usage product, discount, and resource files output by the OUT_GenericStream module:

- **FirstUsageProductsDiscounts.xml**: This template defines the format of the first-usage product and discount files.
- **FirstUsageResources.xml**: This template defines the format of the first-usage resource files.

You must load these templates into the BRM database by running the **pin_uei_deploy** utility.

For example:

```
pin_uei_deploy -c -t FirstUsageProductsDiscounts -i BRM_
Home/formatDesc/Formats/FirstUsageNotify/FirstUsageProductsDiscounts.xml

pin_uei_deploy -c -t FirstUsageResources -i BRM_
Home/formatDesc/Formats/FirstUsageNotify/FirstUsageResources.xml
```

For more information, see "pin_uei_deploy" in *BRM Developer's Guide*.

Configuring the ConfigurableValidityHandler Batch Handler

BRM provides the ConfigurableValidityHandler batch handler for loading first-usage validity data. You must configure ConfigurableValidityHandler to run the following utilities:

- **pin_rel:** This utility loads data for events that are rated in batches.
- **uel**: This utility loads validity period data for products, discounts, and resources when they are used for the first time.
- **pin_load_rerate_jobs**: You must also configure to run this utility if you configured Pipeline Manager to detect and rerate events that are rated out of order. This utility creates rerate jobs for the events that were rated out of order.

For more information, see "About Using a Single Batch Handler to Run Multiple Loading Utilities".

To configure the ConfigurableValidityHandler batch handler, you edit the handler's configuration file (*BRM_Home*/**apps/pin_rel/ConfigurableValidityHandler_config.values**). You specify handler, processing, and staging directories for each loading utility that this batch handler runs.

Configuring Batch Controller to Start the ConfigurableValidityHandler Batch Handler

Batch Controller polls the pipeline output directories and starts the ConfigurableValidityHandler batch handler when a data file is ready to be loaded.

Important: If you use the ConfigurableValidityHandler batch handler, do not configure Batch Controller to run a separate instance of the handlers that load pipeline batch-rated events, out-of-order rerating requests, or first-usage validity data. (You may configure a separate instance of the Rated Event (RE) Loader handler that loads suspended events into **/suspended_usage** objects.) If you have already configured Batch Controller to run the out-of-order event handler (OODHandler), remove those entries or comment them out.

To configure Batch Controller to start the ConfigurableValidityHandler batch handler:

1. Open the Batch Controller **Infranet.properties** file in *BRM_Home***/apps/batch_controller**.
2. Edit the file to include entries for the ConfigurableValidityHandler batch handler.

Note:

- If you have already configured the Batch Controller **Infranet.properties** file for RE Loader, you can use the existing RE Loader entries and most of their values or you can change them. Entries that you must change are marked with an asterisk (*).
- If you have not yet configured the Batch Controller **Infranet.properties** file for RE Loader, you must add ConfigurableValidityHandler values for all of the following entries.

Table 2–2 lists the entries you must set and the default values used for RE Loader:

Table 2–2 Entries for RE Loader

Entry	Description
batch.random.events	Identifies this specific configuration for triggering Batch Controller. For example: `batch.random.events CdrFileEvent`
event_name.**name**	(Optional) A name for the configuration identifier. For example: `CdrFileEvent.name CdrFileEvent`
event_name.**handlers**	The batch handler identifier. For example: `CdrFileEvent.handlers ConfigurableValidityHandler` The default is **relHandler**.
event_name.**file.location**	The full path to the pipeline output directory where the rated-event files are deposited. For example: `CdrFileEvent.file.location /export/portal/integRate`
event_name.**file.pattern**	The rated-event output file name. You can use an asterisk (*) to represent zero or more characters in the file name. No other wildcards (metacharacters) are supported. For example: `CdrFileEvent.file.pattern cdr*.dat`

Table 2–2 (Cont.) Entries for RE Loader

Entry	Description

******handler_name*.name**	The name of the batch handler that is run. For example:
	`ConfigurableValidityHandler.name ConfigurableValidityHandler`
	Note: If you did not change the default RE Loader value for *event_name.handlers*, this entry should be:
	`relHandler.name ConfigurableValidityHandler`

handler_name.max.at.lowload.time
handler_name.max.at.highload.time
	The number of batch handler instances that can run concurrently during periods of low load and high load usage. Typical default settings are **6** at low load and **3** at high load. For example:
	`ConfigurableValidityHandler.max.at.highload.time 3`
	`ConfigurableValidityHandler.max.at.lowload.time 6`

******handler_name*.start.string**	The full path to the ConfigurableValidityHandler handler. For example:
	`ConfigurableValidityHandler.start.string BRM_Home/apps/pin_rel/ConfigurableValidityHandler.pl`
	Note: If you did not change the default RE Loader value for event_name.handlers, this entry should be:
	`relHandler.start.string BRM_Home/apps/pin_rel/ConfigurableValidityHandler.pl`

3. Save and close the file.
4. Stop and restart Batch Controller.

For more information about configuring Batch Controller, see "Controlling Batch Operations" in *BRM System Administrator's Guide*.

Setting Up Recycling for Events whose Product or Discount Validity Starts on First Usage

To suspend and recycle EDRs while product and discount validity is being set, you must configure BRM standard recycling. See "Configuring Standard Recycling".

For information about standard recycling, see "About Standard Recycling".

To recycle events suspended due to first-usage validity, you run the **pin_recycle** utility. Use the **-k** parameter and specify the **FirstUsageValidity** recycle key. For example:

```
pin_recycle -k FirstUsageValidity
```

For more information, see "Using Standard Recycling to Recycle Suspended EDRs" and "pin_recycle".

About First-Usage Validity for Events Rated Out of Order

In the batch rating pipeline, if events are rated in a different order than they occurred, validity periods that are based on first usage may be incorrectly set. This can happen if the first event rated, which initiated the validity period, wasn't actually the first usage event. To correct the validity periods, you must rerate the events. Rerating corrects the order of the events and resets the validity period based on the actual first-usage event. For more information, see "About Automatic Rerating of Out-Of-Order Events" in *BRM Setting Up Pricing and Rating*.

About Customer Rating

Customer rating assigns a rate plan to an EDR based on customer data. The FCT_CustomerRating module performs customer rating. The module creates an associated charge breakdown record and one charge packet, which includes the rate plan code. The FCT_MainRating module uses the rate plan code to rate the event.

To assign the rate plan, the FCT_CustomerRating module does the following:

- If the service that generated the event includes a service-level rate plan extended rating attribute (ERA), that rate plan is used.

  If you're using subscription services, and a subscription service and member service both own a service-level rate plan ERA, the member service's ERA has priority and is used for selecting the rate plan. For information about subscription services, see "Managing Customers' Subscription-Level Services" in *BRM Managing Customers*.

- If there is no service-level rate plan ERA, but there is an account-level rate plan ERA, that rate plan is used.

- If there is no rate plan ERA, the rate plan from the BRM price list associated with the last product found is used when there is more than one product available.

For more information, see "How Pipeline Manager Chooses a Rate Plan" in *BRM Managing Customers*.

To configure customer rating, see "FCT_CustomerRating".

Assigning a Default Rate Plan and Default Segment for Customer Rating

When you configure the FCT_CustomerRating module, you can assign a default rate plan and default segment to use if no customer information for the A number is found.

- Use the **DefaultRateplan** entry to specify the default rate plan name in case there is not enough information to assign a rate plan. You can change this value by using a semaphore.

- Use the **DefaultSegment** entry to specify if segment rating is used, the default segment to use if no segment is found. You can change this value by using a semaphore.

See "FCT_CustomerRating".

About Using the FCT_CustomerRating Module for Multi-Segment Rating

The FCT_CustomerRating module can also choose a rate plan to perform multi-segment rating. In that case, the module reads the segment that applies to the data in the CUST_A data block and uses the rate plans defined for that segment in the IFW_SEGRATE_LNK database table. The module creates one charge packet for each rate plan. You create segments by using Pricing Center.

For information about multi-segment rating, see "About Multi-Segment Rating".

About Customer Rating and Service Level Agreements

You define service-level agreement (SLA) codes in Pricing Center to map service level agreements (SLAs) to a usage-scenario (USC) group, rate-service class (RSC) group, and rule set. The SLA mapping is stored in the IFW_SLA database table.

During customer rating, if the FCT_CustomerRating module finds an SLA code, the module looks up the code in the IFW_SLA database table and adds the following data to the charge packet:

- RSC group
- USC group
- Rule set

Pipeline Manager can use any one of these to determine the rate for the service level usage:

- The RSC group is used by the FCT_RSC_Map module to find the RSC map. FCT_RSC_Map maps the usage class, usage type, service code, and impact category to a new service class. See "About Setting Up RSC Mapping".
- The USC group is used by the FCT_USC_Map module to find the USC map. FCT_USC_Map maps the usage class, usage type, service code, service class, and zone to a new usage type and impact category. See "Setting Up Usage Scenario Mapping" in *BRM Setting Up Pricing and Rating*.

About Multi-Segment Rating

Use *multi-segment rating* to collect business data about your rate plans. With multi-segment rating, you use multiple rate plans on the same EDR and compare the rating results.

A *segment* is a set of rate plans that you define in Pricing Center. When the EDR is rated, a charge packet is created for each rate plan in the segment, and each rate plan is rated in parallel. This provides information on the results of rating the same data when rating is performed by different rate plans.

You can use two methods of assigning segments to EDRs:

- Use the FCT_CustomerRating module to assign segments by using the segment defined in the account. Use the business intelligence segment ERA to assign segments to customer accounts.
- Use the FCT_SegRateNoCust module to assign segments when you don't have accounts for the EDRs. For example, you might want to use a test system that doesn't have access to account data. In that case, you configure FCT_SegRateNoCust to assign segments based on the source network ID.

In either case, when the segment has been found, the modules create an associate charge breakdown record and a charge packet for each rate plan. The subsequent modules, such as the FCT_MainRating module, use those charge packets to add rating data.

To set up and use multi-segment rating, do the following:

1. Create segments in Pricing Center. A segment consists of the following:
   - The rate plan to use for rating.
   - The dates that specify when the segment is valid.
2. If you use account data to assign rate plans for segment rating, assign the business intelligence segment ERA to customer accounts.
3. If you use the business intelligence segment ERA, configure FCT_CustomerRating to find the rate plans. See "FCT_CustomerRating".

4. If you use source networks to find segments, configure FCT_SegRateNoCust. See "FCT_SegRateNoCust".

Configuring Segments in the FCT_SegRateNoCust Module

Use the FCT_SegRateNoCust module **Segments** registry entry to specify the segment to use for each source network. Each rule defines the connection between the source network and the segment. For example:

```
26201 = SegmentD1
26202 = SegmentD2
```

Important: You cannot change these mappings during runtime.

About Rate-Service Class Mapping

You map a rate-service class (RSC) to perform rating based on the quality of service (QoS) when you set up service-level agreements (SLAs). RSC mapping is performed by the FCT_RSC_Map module, which maps the usage class, usage type, service code, and impact category to a new service class.

You create RSC map groups to provide a different service class for each level of service quality. You link the RSC map group to an SLA code in Pricing Center. The FCT_CustomerRating module looks up the SLA code and adds the associated RSC group to the charge packet (see "About Customer Rating and Service Level Agreements"). FCT_RSC_Map uses the RSC group to evaluate the EDR and find the correct RSC map.

The RSC map group is specified in the INTERN_SLA_RSC_GROUP field of the EDR. This field is filled in by FCT_CustomerRating when the product includes a service-level agreement ERA. If INTERN_SLA_RSC_GROUP is empty, the default RSC group is used. You specify the default RSC group in the **DefaultRscGroup** entry of the FCT_RSC_Map registry.

About Setting Up RSC Mapping

To set up RSC mapping, do the following:

1. Use Pricing Center to create RSC maps and RSC map groups.
2. Use Pricing Center to create SLA codes and link them to RSC map groups.
3. Configure FCT_RSC_Map. See "FCT_RSC_Map".

About RSC Maps

To assign a new service class to the EDR, FCT_RSC_Map reads data from the EDR and evaluates it according to the RSC map. An RSC map includes one or more mapping rules that specify the data that must be matched to apply the new service class.

When you create RSC maps, you create a mapping rule for each new service class. You can also define the order in which the rules are evaluated. The new service class is derived from the first rule that matches the data. If no matching rule is found, FCT_RSC_Map uses the default RSC map as defined in the registry. If no default value exists, no mapping is performed.

You can use the following EDR data to create an RSC mapping rule:

- The rate plan used to rate the EDR

- The QoS requested
- Usage class
- Usage type
- Service code
- Service class
- Impact category

When you create the mapping rules, you can use regular expressions. For information on the regular expressions you can use, see "About Using Regular Expressions when Specifying the Data to Extract".

To create a valid mapping, the data in the EDR must match with all of the mapping data.

About Main Rating

The FCT_MainRating module carries out the pipeline rating functionality.

When an EDR is ready for rating, it includes all the data needed by FCT_MainRating; for example, the service class, usage class, zone, and rate plan. The module uses the rate plan to rate the EDR. To rate the EDR, the module uses information about dates, times, and the price model to apply charges. When rating is finished, the EDR contains complete charge breakdown data, including charge packets with charges.

FCT_MainRating uses criteria in the rate plan configuration that apply to an EDR to find the correct price model to use for rating. If a single event is rated by using different time periods, such as peak and off-peak, more than one price model can be used. If an event is mapped to a price model selector, the model selector's rules are evaluated to choose a price model.

If the rate plan configuration includes an alternative price model, the module creates a new charge packet. The EDR is rated again by using the alternative price model. The charge packet is flagged with the A (Alternative) price model type.

During main rating, the module checks for the RUM group associated with the service code. At least one charge packet is added to the EDR for each RUM assigned to the RUM group. A single charge packet is generated for each time period, RUM, and resource that is used. (The resources are defined in the price model.)

To configure main rating, see "FCT_MainRating".

About Rate Adjustment

You use rate adjustments to provide discounts based on date, time, service, and other event attributes. See "About Pipeline Rate Adjustments" in *BRM Setting Up Pricing and Rating*.

To set up rate adjustment, do the following:

1. Create rules that specify which EDRs to adjust and how to adjust them. You have two options:
   - Create rate adjustment rules in Pricing Center. In this case, the rate adjustment data is stored in the Pipeline Manager database. See "About Pipeline Rate Adjustments" in *BRM Setting Up Pricing and Rating*.
   - Create a file that defines usage scenario maps. In this case, the FCT_RateAdjust module reads the file. For information on creating the file, see

"Creating a Rate Adjustment Rules File".

2. Configure FCT_RateAdjust. See "FCT_RateAdjust".

Creating a Rate Adjustment Rules File

The rate adjustment rules can be defined in an ASCII file:

- Each rule consists of a list of fields. The following table describes the meaning of each field.
- Every new line defines an adjustment rule.
- Fields are separated by semicolons (;).
- Comment lines start with #.
- Empty lines are allowed.

Important: The value of the field rank is ignored. The evaluation order of the rules is given by the order of the rules within the file.

Table 2–3 lists the fields in the file:

Table 2–3 Fields in a Rate Adjustment File

Position	Field	Description
1	Rank	Specifies the evaluation order of the rules. This is ignored because the evaluation order is specified within the file.
2	Rate plan	Specifies the rate plan to adjust.
3	Rate plan version	Specifies the rate plan version.
4	Valid from	Specifies the start date for the rate adjustment. This can be either a date with the format *YYYYMMDD* or a weekday with an optional timestamp; for example: ■ 19990524 ■ SAT ■ MON16:00 If the field is left empty, the earliest possible date (19010101) is used.
5	Valid to	Specifies the end date for the rate adjustment. This can be either a date with the format *YYYYMMDD* or a weekday with an optional timestamp. If the field is left empty, the latest possible date (20370205) is used.
6	Time from	Specifies the start time for the rate adjustment. The format is *HH:MM*. The default is **00:00**.

Table 2–3 (Cont.) Fields in a Rate Adjustment File

Position	Field	Description
7	Time to	Specifies the end time for the rate adjustment. The format is *HH*:*MM*. Example: To set up a discount rule that is valid on weekends between 13:00 and 14:00, you have to set the following: ■ **ValidFrom** = SAT ■ **ValidTo** = SUN ■ **TimeFrom** = 13:00 ■ **TimeTo** = 14:00
8	Quantity value	Specifies the maximum quantity value for an EDR container. If this maximum is exceeded, the mapping rule will not be used. If this field is left empty or if a 0 is specified, the rule is valid for every quantity value. Example: You can use this entry to avoid discounting for calls longer than 120 seconds by setting the field to 120.
9	Usage class	Specifies the compare pattern for the usage class.
10	Usage type	Specifies the compare pattern for the usage type.
11	Service code	Specifies the compare pattern for the service code.
12	Service class	Specifies the compare pattern for the service class.
13	Impact category	Specifies the compare pattern for the impact category.
14	Source network	Specifies the compare pattern for the source network.
15	Destination network	Specifies the compare pattern for the destination network.
16	Discount type	Specifies the discount type.
17	Discount value	Specifies the discount value.
18	Comment	Specifies the rate adjustment name.

About Consolidation for BRM Billing

The FCT_BillingRecord module consolidates charge packets and discount packets into an associated BRM billing record in the EDR. This data is loaded as a rated event by RE Loader.

The associated BRM billing record includes the POIDs of the **/account** object and the **/service** object and the POID of the item that receives the balance impact. If an event affects more than one customer balance, an associated BRM billing record is created for each balance.

An associated BRM billing record can contain one or more *balance impact packets*. The data in a balance impact packet is loaded into an **/event** object balance array. Therefore, the data includes information about the charged amount, the rate plan, and resources.

The balance impact packet also includes data in the PIN_INFO_STRING field. This field contains the information about the individual charge packets.

Each balance impact packet includes data for one balance impact per resource. If there are different G/L IDs for the same resource, a balance impact packet is created for each G/L ID.

To configure FCT_BillingRecord, see "FCT_BillingRecord".

Important: Don't use FCT_BillingRecord in a CIBER roaming revenue assurance environment. For more information, see "Billing Consolidation with CIBER Roaming and Revenue Assurance".

How the FCT_BillingRecord Module Works

FCT_BillingRecord uses the data in the EDR to determine if the charges should be included in the associated BRM billing record. To do so, the module checks the following data. If any of these do not match, no associated billing record is created.

- The record type must be 981. This record type is created by the FCT_CustomerRating module for customer rating. In contrast, FCT_BillingRecord does not use record type 984, which is used for multi-segment rating.
- The charge packet must use the following:
  - The standard price model type: PRICEMODEL_TYPE = S
  - A retail rate plan: RATEPLAN_TYPE = R
  - A currency type that matches the currency type specified in the FCT_BillingRecord module registry. The options are Home, Billing, and Rating. See "Defining Currency Exchange Rates" in *BRM Setting Up Pricing and Rating*.
  - A currency that matches one of the entries specified in the FCT_BillingRecord module registry. See "FCT_BillingRecord".

If the charge packet meets the criteria, the module sums the amount, discount, and quantity for each BRM resource and creates the associated BRM billing record and one or more balance impact packets.

For balance monitoring, FCT_BillingRecord generates a monitor packet for each monitor group. See "About Balance Monitoring and Pipeline Rating" in *BRM Managing Accounts Receivable*.

To get data, FCT_BillingRecord connects to the following data modules:

- The DAT_AccountBatch module, which provides data about items.
- The DAT_ItemAssign module, which provides data about items assigned for sponsorship events.
- The DAT_Currency module, which provides data for converting currency symbols to numeric values.
- The DAT_BalanceBatch module, which provides the **ObjectCacheType** value from the **/balance_group** object. See "About Convergent BRM Systems" in *BRM System Administrator's Guide*.

Billing Consolidation with CIBER Roaming and Revenue Assurance

For CIBER roaming and revenue assurance, use the ISC_PostRating iScript instead of FCT_BillingRecord.

ISC_PostRating adds all the retail and wholesale charges and puts them in the DETAIL.RETAIL_CHARGED_AMOUNT_VALUE and DETAIL.WHOLESALE_CHARGED_AMOUNT_VALUE fields.

Note: ISC_PostRating and FCT_BillingRecord perform similar billing consolidation, but ISC_PostRating doesn't load balance impact data into the database.

See "ISC_PostRating".

How the ISC_PostRating iScript Works

ISC_PostRating sets the following attributes:

- Wholesale and retail impact category
- Charged amount value
- Amount currency
- Tax treatment

for an EDR based on these values specified in the Pipeline Manager registry:

- Resource type
- Price model type
- Currency type

Note: This iScript accesses only EDR fields. It doesn't access the database.

Adding Pipeline Rating Data to an Invoice

When you rate usage by using pipeline rating, information about how events are rated are stored in the EDR. You can display this information on invoices. For example, if a call spans two rates, for peak and off-peak time, you can display the rate used for each part of the call. For example:

Table 2–4 Displaying Pipeline Rating Data

Date/time	Called number	Duration	Average rate per unit	Total charge
12/12/2003 1200	4085551212	10min	$0.125	$1.25
12/12/2003 1200	4085551212	5min	$0.15	$0.75
12/12/2003 1200	4085551212	5min	$0.10	$0.50

In the example shown in Table 2–4, the data is stored in the INTERN_PRICE_MDL_STEP_INFO EDR field.

To add pipeline rating invoice data to your invoices, you need to configure the OUT_GenericStream module **AddInvoiceData** registry entry to add the data to the BRM billing record. See "Adding Invoice Data to Pipeline Output" in *BRM Designing and Generating Invoices*.

3

Configuring EDR Input Processing

This chapter describes how to set up input processing for the Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager.

For background information about rating using Pipeline Manager, see "About Pipeline Rating".

To set up EDR output processing, see "Configuring EDR Output Processing".

About the Input Process

To process incoming data, Pipeline Manager input modules convert data into an internal EDR format understood by the pipeline function modules.

The input data can be:

- External files, such as CDRs for telco rating.
- Messages based on Diameter or MBI protocols for processing AAA requests.

The input process works as follows:

1. External files are entered into the BRM system as follows:
   - For rating telco services, your mediation system automatically places CDR files into a directory.

     Important: CDR files must be processed in the correct order. Depending on your mediation system, CDR files might have a sequence number that determines processing order. If not, they are processed according to the last-modified timestamp.

   - For processing AAA requests, incoming messages from prepaid networks are received by the InSocketManager.

   If you start a pipeline and the directory already includes input files, they are processed according to the last-modified timestamp.
2. The input module; for example, "OUT_GenericStream" or InSocketManager; reads data from the external files.
3. The module uses the stream format description file to create separate records from the data. For example, the data is separated into HEADER records, DETAIL records, and TRAILER records. DETAIL records can include data for one service only (for example, GSM *or* GPRS).

4. The input module uses the input grammar file to process each record. The module verifies that the syntactical order for each record is correct. For example, if a particular field is supposed to include 10 characters, the input module uses the grammar file to check that. It also uses the grammar file to normalize data, such as the A number.

   If an error is found, processing stops and an error message is logged.

5. The input module creates an EDR container for the data. See "About EDRs".

6. The input module uses an input mapping file to copy data from the external file into the appropriate EDR container fields.

   Note: A separate input mapping file is required to process each external data format. See "Setting Up an Input Mapping File".

7. The input module puts the EDR into the input buffer for processing by the function modules.

   Note: A separate input grammar file is required to process each external data format. See "Setting Up an Input Grammar File".

The following examples show how the A number in a CDR is mapped to a rating EDR container.

Note: These examples show only selected portions of the stream format description, grammar, and mapping files.

The following example shows part of a stream format description file. This section of the file describes how a DETAIL record is formatted and lists the fields in the record (fields are separated by a semicolon). The first field (SERVICE) stores the service code, the second (A_NUMBER) stores the A number, and so forth. Data in each of the fields must be of type **AscString()**.

```
DETAIL(SEPARATED)
{
Info
{
Pattern = ".*\n";
FieldSeparator = ';';
RecordSeparator = '\n';
}
SERVICE     AscString();
A_NUMBER    AscString();
B_NUMBER    AscString();
...
```

The next example shows part of an input grammar file. The first two lines create a new block of data in an EDR container (**edrNew**) and specify the location in an external file from which the data should be copied (**edrInputMap**). Inside the new block, the next two lines normalize the A number and then add a field to the EDR container called DETAIL.A_NUMBER.

```
edrNew( DETAIL, CONTAINER_DETAIL );
edrInputMap( "SAMPLE.DETAIL.STD_MAPPING" );
```

```
...
number = normalizeNumber( edrString( DETAIL.A_NUMBER ), "00", 0 );
edrString( DETAIL.A_NUMBER ) = number;
```

The next example shows part of an input mapping file. In this example, the A_NUMBER defined in the stream format description example is mapped to the DETAIL.A_NUMBER field defined in the input grammar example. Note how the nested blocks of data correspond to the **edrInputMap** entry (**SAMPLE.DETAIL.STD_MAPPING**) in the input grammar example.

```
SAMPLE
  DETAIL
  {
    HDR_MAPPING
    {
    "010"       -> HEADER.RECORD_TYPE;
    ...
  }
  TRL_MAPPING
  {
    "090"       -> TRAILER.RECORD_TYPE;
    ...
  }

  STD_MAPPING
  {
    "020"       -> DETAIL.RECORD_TYPE;
    0           -> DETAIL.DISCARDING;
    "00"        -> DETAIL.A_MODIFICATION_INDICATOR;
    0           -> DETAIL.A_TYPE_OF_NUMBER;
    "0"         -> DETAIL.A_NUMBERING_PLAN;
    A_NUMBER    -> DETAIL.A_NUMBER;
...
```

You can map one field in an external file to multiple fields in an EDR container. The following example shows part of the same input mapping file. In this part of the file, however, the data block for the GSM service maps the A_NUMBER to a different field:

```
GSMW_MAPPING
  {
  "520"       -> DETAIL.ASS_GSMW_EXT.RECORD_TYPE;
  A_NUMBER    -> DETAIL.ASS_GSMW_EXT.A_NUMBER_USER;
  B_NUMBER    -> DETAIL.ASS_GSMW_EXT.DIALED_DIGITS;
...
  }
```

About Setting Up Input Processing

To set up input processing, do the following:

1. **(CDR processing only)** Define a CDR file input directory in your pipelines. Configure your mediation system to put the files in this folder automatically.

   Note: You can configure your system to route CDR files from a single input directory to multiple identical pipelines. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.

2. Set up a stream format description file. You can start with the sample files that are provided. See "Creating a Stream Format Description File".
3. Set up an input mapping file. You can start with the sample files that are provided. See "Setting Up an Input Mapping File".
4. If necessary, set up an input grammar file. In most cases, you do not need to modify the default grammar file. If you modify an EDR container (for example, if you add fields), you might need to modify the input grammar to ensure that data in your EDR containers is correctly formatted. See "Setting Up an Input Grammar File".

   Important: If you customize an EDR container description, you must ensure that your customizations do not affect existing module functionality. Many modules require data from default EDR fields.

5. Configure these input sections in the registry:
   - The **DataDescription** section. See "Configuring the Input DataDescription Registry Section".
   - The **InputBuffer** section in the **Pipeline** section.
   - The **Input** section. See "Configuring the Input Section in the Registry".

The sample Pipeline Manager registry files include stream format description, input mapping, and input grammar files that convert data using the rating EDR and TAP formats.

About Input Processing File Types

Input processing uses the following types of files:

- **Input file**: The file from the external system.
- **Temporary file**: The same file as the input file, but renamed as a temporary file during processing. If the file is rejected, this file is not used.
- **Done File**: The same file as the input file, but renamed as a done file after processing has been successfully completed.
- **Error File**: The same file as the input file, but renamed as an error file if the file is rejected.

These files are managed by the EXT_InFileManager module. See "About Getting Pipeline Input from Files".

Creating a Stream Format Description File

To create a stream format description file, first identify the data in your external files that you need to use for rating or AAA. You can then either start with one of the sample files or create your own.

In the following example, the external file is a CDR. It includes three records: a HEADER, a TRAILER, and a DETAIL. Each record starts with a character that identifies its EDR container content type (**H** for HEADER, **T** for TRAILER, and **D** for DETAIL), and each record has a fixed structure.

- The example HEADER record in Table 3–1 has two fields:

Table 3–1 Header Record in Stream Format Description

Field	Length	Description
IDENTIFIER	1	The character **H**.
CREATION_TIME	14	The creation time of the CDR stream in the format *YYYYMMDDHHMMSS*.

- The example DETAIL record in Table 3–2 has five fields:

Table 3–2 Detail Record in Stream Format Description

Field	Length	Description
IDENTIFIER	1	The character **D**.
CALLING_PARTY	15	The A number.
CALLED_PARTY	15	The B number.
START_TIMESTAMP	14	The start time of the call in format YYYYMMDDHHMMSS.
DURATION	9	The duration of the call in seconds.

- The example TRAILER record in Table 3–3 has two fields:

Table 3–3 Trailer Record in Stream Format Description

Field	Length	Description
IDENTIFIER	1	The character **T**.
NUMBER_OF_DETAILS	9	The total number of DETAIL records in the stream.

A sample CDR input stream might be the following:

```
H20010613123410D4943311217 4957641506 2001061310011200000045D494106136432 49401531224
2001061310021500000056T000000002
```

The **INP_GenericStream** input module uses the stream format description file to break the CDR input stream into the following records:

```
H20010613123410

D4943311217 4957641506 2001061310011200000045

D494106136432 49401531224 2001061310021500000056

T000000002
```

The input module uses regular expressions to find each record. Table 3–4 lists the three record types and their regular expressions:

Table 3–4 Regular Expressions for the Records

Record	Regular expression	Description
HEADER	"H.{14}"	H followed by 14 arbitrary characters.
DETAIL	"D.{53}"	D followed by 53 arbitrary characters.
TRAILER	"T.{9}"	T followed by 9 arbitrary characters.

The description of the record must contain the following:

- The regular expression used by the input module to recognize the physical record
- The position and types of the fields inside the physical record

A sample format stream description for this example looks like this:

```
SampleFormat
{
  Header(FIX)
  {
    Info
    {
      Pattern = "H.{14}";
    }
    IDENTIFIER        AscString(1);
    CREATION_TIME     AscDate("YYYYmmddHHMMSS");
  }

  Detail(FIX)
  {
    Info
    {
      Pattern = "D.{53}";
    }

    IDENTIFIER        AscString(1);
    CALLING_PARTY     AscString(15);
    CALLED_PARTY      AscString(15);
    START_TIMESTAMP   AscDate("YYYYmmddHHMMSS");
    DURATION          AscInteger(9);
  }

  Trailer(FIX)
  {
    Info
    {
    Pattern = "T.{9}";
  }

    IDENTIFIER        AscString(1);
    NUMBER_OF_DETAILS AscInteger(9);
  }
}
```

Each field in the record is defined by the field type and value. For example, the fields in the DETAIL record are defined as follows:

```
    IDENTIFIER        AscString(1);
    CALLING_PARTY     AscString(15);
    CALLED_PARTY      AscString(15);
    START_TIMESTAMP   AscDate("YYYYMMDDHHMMSS");
    DURATION          AscInteger(9);
```

Record Types

The **INP_GenericStream** input module uses regular expressions to recognize the data records in the input stream. Different types of data records define fields in different ways:

- Fields can defined by fixed-widths.

- Fields can be separated by special characters.
- The length of the field can be included in the input data (as in ASN.1 input).

Therefore, each data record has a record type that tells the input module how to split up the record into fields. Each data record has an **Info** block in its definition that contains some setup parameters for the record, for example, the field separator for a separated record. The record type determines which parameters can be used.

For example, this DETAIL record uses the record type SEPARATED:

```
DETAIL(SEPARATED)
{
Info
{
Pattern = ".*\n";
FieldSeparator = ';';
RecordSeparator = '\n';
}
SERVICE     AscString();
A_NUMBER    AscString();
B_NUMBER    AscString();
...
```

Record Type SEPARATED

The record type SEPARATED is used for records in which fields are separated by a special field delimiter character. The record itself can be terminated by another character (for example, the end-of-line symbol **\n**). Because there is no length information for the record, the regular expression specified as a pattern must match the full record, including the record separator.

There are no restrictions for the data types that can be used inside the SEPARATED record, although it makes no sense to use binary data types inside this record. The length information calculated from the position of the field delimiters overwrites length information specified in the data types. See Table 3–5.

Table 3–5 Parameters in SEPARATED

Parameter	Value	Description	Mandatory
Pattern	String	Regular expression that defines the entire record. This includes all records and the record separator character.	Yes
FieldSeparator	Character	Character that delimits single fields. Default = Comma (,)	No
RecordSeparator	Character	Character that delimits records. Default = No record separator	No

Record Type FIX

This record type is used for records with predefined width for each field. The record must contain all the fields. The record length is calculated as a sum of the widths of individual fields. Only data types with width information can be used. See Table 3–6.

Table 3–6 Parameters in FIX

Parameter	Value	Description	Mandatory
Pattern	String	Regular expression that identifies the record.	Yes

Record Type ASN

This record type is used for file formats defined in ASN.1, for example, TAP. You can use only the TAP and ASN data types in this record type. See Table 3–7.

Table 3–7 Parameters in ASN

Parameter	Value	Mandatory
Application	Integer	No
Context	Integer	No
Private	Integer	No
Universal	Integer	No

Syntax of the Stream Format Description File

The stream format description file is a simple ASCII file. The following grammar defines the syntax of the stream format description file:

```
<format-description-file> ::=  (<use_directive> | <stream-format>)*
<boolean> ::=                  "true" | "false"
<character> ::=                "'" "single character" "'"
<decimal> ::=                  "0..9"* "." "0..9"+
<extension-field-type> ::=     "any field type defined in a user extension"
<extension-record-type> ::=    "any record type defined in a user extension"
<field-name> ::=               <identifier>
<field-type> ::=               "AscString" | "AscDecimal" | "AscLong" | "AscDate" | ... |
                               <extension-field-type>
<format-name> ::=              <identifier>
<identifier> ::=               "a..z,A..Z,_" "a..z,A..Z,0..9,_"*
<info-block> ::=               "Info" "{" <info-parameter>* "}"
<info-parameter> ::=           <identifier> "=" <value> ";"
<integer> ::=                  "0..9"+
<record-definition> ::=        <record-name> "(" <record-type> ")" "{" <info-block>
                               <record-field>* "}"
<record-field> ::=             <field-name> <field-type> "(" [<field-parameter>
                               [, <field-parameter>]*] ")" ";"
<record-name> ::=              <identifier>
<record-type> ::=              "FIX" | "SEPARATED" | "ASN" | <extension-record-type>
<stream-format> ::=            <format-name> "{" <record-definition>* "}"
<string> ::=                   "\"" "any character"* "\""
<use_directive> ::=            "use" <identifier> ";"
<value> ::=                    <decimal> | <integer> | <identifier> | <string> |
                               <character> | <boolean>
```

Supported Data Types for the Stream Format Description File

Each entry in the stream format description file assigns a data type to a field. For example, this line assigns the **AscString** data type to the A number:

```
CALLING_PARTY     AscString(15);
```

Pipeline rating supports the following categories of data types:

- ASCII Data Types
- ASN.1 Data Types
- TAP Data Types

ASCII Data Types

Pipeline Manager supports the following ASCII data types:

- AscDate
- AscDecimal
- AscInteger
- AscString
- AscRawString

AscDate

Use the **AscDate** data type to read and write date/time information as an ASCII string. The **AscDate** data type can be used without any parameters or with a string specifying the used date format.

```
AscDate( [String format] )
```

The format string uses the following patterns:

- **%Y**: The year including the century (1901 ... 2037)
- **%y**: The year without the century (00 ... 99)
- **%m**: Month number (01 ... 12)
- **%d**: Day of the month (01 ... 31)
- **%H**: Hour (00 ... 23)
- **%M**: Minute (00 ... 59)
- **%S**: Seconds (00 ... 59)

When no format string is defined, the following default format is used:

```
%Y%m%d%H%M%S
```

AscDecimal

Use **AscDecimal** to read and write decimal values to and from ASCII streams.

```
AscDecimal( [Integer len [, Bool withPoint [, Integer precision [, Char pointChar [, Identifier
rounding[, Char padChar]]]]]] )
```

- **len**: The total length of the decimal value (default is 0 => unspecified).
- **withPoint**: Boolean flag to specify if there is a decimal point in the string (default is true).
- **precision**: Number of digits after the decimal point (default is 6).
- **pointChar**: Character used as decimal point (default is the point '.').
- **rounding**: Rounding method to use (PLAIN, UP, DOWN, BANK) (default is DOWN).
- **padChar**: Padding character to use (default is '0').

AscInteger

Use **AscInteger** to read and write integer values to and from ASCII streams.

Integer values are supported in the range from -2147483648 to 2147483647.

Note: AscInteger cannot be NULL (empty).

```
AscInteger( [Integer len [, Char padChar]] )
```

- **len**: The total length of the integer value (default is 0 => unspecified)
- **padChar**: The character used to pad integer values to a fixed length (default is the '0')

AscString

Use the AscString data type to read and write strings to and from an ASCII stream.

```
AscString( [Integer len [, Char padChar [, Bool isLeftJustified]]] )
```

- **len**: The total length of the string (default is 0 => unspecified).
- **padChar**: The character used to pad string values to a fixed length (default is a space character).
- **isLeftJustified**: Flag indicating that the string is left justified (default is **true**).

AscRawString

Equivalent to **AscString**, but preserves leading and trailing spaces while **AscString** strips all spaces from strings.

ASN.1 Data Types

Pipeline Manager supports the following ASN.1 data types:

- ASN_Integer
- ASN_LegacyOctetString
- ASN_OctetString
- ASN_RawOctetString
- ASN_BcdString
- ASN_NumberString
- ASN_HexString
- ASN_Tag
- ASN_Blob

ASN_Integer

Use **ASN_Integer** to read and write integer values from and to ASN.1 streams. Integer values are supported in the range from -2147483648 to 2147483647.

Note: ASN_Integer cannot be null (empty).

```
ASN_Integer( Integer TagValue [, String Asn1Class] )
```

- **TagValue**: The value to use as ASN.1 Tag.
- **Asn1Class**: The Class of the ASN.1 object. Values are:
  - **Application**

  - **Context**
  - **Private**
  - **Universal**

  The default is **Application**.

ASN_LegacyOctetString

Use ASN_LegacyOctetString to read and write an Octet string, which is a Byte string without any specific encoding for the data, for example, ascii or hex, from and to ASN.1 streams. ASN_LegacyOctetString removes the leading and trailing spaces after decoding an octet string.

This data type is similar to the ASN_OctetString data type except for the following difference:

- ASN_LegacyOctetString encodes an empty octet string with length = 0 and no value.
- ASN_OctetString builds an empty octet string with length = 1 and a space for the value. See "ASN_OctetString".

```
ASN_LegacyOctetString( Integer TagValue [, String Asn1Class] )
```

- **TagValue**: The value to use as ASN.1 Tag.
- **Asn1Class**: The Class of the ASN.1 object. Values are:
  - **Application** (Default)
  - **Context**
  - **Private**
  - **Universal**

ASN_OctetString

Use ASN_OctetString to read and write strings from and to ASN.1 streams. An Octet string is a Byte string without any specific encoding for the data, for example, ascii or hex.

```
ASN_OctetString( Integer TagValue [, String Asn1Class] )
```

- **TagValue**: The value to use as ASN.1 Tag.
- **Asn1Class**: The Class of the ASN.1 object. Values are:
  - **Application** (Default)
  - **Context**
  - **Private**
  - **Universal**

ASN_RawOctetString

Use ASN_RawOctetString to read and write an Octet string, which is a Byte string without any specific encoding for the data, for example, ascii or hex, from and to ASN.1 streams. Unlike the ASN_LegacyOctetString, ASN_RawOctetString does *not* remove the leading and trailing spaces after decoding an octet string.

See also "ASN_LegacyOctetString".

```
ASN_RawOctetString( Integer TagValue [, String Asn1Class] )
```

- **TagValue**: The value to use as ASN.1 Tag.
- **Asn1Class**: The Class of the ASN.1 object. Values are:
  - **Application** (Default)
  - **Context**
  - **Private**
  - **Universal**

ASN_BcdString

ASN_BcdString is an extension of the **ASN_OctetString** used to read and write strings containing data coded in the Binary Coded Decimal form to and from ASN.1 streams. This type automatically decodes and encodes BCD, so the user accesses the data seamlessly.

```
ASN_BcdString( Integer TagValue [, String Asn1Class] )
```

- **TagValue**: The value to use as ASN.1 Tag
- **Asn1Class**: The Class of the ASN.1 object. Values are:
  - **Application** (Default)
  - **Context**
  - **Private**
  - **Universal**

ASN_NumberString

ASN_NumberString is an extension of the **ASN_OctetString** used to read and write strings containing only numbers (and spaces) but packed in an ascii string from ASN.1 streams. An **ASN_NumberString** can be read and written as a string, date, or long.

```
ASN_NumberString( Integer TagValue [, String Asn1Class] )
```

- **TagValue**: The value to use as ASN.1 Tag.
- **Asn1Class**: The Class of the ASN.1 object. Values are:
  - **Application (Default)**
  - **Context**
  - **Private**
  - **Universal**

ASN_HexString

ASN_HexString is an extension of the **ASN_OctetString** used to read and write strings containing data coded in the Hexadecimal form, but stored as ASCII strings, from and to ASN.1 streams. This type is used because iScript cannot directly manipulate hexadecimal byte strings, so the strings are stored as ASCII representation of hexadecimal strings. For example, 0x28F3 is stored as 28F3.

ASN_HexString supports cases in which a special conversion method of read or write access is necessary.

```
ASN_HexString( Integer TagValue [, String Asn1Class] )
```

- **TagValue**: The value to use as ASN.1 Tag.
- **Asn1Class**: The Class of the ASN.1 object. Values are:

  - **Application** (Default)
  - **Context**
  - **Private**
  - **Universal**

ASN_Tag

Use **ASN_Tag** to read and write constructed ASN.1 objects to and from ASN.1 streams. Only the Parser should create this type of objects, out of record definitions in the block description file.

The **ASN_Tag** object can read both definite and indefinite length ASN.1 objects.

ASN_Blob

ASN_Blob is a special type used to store a complete structured (constructed) ASN.1 Object in the form of a byte string. This is useful when you need to transmit a block of data from the input to the output without processing, thus not needing to map the data into EDR container fields.

The only limitation for this type is that the ASN.1 object must have a definite length.

```
ASN_Blob( Integer TagValue [, String Asn1Class [, String Asn1Form]] )
```

- **TagValue**: The value to use as ASN.1 Tag
- **Asn1Class**: The Class of the ASN.1 object. Values are:
  - **Application** (Default)
  - **Context**
  - **Private**
  - **Universal**
- **Asn1Form**: The Form of the ASN.1 object. Values are:
  - Constructed (Default)
  - Primitive

TAP Data Types

Pipeline Manager supports the following TAP data types. These are defined only to match the type name used in the TAP format description file.

- **TAP_AsciiString.** Same type as a standard **ASN_OctetString**.
- **TAP_Description.** Same type as a standard **ASN_OctetString**.
- **TAP_Currency.** Same type as a standard **ASN_OctetString**.
- **TAP_PercentageRate.** Same type as a standard **ASN_Integer**.

Setting Up an Input Mapping File

To create an input mapping file, first identify the data in your external files that you need to map from the external files to the EDR container. You can then either start with one of the sample input mapping files or create your own.

Each mapping entry contains a list of mappings either from data record fields to EDR container fields or from constant values to EDR container fields. You can map a data

record field to more than one EDR container field by adding more than one mapping. For example:

```
A_NUMBER    -> DETAIL.A_NUMBER;
.
.
.
A_NUMBER    -> DETAIL.ASS_GSMW_EXT.A_NUMBER_USER;
```

The following grammar defines the syntax of the input mapping file:

```
<input-mapping-file> ::=    <file-format-mapping>*
<constant> ::=              <integer> | <decimal> | <string>
<constant-mapping> ::=      <constant> "->" <edr-field>
<decimal> ::=               "0..9"* "." "0..9"+
<edr-field> ::=             <identifier> ("." <identifier>)+
<field-mapping> ::=         <field-name> "->" <edr-field>
<field-name> ::=            <identifier>
<file-format-mapping> ::=   <file-format> "{" <record-mapping>* "}"
<identifier> ::=            "a..z,A..Z,_" "a..z,A..Z,0..9,_"*
<integer> ::=               "0..9"+
<mapping-entry> ::=         <field-mapping> | <constant-mapping>
<mappings> ::=              <mapping-name> "{" <mapping-entry>* "}"
<record-mapping> ::=        <record-name> "{" <mappings>* "}"
<string> ::=                "\"" "any character"* "\""
```

Setting Up an Input Grammar File

The input grammar contains iScript statements to create an EDR container.

The syntax of the input grammar file is similar to the syntax used in Yacc grammars. This file defines the grammar of the input data that are parsed and of the iScript statements that are executed when a certain symbol is found in the input data stream.

Configuring the Input DataDescription Registry Section

You configure a **DataDescription** section in the registry for each pipeline. The **DataDescription** section includes the following entries:

- **StreamFormats**: Specifies the input stream format file.

  See "About the Order of Listing Stream Format Description Files".

- **InputMapping**: Specifies the input mapping description file.
- **OutputMapping**: Specifies the output mapping description file.

> **Note:** You specify the input grammar description file in the **Input** section.

This sample shows the **DataDescription** section:

```
DataDescription
  {
   Standard
   {
   ModuleName = Standard
   Module
   {
   StreamFormats
```

```
    {
      Format1 = ./formatDesc/Formats/Flist/Flist_v01.dsc
    }
    InputMapping
    {
     Mapping1 = ./formatDesc/Formats/Flist/Flist_v01_InMap.dsc
    }
    OutputMapping
    {
     Mapping1 = ./formatDesc/Formats/Flist/Flist_v01_OutMap.dsc
    }
    }
    }
}
```

Note: The DataDescription section also includes an entry for the output mapping file. See "Configuring EDR Output Processing".

About the Order of Listing Stream Format Description Files

Pipeline module instances prioritize the order in which formats are considered for parsing in the order that the stream format description files are listed in the **StreamFormats** section of the registry.

Parsing errors can occur in a pipeline if the stream format description files are listed in the incorrect order. For example, the stream format description file that applies to your custom input stream would be listed in the **StreamFormats** section *before* the output stream format description file.

Configuring the Input Section in the Registry

Note: When you configure the Input section, you configure the Pipeline Input Controller. See "Input Controller".

To configure the **Input** section in the registry, do the following:

- The **UnitsPerTransaction** entry: Use this entry to improve performance. See "Combining Multiple CDR Files Into One Transaction" in *BRM System Administrator's Guide*.
- The **INP_GenericStream** module: Specify the input grammar file in this section. See "INP_GenericStream".

  When configuring the INP_GenericStream module, configure one of the following modules as a submodule of the INP_GenericStream module:

  - **EXT_InFileManager**: Configure this module if the pipeline receives input from files. This module manages the input, temporary, and done files. See "EXT_InFileManager".
  - EXT_InEasyDB: Configure this module if the pipeline receives input from a database. See "EXT_InEasyDB".
  - If the pipeline receives input from a prepaid network, configure **EXT_InSocketMgrFlist** for input from flist-based networks.

About Getting Pipeline Input from Files

To configure a pipeline to read data from files, use the EXT_InFileManager module.

When you configure the module, you specify the directories, suffixes, and prefixes for the following files:

- **Input files**: CDR files from the mediation system. The prefix and suffix are used by the input module to identify which files to process.

  The input module checks periodically for files in this folder with the specified prefix and/or suffix.
- **Done files**: Created when a transaction is successfully completed.
- **Error files**: Created after a transaction rollback.

To manage file names, you can specify the following:

- The prefix for temporary files. Temporary files are used as input until the transaction is complete.
- Whether to replace or append prefixes and suffixes.
- The time period (in seconds) for which the input directory must be empty before the EVT_INPUT_DIR_EMPTY event is sent.

See "EXT_InFileManager".

About Getting Pipeline Input from a Database

To get pipeline input from a database, use the EXT_InEasyDB module.

To set up a pipeline for database input:

1. Create a job file that consists of SQL statements. It can also include iRule variables, which are defined in a parameter file. The EXT_InEasyDB module uses the commands to create EDRs.
2. Place the job file in a specific directory. When Pipeline Manager starts, the EXT_InEasyDB module finds the directory from the registry and starts the input process according to the commands in the job file.

You can stop and restart the pipeline after a system crash by configuring a restart file.

You can start the pipeline by using the **ReadDatabase** semaphore. When the module receives a start command while in process, the new SQL command is written to a job file.

The returned values of the database input stream contain all fields of each selected row divided by the configurable delimiter.

To configure the pipeline to read data from a database, see "EXT_InEasyDB".

Specifying the Maximum Errors Allowed in an Input File

You can configure the Output Controller to reject an entire input stream that exceeds a maximum percentage of errors. For example, you can specify to reject an input stream if over 20% of the EDRs have a particular error.

You specify the error threshold by using the **MaxErrorRates** entry in the Output section of the registry file.

Note: You can also configure a pipeline to reject individual EDRs by using the FCT_Reject module. For information, see "About Standard Recycling" and "Recycling EDRs in Pipeline-Only Systems".

When an input stream exceeds the error threshold, the Output Controller:

- Deletes all output streams associated with the input stream. For example, if a pipeline splits an input stream into five output streams, the Output Controller deletes all five output streams.
- Moves the input stream to the error directory. You define the location of the error directory by using the ErrorPath registry entry. For information, see "EXT_InFileManager".

To set an error threshold:

1. Stop Pipeline Manager, if necessary. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.
2. Open your registry file in a text editor.
3. Edit the Pipeline Output Controller's **MaxErrorRates** registry entries, making sure you:
   - List all error codes that the Pipeline Output Controller should monitor.
   - Specify an error threshold for each error code. The threshold specifies the maximum percentage of EDRs that are allowed to have the particular error.

   For example, to configure the Output Controller to reject a stream if any of the following are true:
   - Over 10% of the EDRs have an INF_EDR_REJECTED error.
   - Over 8% of the EDRs have an ERR_CUST_NOT_FOUND error.
   - Over 20% of the EDRs have an ERR_CHARGED_ZONE_NOT_FOUND error.

```
Output
{
    ...
    MaxErrorRates
    {
        INF_EDR_REJECTED = 10
        ERR_CUST_NOT_FOUND = 8
        ERR_CHARGED_ZONE_NOT_FOUND = 20
    }
    ...
}
```

4. Save and close the registry file.
5. Restart Pipeline Manager. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Reading TAP Files

Pipeline Manager can read the following TAP versions:

- TAP-0301 from the TD57v3.04.00 specification
- TAP-0303 from the TD57v3.07.01 specification
- TAP-0304 from the TD57v3.08.02 specification

- TAP-0309 from the TD57v3.90 specification
- TAP-0310 from the TD57v3.10.01 specification
- TAP-0311 from the TD57v28 specification
- TAP-0312 from the TD57v30.1 specification
- TAP-0312 from the TD57v32.1 specification

You can specify TAP input grammar files when you set up your input modules. The files are located in the *Pipeline_Home*/**formatDesc/Formats/TAP3** directory. *Pipeline_Home* is the directory where you installed Pipeline Manager.

Important: Because the TAP 3.10 standard introduced fundamental changes, files produced according to earlier versions of the TAP standard are not compliant with the TAP 3.10 standard. Therefore, the TAP 3.10 grammar files cannot be used to process previous TAP versions.

Note the following implementation details:

- The following records are generated when TAP is processed by Pipeline Manager:
  - 1 Header record (010): 1 EDR
  - 1 Trailer record (090): 1 EDR
  - 1 GPRS record (040 for SGSN, 042 for GGSN or mixed ticket): 1 EDR
  - 1 mobile supplementary service (MSS) record (029): 1 EDR
  - 1 service center usage (SCU) record (050): 1 EDR
  - 1 value added service (VAS) record (060): 1 EDR
  - 1 content transaction (CONT) record (999): 1 EDR
  - 1 location service (LOCN) record (998): 1 EDR
  - 1 mobile originating call (MOC) record (021): *n* EDRs: one EDR for every basic service used.
  - 1 mobile terminating call (MTC) record (031): *n* EDRs: one EDR for every basic service used.
- For each supplementary service, an SS_PACKET is created and attached to the corresponding EDR. For every charge detail of the VAS array, a charge packet is added to the latest generated EDR. The VAS short description is stored in the DETAIL.ASS_CBD.CP.PRODUCTCODE_USED field.
- The Value added service used block is used to build an associated charge breakdown record containing data for rating.
- The CAMEL service information is stored in the ASSOCIATED_CAMEL_EXTENSION ("700") block of the EDR. The associated charge packets are stored on the same ASS_CBD as others but with the PRODUCTCODE_USED field set to CAMEL to identify them.
- Mobile originating and terminating records (MOC and MTC records) are split into multiple records downstream by the ISC_TapSplitting iScript. This iScript generates one EDR for every basic service in the basic service used array. For more information, see "ISC_TapSplitting".

Note the following restrictions:

- The size of the ASN.1 string is not checked during input.
- Pipeline Manager does not add the MIN to the EDR.
- Pipeline Manager does not add the electronic serial number (ESN) to the EDR.

The TAP output grammar recognizes all record types generated by Pipeline Manager.

About Customizing Mapping of Flist Fields to Rating EDR Container Fields

To process events received from the Connection Manager (CM), a real-time pipeline converts the event data from flist format to rating EDR format for processing by Pipeline Manager.

BRM provides default flist-to-rating-EDR mappings for GSM, GPRS, and SMS events in the *Pipeline_Home*/**formatDesc/Formats/Realtime/rate_event.xml** file. When the real-time pipeline starts, the INP_Realtime module uses the descriptions in this XML file to construct an in-memory representation of the flist-to-rating-EDR mapping. The pipeline uses the in-memory mapping to convert incoming flists to rating EDR format.

Table 3–8 displays the XML elements are used in the flist-to-rating-EDR mapping:

Table 3–8 XML Elements in flist-to-rating-EDR Mappings

XML element	Description
OpcodeMap	The root node of the XML document. The XML document can have only one OpcodeMap element. This element requires the *opcode* attribute, which is a unique string used to differentiate between OpcodeMap elements in other XML files.
InMap	The flist-to-rating-EDR mappings for the input flist. The XML document can have only one InMap element. The *containerType* attribute specifies the root-level name of the rating EDR container.
FlistField	A field in the input flist. If the *target* attribute is present, the flist field is mapped to the target field in the rating EDR container.
EdrBlock	Indicates when a rating EDR Block should be created.

Sample input flist:

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 12035 15
0 PIN_FLD_EVENT        SUBSTRUCT [0] allocated 52, used 52
1     PIN_FLD_POID           POID [0] 0.0.0.1 /event/delayed/session/telco/gsm
1373193266068995136 0
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 12035 0
1     PIN_FLD_START_T      TSTAMP [0] (1081882800) Tue Apr 13 12:00:00 2004
1     PIN_FLD_END_T        TSTAMP [0] (1081883200) Tue Apr 13 12:00:00 2004
1     PIN_FLD_QUANTITY    DECIMAL [0] 400
1     PIN_FLD_TELCO_INFO   SUBSTRUCT [0] allocated 20, used 12
2         PIN_FLD_CALLING_FROM    STR [0] "0049100050"
2         PIN_FLD_CALLED_TO       STR [0] "0049100051"
2         PIN_FLD_USAGE_CLASS     STR [0] "NORM"
2         PIN_FLD_TERMINATE_CAUSE   ENUM [0] 0
1     PIN_FLD_GSM_INFO     SUBSTRUCT [0] allocated 20, used 14
2         PIN_FLD_CALLED_NUM_MODIF_MARK   ENUM [0] 0
2         PIN_FLD_DIRECTION          ENUM [0] 0
2         PIN_FLD_CELL_ID            STR [0] "123456"
2         PIN_FLD_SUB_TRANS_ID       STR [0] "S"
```

```
                2         PIN_FLD_DESTINATION_SID    STR [0] ""
                2         PIN_FLD_NUMBER_OF_UNITS    DEC[0] 1.0
```

The default flist-to-rating-EDR mapping in XML for the above flist:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<OpcodeMap xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="./OpcodeMapping.xsd"
opcode="PCM_OP_RATE_PIPELINE_EVENT">
   <InMap containerType="DETAIL">
      <!--CONSTANT EDR DETAIL ITEMS-->
      <EdrField name="DETAIL.RECORD_TYPE" value="020" />
      <EdrField name="DETAIL.DISCARDING" value="0" />
      <!--EVENT FLIST ITEMS-->
      <FlistField name="PIN_FLD_EVENT">
         <!--EVENT POID-->
         <FlistField name="PIN_FLD_POID" format="type" target="DETAIL.EVENT_TYPE"
alias="EventType"/>
         <!--ACCOUNT_OBJ-->
         <FlistField name="PIN_FLD_ACCOUNT_OBJ" format="short" target="DETAIL.CUST_A.ACCOUNT_
PARENT_ID" />
         <!--CHARGING_START-->
         <FlistField name="PIN_FLD_START_T">
            <EdrField name="DETAIL.CHARGING_START_TIMESTAMP" />
            <EdrField name="DETAIL.NE_CHARGING_START_TIMESTAMP" />
         </FlistField>
         <!--CHARGING_END-->
         <FlistField name="PIN_FLD_END_T">
            <EdrField name="DETAIL.CHARGING_END_TIMESTAMP" />
            <EdrField name="DETAIL.NE_CHARGING_END_TIMESTAMP" />
         </FlistField>
         <!--QUANTITY FIELD-->
         <FlistField name="PIN_FLD_QUANTITY" target="DETAIL.DURATION" />
         <!--TELCO FLIST ITEMS-->
         <FlistField name="PIN_FLD_TELCO_INFO">
            <!--A_NUMBER-->
            <FlistField name="PIN_FLD_CALLING_FROM">
               <EDRField name="DETAIL.A_NUMBER"/>
               <EDRField name="DETAIL.ASS_GSMW_EXT.A_NUMBER_USER" onAliasName="EventType"
onAliasValue="/event/delayed/session/telco/gsm" />
            </FlistField>
            <!--B NUMBER-->
            <FlistField name="PIN_FLD_CALLED_TO">
               <EDRField name="DETAIL.B_NUMBER"/>
               <EDRField name="DETAIL.ASS_GSMW_EXT.DIALED_DIGITS" onAliasName="EventType"
onAliasValue="/event/delayed/session/telco/gsm" />
            </FlistField>
            <!--USAGE_CLASS-->
            <FlistField name="PIN_FLD_USAGE_CLASS" target="DETAIL.USAGE_CLASS" />
            <!--CALL_COMPLETION_INDICATOR-->
            <FlistField name="PIN_FLD_TERMINATE_CAUSE" target="DETAIL.CALL_COMPLETION_INDICATOR" />
         </FlistField> <!--END TELCO FLIST ITEMS-->
         <!--GSM FLIST ITEMS-->
         <FlistField name="PIN_FLD_GSM_INFO" onAliasName="EventType"
onAliasValue="/event/delayed/session/telco/gsm" optional="true">
            <EdrBlock name="DETAIL.ASS_GSMW_EXT"/>
            <!--CONSTANT DETAIL.ASS_GSMW_EXT EDR ITEMS-->
            <EdrField name="DETAIL.ASS_GSMW_EXT.RECORD_TYPE" value="520" />
            <FlistField name="PIN_FLD_CALLED_NUM_MODIF_MARK" target="DETAIL.B_M ODIFICATION_
INDICATOR" />
            <FlistField name="PIN_FLD_DIRECTION" target="DETAIL.USAGE_DIRECTION" />
```

```
            <FlistField name="PIN_FLD_CELL_ID" target="DETAIL.ASS_GSMW_EXT.CELL_ID"/>
            <FlistField name="PIN_FLD_SUB_TRANS_ID" target="DETAIL.LONG_DURATION_INDICATOR" />
            <FlistField name="PIN_FLD_NUMBER_OF_UNITS" target="DETAIL.NUMBER_OF_UNITS" />
          </FlistField> <!--END GSM FLIST ITEMS-->
          <!--END EVENT FLIST ITEMS-->
        </FlistField>
    </InMap>
</OpcodeMap>
```

You can customize the default GSM, GPRS, and SMS mappings and create custom mappings for other types of events. To customize flist-to-rating-EDR mappings, you must be familiar with the following topics:

- BRM flists. See "Understanding the PCM API and the PIN Library" in *BRM Developer's Guide*.
- XML
- XML Schema

To create and use a custom mapping:

1. Edit the pipeline event XML file (*Pipeline_Home*/**formatDesc/Formats/Realtime/rate_event.xml)** to add your custom mappings.
2. Validate the XML file using *Pipeline_Home*/**formatDesc/Formats/Realtime/opcode_ifw_mapping.xml** file.

> **Important:** Using an invalid XML file prevents the INP_Realtime module from successfully starting. Make sure you validate the XML file against the XML schema.

About the POID Format in the Rating EDR Container

If an flist field is a POID, the *format* attribute is required to indicate the format of the POID in the rating EDR container. The following format types are supported: long, short, id, and type.

For example, the format of POID 0.0.0.1 /account 12065 is mapped as follows:

- **1_12065 /account** when the format attribute is *long*
- **1_12065** when the format attribute is *short*
- **12065** when the format attribute is *id*
- **/account** when the format attribute is *type*

Mapping an Flist Field to Multiple Rating EDR Container Fields

If the FlistField element contains child **EDRField** elements, the flist field is mapped to multiple rating EDR fields.

In the following example, PIN_FLD_END_T is mapped to multiple fields in the DETAIL EDR block:

```
<FlistField name="PIN_FLD_END_T">
<EdrField name="DETAIL.CHARGING_END_TIMESTAMP" />
<EdrField name="DETAIL.NE_CHARGING_END_TIMESTAMP" />
</FlistField>
```

Using Conditions to Map an Flist Field to a Rating EDR Container Field

You can use the FlistField attributes *alias*, *onAliasName*, and *onAliasValue* for conditional mappings.

In this example, the input flist is a **/event/delayed/session/telco/gsm** event. The PIN_FLD_GSM_INFO substruct of the event is mapped to the DETAIL.ASS_GSMW EDR block.

```
<FlistField name="PIN_FLD_POID" format="type" target="DETAIL.EVENT_TYPE" alias="EventType"/>
<!--GSM FLIST ITEMS-->
<FlistField name="PIN_FLD_GSM_INFO" onAliasName="EventType"
onAliasValue="/event/delayed/session/telco/gsm" optional="true"> <EdrBlock name="DETAIL.ASS_GSMW_
EXT"/>
<!--CONSTANT DETAIL.ASS_GSMW_EXT EDR ITEMS-->
<EdrField name="DETAIL.ASS_GSMW_EXT.RECORD_TYPE" value="520" />
<EdrField name="DETAIL.ASS_GSMW_EXT.TIME_BEFORE_ANSWER" value="0" />
<EdrField name="DETAIL.ASS_GSMW_EXT.NUMBER_OF_SS_PACKETS" value="0" />
<FlistField name="PIN_FLD_CALLED_NUM_MODIF_MARK" target="DETAIL.B_MODIFICATION_INDICATOR" />
<FlistField name="PIN_FLD_DIRECTION" target="DETAIL.USAGE_DIRECTION" />
<FlistField name="PIN_FLD_CELL_ID" target="DETAIL.ASS_GSMW_EXT.CELL_ID" />
<FlistField name="PIN_FLD_SUB_TRANS_ID" target="DETAIL.LONG_DURATION_INDICATOR" />
<FlistField name="PIN_FLD_NUMBER_OF_UNITS" target="DETAIL.NUMBER_OF_UNITS" />
</FlistField> <!--END GSM FLIST ITEMS-->
```

You can also map an alias to an **EdrField** defined elsewhere. For example:

```
<FlistField name="PIN_FLD_START_T" target="DETAIL.CHARGING_START" alias="startTime"/>
<!--GSM FLIST ITEMS-->
<FlistField name="PIN_FLD_GSM_INFO" >
<EdrBlock name="DETAIL.ASS_GSMW_EXT"/>
<!--CONSTANT DETAIL.ASS_GSMW_EXT EDR ITEMS-->
<EdrField name="DETAIL.ASS_GSMW_EXT.CHARGING_START" useAlias="startTime" />
</FlistField> <!--END GSM FLIST ITEMS-->
```

4

Configuring EDR Output Processing

This chapter describes how to set up output processing for the Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager. The input can be CDRs for telco rating.

For background information about Pipeline Manager, see "About Pipeline Rating".

To set up EDR input processing, see "Configuring EDR Input Processing".

About the Output Process

Pipeline Manager can generate data for various purposes:

- To provide rated events for the Rated Event (RE) Loader to load into the BRM database.
- To collect rejected EDRs for recycling.
- To collect duplicate EDRs.
- To send data to the Pipeline Manager database for additional processing in another pipeline.
- To handle discarded EDRs.

The output process works as follows:

1. The completed EDRs are moved to the output buffer.
2. The output module reads data from the output buffer and processes the data. The processing performed depends on the output configuration, for example:
   - If the output consists of rated EDRs, the output module uses grammar and description files to convert the EDRs to a format that the RE Loader uses.
   - If the output consists of rejected EDRs, the output module creates reject files.
   - If the output consists of AAA EDRs, the output module uses grammar and description files to convert the EDRs to a format that the external prepaid network understands, such as flist, Diameter, or MBI.
3. The output module writes the data to the specified output stream. The destination of an output stream can be a file directory, a location in the database, or an external network. You can configure different directories, file names, and network destinations for each output stream.

About the Output Processing System Components

Output processing is managed by the output controller and the output collection module.

- The *output controller* manages the overall output process. You can configure output properties that apply to all streams. See "Output Controller".

  The Output controller performs the following functions:

  - Manages the output stream's trailer information.
  - Rejects input streams when they exceed a specified maximum number of errors.
  - Checks for duplicate CDR files.
  - Notifies the Transaction Manager when a transaction ends.
  - Stops the pipeline when critical errors occur.

  The input stream's trailer information can become invalid when a pipeline discards or rejects individual EDRs or when a pipeline splits EDRs into multiple output streams. For example, the input stream's trailer contains a total charge amount field, which contains the charge amount for all EDRs. When a pipeline discards some EDRs in the input stream, the total charge amount is no longer valid. To correct this, the Output Controller automatically recalculates the trailer information for each output stream.

- *Output collection* defines all the output streams for the pipeline. The Output Collection module performs the following functions:

  - Generates the output devices that are specified in the registry at system startup.
  - Passes the EDR container to the specified output device.

  You configure the Output Collection module by editing a pipeline's **OutputCollection** section of the registry file. For information, see "Output Controller".

About the Output Modules

The following output modules are available:

- **OUT_GenericStream**: Used to process rated events. This module converts the EDR format to a format needed for further processing, for example, RE Loader format to load rated events or TAP format for outcollect processing of roaming records.

  See "Configuring Output for Rated Events and AAA Responses" and "OUT_GenericStream".

- **OUT_Reject**: Used to process rejected or duplicate events. See "Configuring Output for Rejected or Duplicate EDRs" and "OUT_Reject".
- **OUT_DB**: Used to send data to the database. See "Sending Output to a Database" and "OUT_DB".
- **OUT_DevNull**: Used to discard EDRs that you don't want to recycle. See "Configuring Output of Discarded EDRs" and "OUT_DevNull".
- **EXT_OutFileManager**: Used to manage the output files of the OUT_GenericStream module and the OUT_Reject modules. See "EXT_OutFileManager".

About Output Processing File Types

Output processing uses the following types of files:

- **Temporary data file**: Stores records during processing. There is one temporary file for each external file being processed. If the entire external file is rejected, the temporary file is not used.
- **Output File**: Stores the EDRs after processing. This file is derived from the temporary file. When processing is completed successfully, the temporary file is renamed and becomes the output file.
- **Temporary stream file list**: Stores the names of temporary data files. This is required when the Replace registry entry is enabled. In that case, the output file name is appended with a sequence number. Each output stream uses a separate temporary stream file list. For example, the telephony output stream registry includes these entries:

```
TempDataPath       = ./samples/wireless/data/telout
TempDataPrefix     = tel.tmp.
TempDataSuffix     = .data
```

> **Note:** These registry entries are used for internal pipeline processing only and should not be changed.

These output processing files are managed by the "EXT_OutFileManager" module.

About ASN.1 Output

ASN.1 objects are composed of a triplet TLV (Tag/LengthOfValue/Value). Therefore, before writing ASN.1 to the output, you must calculate the **LengthOfValue** field for every element of the ASN.1 tree that you are building. To do this, use the EXT_AsnTree module.

This extension has functions to perform the following tasks:

- Builds a tree of ASN.1 objects.
- Updates the **LengthOfValue** fields of the ASN.1 objects.
- Flushes the resulting ASN.1 data block to an output stream.

For more information, see "ASN.1 Functions" in *BRM Developer's Reference*.

About Configuring Output Processing

To configure output processing, do the following:

1. Create the directories for the output streams.
2. For output of rated events and AAA responses:
   - Set up an output stream format description file.
   - Set up a mapping file.
   - Set up a grammar file.

   In most cases, you need to modify only the sample stream format description and output mapping files.
3. Configure these output sections in the registry:

- The output mapping in the **DataDescription** section. This specifies the output mapping for rated events. See "Configuring the Output DataDescription Registry Section".
- The **OutputBuffer** section in the **Pipeline** section.
- The **Output** section. This section includes the configuration for the pipeline output controller and for output modules such as the OUT_GenericStream module. See "About the Output Modules".

4. Configure the output stream for the function modules that create the output. For example, configure the FCT_Reject module to send rejected EDRs to the reject output stream.

About Configuring the Output Section in the Registry

The output section in a pipeline has this hierarchy:

```
Pipeline
  Output
      Output controller
         Output collection
             Output streams
```

- The *output controller* manages the overall output process. You can configure output properties that apply to all streams. See "Output Controller".
- *Output collection* defines all the output streams for the pipeline. You configure the Output Collection module by editing the **OutputCollection** section of the registry file. For information, see "Output Collection".
- Output streams send EDRs to the appropriate output location. See "About the Output Modules".

About Configuring Statistics Information in the Output Section

The **Statistic** subgroup controls the statistics related to Pipeline Manager's EDR processing rate. You can view these statistics in the output logs, HTTP browser, and the console output from the SNMP binaries.

For more information regarding the use of the SNMP binaries to view performance statistics, see "SNMP Utilities" in *BRM System Administrator's Guide*.

For more information regarding the use of HTTP browser to view performance statistics, see "Monitoring Pipeline Manager EDR Throughput" in *BRM System Administrator's Guide*.

The **Statistic** subgroup has one registry option, **EDRCountCriteria**. This option can have two possible values:

- INPUT: The Pipeline Manager considers only the detail CDRs that are passed through it as input for calculating the EDR statistics.
- ALL: This is the default option. The Pipeline Manager considers all the EDRs that are passed through it for calculating the EDR statistics. This includes the duplicate EDRs created and the EDRs directed to an additional output stream.

Note: The **Statistic** subgroup is optional. If this subgroup is not present, the behavior of the Pipeline Manager is similar to that when **EDRCountCriteria** is set to ALL.

This sample shows the output hierarchy:

```
Pipelines
{
    W_SAMPLE
    {
        Output
        {
            WriteDefaultEdr = False
            MaxErrorRates
        {
        }
          #The following subgroup is optional
            Statistic
            {
                    EdrCountCriteria = ALL
            }
            OutputCollection
            {
                Output stream 1
                {
                    ModuleName = OUT_Module_1
                    ...
                }
                Output stream 2
                {
                    ModuleName = OUT_Module_2
                    ...
                }
            } # end of Output
        } # END W_SAMPLE
    } # END Pipelines
```

Configuring Output for Rated Events and AAA Responses

To send rated events from pipeline rating and responses from AAA request processing, you configure the OUT_GenericStream module.

For pipeline rating, this module converts EDRs to the output format used by RE Loader. The output is a file that is loaded by RE Loader.

To convert data from EDR format to a file, the module uses the following files:

- **Stream format description**
- **Output grammar:** Specifies which records to include in the output.
- **Output mapping**: Specifies how to map the data in EDRs to data in the format defined by the output grammar file.

The sample registry files include stream format description, output grammar, and output mapping files that convert data from the BRM EDR, TAP, and CIBER formats.

To configure the OUT_GenericStream module, you specify the following:

- The output grammar file.
- The BRM event type that the output file contains, such as **/event/delayed/session/gsm**.
- The type of pipeline, for example, rating or backout.

- Whether to delete empty output streams. See "Configuring Pipeline Manager to Delete Empty Output Streams".
- The output module, either "EXT_OutFileManager".

See "OUT_GenericStream".

Creating Separate Output Streams for Each Service

Events from different services (GSM, SMS, and so forth) must be delivered to the RE Loader or a prepaid network in separate files. To do this:

- Configure the IRL_EventTypeSplitting iScript to split EDRs by service code. See "Sending EDRs to Pipeline Output Streams".
- In the registry, configure an instance of the OUT_GenericStream module for each service.

Creating Multiple Output Streams in One Output Registry

Each EDR must have a default output stream and may also contain additional output streams. Use the following iScript functions to manipulate multiple output streams in a single registry:

- "edrAddAdditionalStream" in *BRM Developer's Reference*.
- "edrRemoveAdditionalStream" in *BRM Developer's Reference*.
- "edrGetAdditionalStream" in *BRM Developer's Reference*.
- "edrContainsAdditionalStream" in *BRM Developer's Reference*.

To add output streams to the default stream in the sample registry:

1. Create an iScript file that adds the stream.

   The **edrAddAdditionalStream** iScript document contains an example **addoutmod.isc** iScript file that shows how to add two additional output module streams.

2. Reference this iScript file in the pipeline FunctionPool registry section.

   The **edrAddAdditionalStream** iScript document contains an example function registry section that shows how to use the **addoutmod.isc** file to add an output stream in the function registry.

3. Define the stream in the pipeline output registry section.

   The **edrAddAdditionalStream** iScript document contains an example output registry section that shows how to configure additional output streams by using OUT_GenericStream.

See **edrRemoveAdditionalStream** for an example that shows how to remove an EDR output stream.

Configuring the Output DataDescription Registry Section

You configure a **DataDescription** section in the registry for each pipeline. The **DataDescription** section specifies the stream format description and input mapping files (see "About the Input Process") and the output mapping file.

```
DataDescription
  {
   Standard
```

```
    {
    ModuleName = Standard
    Module
    {
    StreamFormats
    {
      Format1 = ./formatDesc/Formats/Flist/Flist_v01.dsc
    }
    InputMapping
    {
     Mapping1 = ./formatDesc/Formats/Flist/Flist_v01_InMap.dsc
    }
    OutputMapping
    {
     Mapping1 = ./formatDesc/Formats/Flist/Flist_v01_OutMap.dsc
    }
    }
    }
}
```

About the Order of Listing Stream Format Description Files

Pipeline module instances prioritize the order in which formats are considered for parsing in the order that the stream format description files are listed in the **StreamFormats** section of the registry.

Parsing errors can occur in a pipeline if the stream format description files are listed in the incorrect order. For example, the stream format description file that applies to your custom input stream would be listed in the **StreamFormats** section *before* the output stream format description file.

Configuring Output for Rejected or Duplicate EDRs

The OUT_Reject module writes rejected EDRs to a reject file. A rejected EDR is identical to the EDR input.

To configure output for rejected EDRs, do the following:

- For rejected EDRs, configure the FCT_Reject module.

  See:

  - Configuring Standard Recycling
  - Recycling EDRs in Pipeline-Only Systems

- For duplicate EDRs, configure the FCT_DuplicateCheck module. See "Handling Duplicate EDRs".

- In the registry, configure an instance of the OUT_Reject module for rejected EDRs and an instance for duplicate EDRs. See "OUT_Reject".

  You can use the same output directory with different file suffixes and/or prefixes for rejected or duplicate EDRs.

The OUT_Reject module is a submodule of the Output Collection module. All registry parameters and error messages are handled by the Output Collection module. See "Output Collection".

File handling for rejected and duplicate EDRs is performed by the EXT_ OutFileManager module. See "Sending Output to a File".

Sending Output to a File

To send output to a file, use the "EXT_OutFileManager" module. The EXT_OutFileManager module handles file prefixes, suffixes, and paths for the OUT_GenericStream and OUT_Reject modules.

The EXT_OutFileManager module writes the output data to a temporary data file. When the TAM reports a successful completion of a transaction, the file is renamed to an output file and the temporary data file is removed.

If a transaction is rolled back, the original input is restored, and the temporary data file is moved to an error directory and renamed with an error prefix and/or suffix.

By default, the EXT_OutFileManager module is configured to delete empty output files. There is a short period of time during which empty output files are renamed and deleted. If another process, such as one that manages output files, tries to manipulate the file during this period, the pipeline may experience errors. To avoid this problem, configure any output file management processes to wait approximately one minute before attempting to manipulate files.

Configuring the Temporary File Name

Use the **TempPrefix** registry entry to specify the prefix for temporary data files.

> **Important:** Do not change the **TempDataPrefix**, **TempDataSuffix**, and **TempDataPath** registry entries. These entries are used by the pipeline for internal data processing only.

See "EXT_OutFileManager".

Configuring File Prefixes and Suffixes

Use the **OutputPath**, **OutputPrefix**, and **OutputSuffix** registry entries to manage the output files for each stream.

> **Important:** To ensure output file integrity, specify a unique combination of **OutputPath**, **OutputSuffix**, and **OutputPrefix** values for each output stream defined in the registry.

See "EXT_OutFileManager".

Creating an Output File Name from the Input File Name

Use the **UseInputStreamName** registry entry to specify to use the input file name to build the output file name.

For example, when **UseInputStreamName = [2,4;4,6;8,&]**, the following characters from the input file name are used to build the output file name:

- Characters 2 to 4.
- Characters 4 to 6.
- Characters 8 to end of string, where the '&' symbol indicates the end of the string.

For instance, if the name of the input file is **test12345678**, **Outputprefix** is **test**, and **Outputsuffix** is **.edr**, the output file name will be **testestt1245678.edr**

See "EXT_OutFileManager".

Applying a Prefix to the Sequence Number

Use the **SequencerPrefix** registry entry to specify a prefix to the sequence number before it gets appended to the generated output file name.

> **Note:** This entry is used only when **AppendSequencerNumber** is set to **True**.

For example, when **SequencerPrefix** is "+", **Outputprefix** is test, **Outputsuffix** is **.edr**, the output file name will be **test+000002.edr**.

By default, the **SequencerPrefix** is "_" and if no **SequencerPrefix** is needed, it has to be specified as **SequencerPrefix** = ""

> **Important:** Do not use the characters #, $, =, /, or \ to specify SequencerPrefix.

Using the Output of One Pipeline as the Input to Another Pipeline

If you want to use the output of one pipeline as the input for another pipeline, you must move the output file to another directory before it can be used by the next pipeline. You move the files by using Event Handler and an external script. Event Handler would run your external script when prompted by a specified internal event. The external script would move the output file from one directory to another. For more information, see "Using Events to Start External Programs" in *BRM System Administrator's Guide*.

For example, if you want to use the output file from pipeline A as the input file to pipeline B:

- Configure pipeline A to generate output files in directory 1.
- Configure pipeline B to process input files from directory 2.
- Create a simple script (*ExternalScript*) that moves the output files from directory 1 to directory 2.
- Configure Event Handler to run *ExternalScript* whenever pipeline A generates an output file. For example, you can configure Event Handler to run *ExternalScript* when it receives an EVT_OUTPUT_FILE_READY event from the "EXT_OutFileManager" module.

Sending Output to a Database

Use the OUT_DB module to send data to a database. You configure the following:

- A SQL command parameter file
- Files that define the following:
  - SqlBeginStream statement
  - HEADER records
  - DETAIL records
  - TRAILER records

  - SqlEndStream statement
- The OUT_DB module

The OUT_DB module reads the parameter file for each new output stream.

When you configure the OUT_DB module, you configure the following:

- The database to load data into.
- Path and file names for the configuration files.
- Aliases for the stream name and row number.
- Source and destination for the header record.
- File management options.

For more information about the OUT_DB module registry entries, see "OUT_DB".

About the OUT_DB Module Configuration Files

The parameter file contains key and value pairs. The keys are used by other configuration files (**SqlBeginStream**, **HeaderTableDefinition**, **DetailTableDefinition**, **TrailerTableDefinition**, **SqlEndStream**). The syntax to recognize keys is ${KEY}. When the other files are processed, all found keys are replaced by the corresponding values. The files are not deleted.

It is possible to change the database tables in a running system if there are no open streams.

The **SqlBeginStream** file and the **SqlEndStream** file are used to define SQL statements that are passed to the database. The begin stream statement is executed before the first EDR arrives, and the end stream statement is executed after the last EDR is processed.

The **HeaderTableDefinition**, **DetailTableDefinition** (only if the **NumberOfRows** registry entry is set to **1**), and **TrailerTableDefinition** files are used to describe the table in which the EDRs must be stored. For more information, see "HEADER, TRAILER, and DETAIL Table Definitions"

First the table name is defined with ${TABLE}, which is replaced by the definition in the parameter file. Then the column names of the table are defined. Each line holds a column of the table and its EDR container field, which is mapped into this column. The value is the EDR container field with external and alias names delimited by a comma (**,**).

```
${HEADER_TABLE}
;RECORD_TYPE      = HEADER.RECORD_TYPE    , HDR_RECORD_TYPE
```

Each column starts with the **FieldDelimiter** after the table name.

Because schema definition is not supported by the **RWDBBulkInserter**, when the **NumberOfRows** registry entry is set to **1**, the columns are defined like the example above. For bulk insertion into the database when **NumberOfRows** is greater than 1, the column definition is deleted from the table definition file and the schema of the table definition in the database is used to create the **BulkInsert**. The following example shows the DETAIL table definition file for **BulkInsert**:

```
${DETAILTABLE}
;DETAIL.RECORD_TYPE   , RECORD_TYPE
;DETAIL.RECORD_NUMBER , RECORD_NUMBER
...
```

In this case, each sequence of the EDR container field definition in the table definition file must be in the same order as the column definition in the database table.

Each logical block must end with two number signs (##) except within the table definition file. A comment line must start with two slashes (//).

All these files result in one final configuration file for each stream with replaced parameter keys within the SQL statements, except optional registry keys. The registry keys (**StreamNameAlias** and **RowNumAlias**) are replaced immediately before the SQL statement is passed to the database.

The name of the final configuration file is the stream name. It is located in the ControlPath. If the **SaveConfigurationFile** registry entry is enabled, a suffix is added. If processing is successful, the suffix is **.done**; otherwise, it is **.err**.

Specifying the Destination

The destination value can be applied to an appropriate field in the HEADER record of an output module. In the BRM format, this entry is used to fill the recipient field in the HEADER record.

Specifying the Source

The source value can be applied to an appropriate field in the HEADER record of an output module. In the BRM format, this entry is used to fill the sender field in the HEADER record.

Handling Empty Output Streams

Because of splitting and rejecting, there is a possibility that an output stream may contain only the HEADER and TRAILER records. In this case, the production of a default DETAIL record can be forced. If the **WriteDefaultEdr** registry entry is set to **True**, every DETAIL table contains at least one DETAIL record.

If the **DeleteWithoutDetails** registry entry is set to **True**, all insert and update operations will be rolled back and the default record will be deleted.

These settings are ignored if the output is for a reject stream.

Parameter File

Items in bold text are parameter keys.

```
DETAIL_TABLE
sol42_out
##
HEADER_TABLE
sol42_out_header
##
TRAILER_TABLE
sol42_out_trailer
##
```

HEADER, TRAILER, and DETAIL Table Definitions

HeaderTableDefinition

The format of **HeaderTableDefinition** does not depend on the **NumberOfRows** registry entry. The format of **HeaderTableDefinition** is as follows:

```
${HEADER_TABLE}
;RECORD_TYPE      = HEADER.RECORD_TYPE    , HDR_RECORD_TYPE
```

TrailerTableDefinition

The format of **TrailerTableDefinition** does not depend on the **NumberOfRows** registry entry. The format of **TrailerTableDefinition** is as follows:

```
${TRAILER_TABLE}
;RECORD_TYPE      = TRAILER.RECORD_TYPE   , TRR_RECORD_TYPE
```

DetailTableDefinition

The format of **DetailTableDefinition** depends on the **NumberOfRows** registry entry. When NumberOfRows is set to **1**, the format of **DetailTableDefinition** is as follows:

```
${DETAIL_TABLE}
;record_type      = DETAIL.RECORD_TYPE   , RECORD_TYPE
;record_number    = DETAIL.RECORD_NUMBER , RECORD_NUMBER
```

When NumberOfRows is greater than **1**, the format of **DetailTableDefinition** is as follows:

```
${DETAIL_TABLE}
;DETAIL.RECORD_TYPE   , RECORD_TYPE
;DETAIL.RECORD_NUMBER , RECORD_NUMBER
```

SqlBeginStream

Identifiers in bold text are registry entries. They are replaced before the statement is passed to the database.

One statement:

```
insert into stream_process (streamname, startdate,runmode,numberofrow,destination,source,info)
values
// this is a comment line
(__StreamName__, sysdate,'Debug',500,'${TABLE}','sol42_detailin','testing')
```

More statements:

```
begin
insert into tab1 (col1, col2) values (val1, val2);
insert into tab2 (col1, col2, col3) values
(val1, val2, val3);
// if an SQL block is used there must be a semicolon at the end
end;
```

SqlEndStream

Identifiers in bold text are registry entries. They are replaced before the statement is passed to the database.

```
update stream_process set enddate = sysdate,
edrnum = __RowNum__,
// duration only valid, if startdate and sysdate within one day
duration=
(3600*to_char(sysdate,'HH24')+60*to_char(sysdate,'MI')+to_char(sysdate,'SS'))-
(3600*to_char(startdate,'HH24')+60*to_char(startdate,'MI')+to_char(startdate,'SS'))
where
streamname = __StreamName__
```

Generated Configuration File

```
SqlBeginStream
insert into stream_process (streamname, startdate,runmode,numberofrow,destination,source,info)
```

```
values
(__StreamName__, sysdate,'Debug',500,'sol42_out','sol42_detailin','testing')
##
HeaderTableDefinition
sol42_out_header
##
DetailTableDefinition
sol42_out
##
TrailerTableDefinition
sol42_out_trailer
##
SqlEndStream
update stream_process set enddate = sysdate,
edrnum = __RowNum__,
duration=
(3600*to_char(sysdate,'HH24')+60*to_char(sysdate,'MI')+to_char(sysdate,'SS'))-
(3600*to_char(startdate,'HH24')+60*to_char(startdate,'MI')+to_char(startdate,'SS'))
where
streamname = __StreamName__
##
```

5

Configuring EDR Preprocessing

This chapter describes how to configure the modules used for preprocessing event data records (EDRs) in the Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager; for example, it describes how to discard duplicate EDRs.

Before reading this document, you should understand how Pipeline Manager works and how to configure it. See the following:

- "About Creating a Price List" in *BRM Setting Up Pricing and Rating*
- About Pipeline Rating
- "Configuring Pipeline Manager" in *BRM System Administrator's Guide*

For information about recycling EDRs, see "About the EDR Recycling Features".

Handling Duplicate EDRs

Use the FCT_DuplicateCheck module to find EDRs that have already been processed and send them to a special output stream. This prevents you from charging a customer twice for the same usage.

As a pipeline processes EDRs, the following occurs:

1. The FCT_DuplicateCheck module keeps a record of EDRs that have already been processed.
2. As new EDRs are processed, the module checks for duplicate EDRs by comparing data in the incoming EDRs with data in the EDRs that have already been processed.

   The FCT_DuplicateCheck module uses the following criteria to check for duplicate EDRs:

   - **The date of an EDR.** If an EDR is older than a certain date, it is not processed and no further checking is performed. It is unlikely that a duplicate EDR will be processed much later than an original EDR.
   - **The data in an EDR.** You configure which data to use when comparing a new EDR against EDRs that have already been processed.
   - **A search key**. An EDR is considered a duplicate if it has the same search key and the same values contained in the fields used for data comparison.
3. If an EDR is a duplicate, it is flagged with an error so other modules do not need to process it, and it is moved to a separate output stream.

Configuring Duplicate EDR Checking

To enable duplicate EDR checking, configure the FCT_DuplicateCheck module. See "FCT_DuplicateCheck" and the following topics:

- Setting Date Parameters for Storing Processed EDRs
- Specifying the Fields to Use for Duplicate Check
- Specifying a Search Key for Duplicate Check
- Managing FCT_DuplicateCheck Data Files
- About Storing EDRs in a Database Instead of Files
- Using Duplicate Check with Multiple Pipelines
- Suspending Duplicate EDRs

To define the output stream for duplicate EDRs, use the **Output** configuration in the registry. You typically use the OUT_Reject module. See "Sample Output Configuration" and "OUT_Reject" for more information.

Important: When you enable duplicate checking, you must also set the transaction manager **RedoEnabled** entry to **True**. See "About Cancelling Transactions When a Rollback Occurs" in *BRM System Administrator's Guide* and "Transaction Manager" for more information.

Setting Date Parameters for Storing Processed EDRs

To check for duplicate EDRs, you need to store a record of the EDRs that have already been processed. The FCT_DuplicateCheck module then checks incoming EDRs against the previously processed EDRs. If you receive a high volume of EDRs, you cannot store a record of *all* EDRs. Therefore, to limit the number of EDRs you store, you specify date criteria. If an EDR is older than the specified date, it is not processed.

Note: The date of an EDR is derived from its DETAIL.CHARGING_START_TIMESTAMP field.

To specify date parameters for EDR storage, you use the following date settings, specified in the FCT_DuplicateCheck registry. Depending on an EDR's date relative to these settings and on whether the FCT_DuplicateCheck module is connected to the database, the EDR is ignored, stored in a file, stored in the database, or stored in memory.

- **StoreLimit**: Specifies the oldest date that previously-processed EDRs can be stored. EDRs dated earlier than the **StoreLimit** date are ignored and not processed by the FCT_DuplicateCheck module.
- **BufferLimit**: Specifies the oldest date that previously-processed EDRs can be stored in memory. All EDRs whose date is equal to or later than the **BufferLimit** date are stored in memory. The FCT_DuplicateCheck module searches the memory directly, thus improving performance.

The **StoreLimit** date must be equal to or earlier than the **BufferLimit** date. For example, if the **StoreLimit** is June 1, the **BufferLimit** date can be June 1 or later.

Important: Because **StoreLimit** and **BufferLimit** are specified as absolute dates (for example, August 2, 2004), you need to change them daily. You change them by using an FCT_DuplicateCheck semaphore file entry. See "FCT_DuplicateCheck" for more information.

Specifying the Fields to Use for Duplicate Check

You configure which data to use when comparing a new EDR against EDRs that have already been processed. A typical duplicate check for a phone call compares the A number, B number, and start time in a new and processed EDR. If the data in all fields match, the new EDR is flagged as a duplicate.

To specify which fields to use, use the FCT_DuplicateCheck **Fields** registry entry. See "FCT_DuplicateCheck" for more information.

The following example shows a typical configuration:

```
Fields
{
    1  =  DETAIL.BASIC_SERVICE
    2  =  DETAIL.B_NUMBER
}
```

Important: Do not use the DETAIL.CHARGING_START_TIMESTAMP field for duplicate checking.

Specifying a Search Key for Duplicate Check

The duplicate check search key identifies duplicate EDRs. An EDR is considered a duplicate if it has the same time, the same search key value, and the same values for fields listed in the IFW_DUPLICATECHECK table or in the **Fields** lists stored as a record in memory.

The search key is used as a key to the internal data structure. For example, if the search key is **A_NUMBER**, then **A_NUMBER** is the hash key used to find the data in memory or in a file for the EDR being checked.

Managing FCT_DuplicateCheck Data Files

The FCT_DuplicateCheck module uses data files to store EDR data while the EDRs are processed. The file name syntax is:

File_Name_Transaction Id **.dat**

File_Name is the value entered in the **FileName** registry entry:

```
FileName = duplicateData
```

The file contains the data from the fields specified in the registry and the EDR date. At the end of each transaction, the FCT_DuplicateCheck module saves the data from memory to disk in a new transaction file.

Note:

- To store EDRs in files for duplicate checking, configure unique **FileName** settings, **Path** registry settings, or both for each module.
- To store EDRs in the database for duplicate checking, use the FCT_DuplicateChecking module's **DataConnection** registry entry to connect the module to the database. See "About Storing EDRs in a Database Instead of Files".

The duplicate check transaction files should be backed up routinely when the pipeline is not processing EDRs. Temporary files, however, should not be backed up.

To restore transaction files from a backup, shut down the pipeline and restart after restoring backed up transactions files. You can restore duplicate check data without shutting down provided that the pipeline is not processing any EDRs. There is no semaphore to reload the data file, but resetting the **StoreLimit** or **BufferLimit** settings results in a reload. The values of these settings do not need to be changed from their original startup registry settings.

After an abnormal termination, temporary files may be left behind (there should be only one because file mode should only be used with a single pipeline). These files correspond to transactions that were never committed, and the input files associated with these transactions will be reprocessed upon restarting. You should delete these temporary files before restarting. Temporary files use the suffix **.tmp**.

For information about managing transaction rollback files, see "About Pipeline Manager Transactions" in *BRM System Administrator's Guide*.

About Storing EDRs in a Database Instead of Files

If you use the FCT_DuplicateCheck module's **DataConnection** registry entry to connect the module to the database (see "About Storing EDRs in a Database Instead of Files"), the module handles EDRs as follows:

- EDRs whose date is equal to or later than the **StoreLimit** date and earlier than the **BufferLimit** date are stored in the database instead of in files. If an EDR is stored in the database, files are not created.
- EDRs whose date is equal to or later than the **BufferLimit** date are stored in memory and in files.

Tip: To avoid using excessive disk space when checking for duplicate EDRs, use the FCT_DuplicateCheck module's **DataConnection** registry entry to connect the module to the database. (The **StoreLimit** and **BufferLimit** date settings, and connecting the module to the database, are designed to help maintain performance without overloading memory.)

When the FCT_DuplicateCheck module is connected to the database, an entry is added to the IFW_DUPLICATECHECK database table for each unique EDR. For each duplicate EDR, the error INF_DUPLICATE_EDR is reported and no entry is added to the table.

If the pipeline transaction is cancelled, all the rows with the current transaction ID are removed.

If you do not use the FCT_DuplicateCheck module's **DataConnection** registry entry to connect the module to the database (the default), the module handles EDRs as follows:

- EDRs whose date is equal to or later than the **StoreLimit** date and earlier than the **BufferLimit** date are stored in files instead of the database. (Each EDR is assigned a value that is stored in memory. If the EDR is stored in files, the module checks the memory for the value, which points to the EDR in the file. The module then reads the EDR data from the file.)
- EDRs whose date is equal to or later than the **BufferLimit** date are stored in memory and in files.

> **Note:** Although not connecting the module to the database enables faster checking for duplicate EDRs, it uses a large amount of disk space.

Create Database Tables for Duplicate Check Data

Use the FCT_DuplicateCheck **TableSuffix** registry entry to create multiple IFW_DUPLICATECHECK tables when you run multiple pipelines. You typically do this when you run a separate pipeline for each type of service.

For example, if the pipeline processes GSM EDRs, you can use GSM as the table suffix to use a table named IFW_DUPLICATECHECK_GSM.

You need to manually create the tables and the indexes. For example:

- IFW_DUPLICATECHECK_GSM
- BIDX_DUPCHK_DATA_GSM

Using Duplicate Check with Multiple Pipelines

When you use the FCT_DuplicateCheck module, you must process the EDRs for each account in the same pipeline. If you use duplicate check in multiple pipelines for the same account, duplicate calls may get processed by different pipelines and will not be recognized as being duplicate.

Suspending Duplicate EDRs

By design, a duplicate EDR is not considered as a suspended EDR and is sent to a different output stream than suspense handling stream. Therefore, duplicate EDRs are not handled by suspense handling.

If your business requires to process duplicate EDRs as suspended EDRs, you can do the following:

1. Change the entry **StreamName = DuplicateOutput** to **StreamName = SuspenseCreateOutput** in the FCT_DuplicateCheck registry.

   This will cause the duplicate check reject file to be routed to the suspense handling stream.

2. Add an iScript in the pipeline after duplicate check processing, to check for EDR error. If the error is duplicate check error, set the error to some other error and set error status as major.

   This will cause the duplicate EDR to be processed by FCT_Reject and FCT_Suspense plugins.

Note: You should ensure that duplicate EDRs are associated with a specific error code to be able to manage and monitor the EDRs in Suspense Manager Center.

Assembling EDRs

Use the FCT_CallAssembling module to assemble multiple CDRs into a single EDR that pipeline modules can rate. You typically need to assemble CDRs for long phone calls or GPRS sessions that have been recorded in multiple CDRs.

The default behavior of FCT_CallAssembling is designed to assemble *time duration* calls. This is appropriate for wireless voice calls that are rated based on how long the call lasts. You can also configure this module to assemble calls in a manner appropriate for EDRs rated based on the *volume of data sent*. This is appropriate for a long data transfer session, such as downloading a movie. You can choose to collect both time duration and data volume from multiple CDRs, and a number of other matrixes, such as:

- Volume sent
- Volume received
- Number of units
- Retail charge amount
- Wholesale charge amount

For example, a GPRS session might last for 24 hours. The network might be configured to generate an intermediate CDR every 30 minutes. This GPRS data session is recorded by several partial CDRs. If you rate by volume per session, you use the FCT_CallAssembling module to assemble the partial CDRs into one EDR before rating.

However, remember that the more metrics that you collect, the more system resources you will use.

How FCT_CallAssembling Classifies EDRs

FCT_CallAssembling uses the LONG_DURATION_INDICATOR EDR field to classify assembled calls.

The purpose of LONG_DURATION_INDICATOR changes when calls are assembled. CDRs arrive with this field set to one of these values, which identify the type of call segment:

- **F**: The first segment of the call.
- **L**: The last segment of the call.
- **I**: An intermediate segment of the call.

After FCT_CallAssembling assembles these call segments into an EDR, it gives the EDR one of these long duration indicators to specify its status.

- **C**: Complete call. The first and last segments have both arrived, enabling the call's duration to be calculated. If intermediate call segments arrive after the call is complete, they are given a long duration indicator of **XC**, **XO**, or **XP** and may not be rated.
- **S:** A single CDR containing the entire call.

- **SL**: Slice of a call. Used if **KeepCallOpen = True** or **MaxDuration = True**. Composed of the first call segment and any intermediate call segments that have arrived when the call is flushed. This part of the call is rated. When any other intermediate segments and the last segment arrive, the call is given a long duration indicator of **C** (complete), and these segments are rated.
- **P**: Partially assembled call. Used if **FlushLimit = True** or **KeepCallOpen = false**. Partially assembled calls are rated with the information in whatever segments have arrived. Subsequent call segments are given a long duration indicator of **XP** and are not rated.
- **XC:** Late intermediate EDR. This long duration indicator is for intermediate call segments that arrive after the call is marked complete and rated. Late intermediate call segments are not rated; instead they are used for auditing.
- **XO**: Late overlap EDR. Used if time duration rating is used, and **DropLateCDRs=False**. This status indicates that the call segment was flushed before this CDR arrived, and it represents a time duration period already rated. These late segments are not rated; instead they are used for auditing.
- **XP**: Late timeout EDR. Used if **DropLateCDRs** = **False**. This status indicates that the call timed out before this CDR arrived. These late segments are not rated; instead they are used for auditing.

Managing the Call Assembling Data Files

Open EDRs are stored in a data file. You configure the path and file name in the **Path** and **FileName** entries in the FCT_CallAssembling registry. The syntax for the file name is:

*File_Name_Transaction_ID***.dat**

This file is read at startup and reloaded after rollback. While processing, data is stored in a temporary file.

The FCT_CallAssembling module processes the EDRs in a single transaction. You should backup the work files routinely when the pipeline is not processing any EDRs.

To restore from a backup, shut down the pipeline and restart after restoring backed up work files. Work files use the suffix **.dat**.

After abnormal termination, temporary files may be left behind. Temporary work files use the suffix **.tmp**. These correspond to transactions that never committed. Therefore, the input files associated with these transactions will be reprocessed upon restarting. Delete these temporary files before restarting.

Important: Never delete the most recent **.dat** file.

For information about managing transaction rollback files, see "About Pipeline Manager Transactions" in *BRM System Administrator's Guide*.

Configuring Call Assembling

To configure call assembling, see "FCT_CallAssembling" and the following topics:

- Rating Calls by Time Duration
- Rating Calls by Implied Time Duration
- Rating Calls by Volume of Data Sent

In addition to specifying how to perform call assembly, you can configure FCT_CallAssembling to do the following:

- Specify an amount of time (in seconds) that is an acceptable amount of time error for each call. See "Specifying a Time Error" for more information.
- Keep calls open indefinitely, and rate them in segments periodically. See "Rating Continuous Data Calls by Segment" for more information.
- Limit the effect of **FlushLimit** to calls with specific service codes. See "Rating Partial Calls by Service" for more information.
- Capture data from **Basic Detail Record** fields for the **L** call segment. See "Capturing Fields From the Last Call Record" for more information.
- Get a report about calls being assembled. See "Tracking the Status of Assembled Calls" for more information.

If you are upgrading, see "Migrating Call Assembling Data Between Releases and Pipelines".

Rating Calls by Time Duration

By default, the FCT_CallAssembling module assembles the time duration of a call so it can be rated.

To assemble EDRs for their time duration, the FCT_CallAssembling module identifies partial EDRs by using the following EDR attributes:

- **The chain reference.** This ID identifies which event the partial EDR belongs to. Multiple partial EDRs that belong to the same event all have the same chain reference.

  Important: Chain reference must be unique for each call instance. Oracle does not support identical chain references across several call events.

- **The long duration indicator.** For a list of these indicators, see "How FCT_CallAssembling Classifies EDRs".

Parts of EDRs can be processed out of order; for example, the Last segment might arrive before the First segment. The FCT_CallAssembling module manages EDRs by tracking their status:

- As a soon as a first or intermediate call segment record arrives, the EDR is stored in a data file and the state is set to *Open*. See "Managing the Call Assembling Data Files".
- By default, if a First or Last segment is already stored in a file, and the matching Last segment or First segments arrive, the record state is changed to *Closed*. The EDR is moved back into the pipeline. Any Intermediate segment that belong to the assembled EDR that arrive after the assembled EDR is closed are ignored.

  You can also direct this module to wait for all CDRs before closing a call, or close it after you send in a semaphore. This allows you to rate incomplete calls or calls that never receive a first or last record.

The time duration for a call is calculated as follows:

```
Call Duration = [(Start_Time_Of_Last_Segment) - (Start_Time_Of_First_Segment)] +
Duration of the Last Segment
```

For example, if there is a call which starts at Jan 01, 2009 12:00:00 PM and ends at Jan 01, 2009 at 12:30:00 PM. The EDR will have the following information:

- First Segments TimeStamp as Jan 01, 2009 12:00:00 PM and Duration
- Last Segment TimeStamp would be Jan 01, 2009 12:25:00 PM + 300(Seconds)

In this case, the call duration will be calculated as follows:

```
[(12:25:00 PM) - (12:00:00 PM)] + 300 = 1800 Seconds
```

Rating Incomplete Time Duration Calls

Some EDRs can never be completely assembled because the First, Intermediate, or Last segment never arrives. In this situation, you can choose to close these calls using either the **FlushLimit** semaphore entry or the **MaxDuration** startup registry entry. Both of these entries force calls to be flushed and rated after the time limit you set.

Here is a comparison of the two entries:

- **MaxDuration** is a *startup* registry entry. It takes effect when you start the pipeline. Each time a new call segment arrives, it recalculates the total time duration for that call. If the new time duration exceeds the limit you set, the call is flushed and it remains open for more CDRs. You set the time duration in seconds. To change the setting, you must restart the pipeline.

  For details, see "Using MaxDuration to Rate Incomplete Calls".

- **FlushLimit** is a *semaphore* registry entry used with **KeepCallOpen=True**. It sets a maximum age a call can have before being flushed. When you send in the semaphore, the pipeline calculates whether a call exceeds the maximum age. If a call exceeds the **FlushLimit** setting, FCT_CallAssembling creates and rates the EDR and then closes the call. You set the time limit in days. A setting of **0** flushes all calls. A setting of **1** flushes all calls that have been opened more than 1 day. You can change the **FlushLimit** setting every time you send in a semaphore.

  For details, see "Using FlushLimit to Rate Incomplete Calls".

Using MaxDuration to Rate Incomplete Calls

MaxDuration is a startup registry entry that directs FCT_CallAssembling to rate segments of a wireless call periodically. This entry specifies the total amount of time duration (in seconds) that a call can have before the call segments that have arrived are rated. FCT_CallAssembling recalculates the call duration for every call each time a new call segment arrives and compares it to the **MaxDuration** setting. If the new time duration equals or exceeds the setting for **MaxDuration,** FCT_CallAssembling creates an EDR to rate the existing portion of the call.

To use the **MaxDuration** entry, add it to the startup registry and start (or restart) the pipeline.

Table 5–1 illustrates FCT_CallAssembling behavior with **MaxDuration** set. In the example, the CDRs did not arrive in chronological order.

Example call with KeepCallOpen=True and MaxDuration=28800 (8 hours)

Table 5–1 FCT_CallAssembling and CDRs

CDR	Start time	End time	Duration (hours)	FCT_CallAssembling takes this action...
F	1:00	4:00	3	Waits. **MaxDuration** setting not reached.

Table 5–1 (Cont.) FCT_CallAssembling and CDRs

CDR	Start time	End time	Duration (hours)	FCT_CallAssembling takes this action...
I1	4:00	7:00	3	Waits. **MaxDuration** setting not reached.
I2	7:00	10:00	3	Creates EDR for 9 hours of call duration. **MaxDuration** setting exceeded.
I3	10:00	13:00	3	Waits. **MaxDuration** setting not reached.
I5	16:00	19:00	3	Creates EDR for 9 hours of call duration, even though a CDR is missing. **MaxDuration** setting exceeded.
I4	13:00	16:00	3	Not rated; call duration already rated. This EDR is either emitted as **XO** or dropped, depending on the **DropLateCDRs** setting.

In this example, if **MaxDuration** is set to **28800** seconds (8 hours), FCT_CallAssembling rates the EDRs of a call with a total time duration of more than 8 hours. This call arrives in 3-hour segments and will be rated after the third segment arrives, and FCT_CallAssembling calculates that the 9-hour call duration exceeds the 8-hour limit. When the **MaxDuration** setting is reached, the call segments are flushed and rated, the long duration indicator for the call is set to **SL**, and the call is left open for more call segments.

If a CDR is missing, FCT_CallAssembling adds the missing call time represented by the EDR if it can. In the example above, the I4 call segment arrived last, but the time duration it represented had already been rated when the I5 segment arrived. FCT_CallAssembling calculated the time duration by subtracting the start time from I3 and the end time from I5. The difference was 9 hours of time duration, which exceeds the 8-hour setting, so an EDR was created to rate that duration.

Using FlushLimit to Rate Incomplete Calls

FlushLimit assembles the calls and emits them into the pipeline for rating. The call has a long duration indicator set to **P** for "Partial."

In this example, **FlushLimit** flushes all calls the have a CHARGING_START_TIMESTAMP older than five days from today.

```
FlushLimit=5
```

To flush all incomplete calls, use 0; for example:

```
FlushLimit=0
```

The flush operation does not happen immediately at the time of semaphore execution. Instead, it happens at the arrival of the next pipeline transaction. This is done to ensure that Flush operations happen within the context of a transaction.

Removing Incomplete Time Duration Calls

When you flush EDRs, they re-enter the pipeline as part of the current transaction. The EDRs are still stored in the work files (**.dat** and **.EDR**), with a state of Timeout. This prevents a late-arriving call segment from re-opening a call that has already been flushed. If you are sure that no further segments will arrive, use the **RemoveLimit** semaphore entry to remove calls from the work files.

The **RemoveLimit** entry removes all calls with the Closed or Timeout status, but leaves calls with a state of Closed_Rejected or Timeout_Rejected alone. Closed_

Rejected or Timeout_Rejected calls will be recycled. However, if you are sure that calls with the Closed_Rejected or Timeout_Rejected status will not be recycled, use a **RemoveRejectedLimit** semaphore entry to remove these calls from the work files.

Dropping Late Calls

You can drop late EDRs from the pipeline entirely, or send them through as non-valid EDRs. If you send them through the pipeline, you can use them for auditing.

Use the **DropLateCDR** registry entry to specify how to handle the output of late EDRs:

- **True** (Default) = Drop late EDRs from the pipeline. The EDRs are counted in the report of late-arriving EDRs.
- **False** = Send late EDRs through the pipeline as non-valid. The EDRs are not counted in the report of late-arriving EDRs. They are not rated.

Rating Calls by Implied Time Duration

By default, FCT_CallAssembling calculates a call's time duration by using the difference between the first EDR's start time and the last EDR's end time. This includes the time duration for any CDRs between the first and last that have not yet arrived. The time duration for these missing CDRs is included in the EDR and rated.

If **KeepCallOpen=True** calls are rated in segments. When a semaphore with **KeepCallOpen=True** is sent in:

- All the CDRs that have arrived for a call are assembled.
- An EDR is emitted and rated
- The call is kept open for more CDRs.

When the next semaphore is sent in, all CDRs that have arrived since the last semaphore are assembled as an EDR and rated.

By default, FCT_CallAssembling calculates a call's time duration by subtracting the end time of the last CDR that has arrived from the start time of the first. Because the CDRs between the first and last are not used in the calculation, it does not matter if they have arrived when the EDR is created. If a CDR between the first and last CDRs is missing when the time duration is calculated, the missing CDR is dropped when it finally does arrive.

Example time duration registry

```
CallAssembling
 {
  ModuleName = FCT_CallAssembling
  Module
   {
      Active            = True
      AssembleVolume    = False
      AssembleSGSN      = False
      SplitAtGaps       = False
      MaxDuration       = 900
      Path              = ./data/assy
      FileName          = calls
      Mode              = Normal
      RejectMissingChain = True
      CallDurationTolerance = 59
      DropLateCDRs          = False

      AssembledEDR  {
```

```
            1 = Detail.custom_fields_from_last_edr1
            2 = Detail.custom_field_from_last_edr2...
        }
    }
}
```

Note: If you use both the FCT_CallAssembling and FCT_Reject in the same pipeline, use the FCT_Reject module **CallAssemblingModule** registry entry to ensure that complete EDRs are recycled. See "FCT_Reject".

Rating Calls by Volume of Data Sent

Use the **AssembleVolume** and/or **AssembleSGSN** registry entries to direct FCT_CallAssembling to rate calls by the volume of data sent. Both **AssembleVolume** and **AssembleSGSN** ensure that you capture the entire volume of data sent for a single call, and they both direct FCT_CallAssembling to rate a call only after all of its call records have arrived.

If you do not plan to rate calls by volume of data sent, leave **AssembleVolume** set to **False**. This saves system resources by disabling the registry entries which rate by volume.

The volume-based rating entries protect against lost revenue. Call records can arrive in any order, so it is not unusual for an intermediate segment to arrive after the first and last segments have arrived and been rated. In this case, any intermediate segments that arrive after the call is closed are dropped, and all the volume they contain are lost (and so is the revenue).

By default, FCT_CallAssembling calculates the time duration for each CDR *individually*. FCT_CallAssembling subtracts the end times from the start times of each CDR to calculate the time duration. This module then adds the time durations of all CDRs as they arrive to create a grand total for the call.

If your business requires that all non-contiguous CDRs be rated as separate EDRs (TAP requires this) set the **SplitAtGaps** registry entry to **True**. If not, then set this entry to **False**, and non-contiguous CDRs will be collected into a single EDR, saving system resources.

However, if you are rating calls by volume of data sent, lost intermediate call records will cause calls to remain open indefinitely. Send in a **Flushlimit** entry frequently to avoid this.

For information on data calls that you want to keep open indefinitely and rate periodically, see "Rating Continuous Data Calls by Segment".

Example volume of data call registry

```
CallAssembling
 {
  ModuleName = FCT_CallAssembling
  Module
   {
      Active            = True
      AssembleVolume    = True
      AssembleSGSN      = True
      SplitAtGaps       = False
      Path              = ./data/assy
      FileName          = calls
      RejectMissingChain = True
```

```
      CallDurationTolerance = 59
      DropLateCDRs          = False

      AssembledEDR  {
          1 = Detail.custom_fields_from_last_edr1
          2 = Detail.custom_field_from_last_edr2...
      }
   }
 }
```

Example TAP volume of data call registry

This example adheres to the TAP 3.10 standard. Volume and SGSN data are recorded precisely at the start and end times of each call segment. Because **SplitAtGaps = True**, if any call segments are missing, the segments before and after it are emitted as separate EDRs.

```
CallAssembling
 {
  ModuleName = FCT_CallAssembling
  Module
   {
      Active            = True
      AssembleVolume    = True
      AssembleSGSN      = True
      SplitAtGaps       = True
      Path              = ./data/assy
      FileName          = calls
      RejectMissingChain = True
      CallDurationTolerance = 59
      DropLateCDRs          = False

      AssembledEDR  {
          1 = Detail.custom_fields_from_last_edr1
          2 = Detail.custom_field_from_last_edr2...
      }
   }
 }
```

Specifying a Time Error

Use the **CallDurationTolerance** startup registry entry to specify an amount of time (in seconds) that is an acceptable amount of time error for each call.

> **Important:** The default value for this entry correctly handles most calls, so you do not need to change this entry unless you notice a problem.

Mobile phone calls are commonly split into multiple call records. Each call record should start at the same second that the previous one ended. For example, a call record with an *end* time of 12:01:30 should be followed by the next call record with a *start* time of 12:01:30. Unfortunately, it is common for these start and end times to either not quite match, or to overlap slightly (time error). The reason may be that the call records come from different routing switches with clocks that have not been synchronized, or the switches themselves have difference time tolerances.

The **CallDurationTolerance** default value is 60 seconds of tolerance to compensate for this time error. If a call has less than 60 seconds of error, a call is considered complete

and sent for rating. Otherwise the call is left open and must be closed with a semaphore entry. A single call with 10 call records, each overlapping by 3 seconds, creates a call with 30 seconds of time error. This 30-second time error is less than the **CallDurationTolerance** 60-second time limit, so the call is considered complete and sent down the pipeline for rating.

Important: Setting this entry too low causes an inordinate number of calls to be left open indefinitely. Setting this entry too high can cause the pipeline to rate calls with missing call segments. The 60-second default value is appropriate for most BRM implementations.

Rating Continuous Data Calls by Segment

Use the **KeepCallOpen** semaphore registry entry together with **FlushLimit**, to keep calls open indefinitely, and rate them in segments periodically. This feature is designed for data calls that are kept open continuously (days at a time). These long data calls are usually rated periodically to capture revenue.

KeepCallOpen is an update registry entry sent in the **FlushLimit** semaphore.

For example, a bank might keep a continuous call open to each of its ATMs to pass data back and forth. Using the default behavior, the call would not be rated for days. You will probably want to capture the revenue for these types of calls periodically, perhaps every 12 or 24 hours. Setting **KeepCallOpen** to **True** keeps, these calls open. You then rate these calls in segments by sending in a **FlushLimit** semaphore entry.

If you set **KeepCallOpen** to **True** and send it in with the **FlushLimit** semaphore every 12 hours, an EDR is created twice a day, each EDR rating the previous 12 hours of call time and volume.

KeepCallOpen is an entry in the **FlushLimit** update registry entry.

By default **KeepCallOpen** is set to **False**. The default behavior directs this module to rate the call when the first and last segments have arrived, or when the **FlushLimit** is sent, whichever comes first. Any subsequent records are ignored.

Rating Partial Calls by Service

Use the **FlushServiceCode** semaphore registry entry to limit the effect of **FlushLimit** to calls with specific service codes. **FlushServiceCode** is sent as an entry in the **FlushLimit** update registry entry. If the **FlushServiceCode** entry matches the value of the DETAIL.INTERNAL_SERVICE_CODE field of the call segment, the call is flushed. All other partial calls are ignored.

For example, the following **FlushServiceCode** entry directs the pipeline to rate only calls with the **dat** service code:

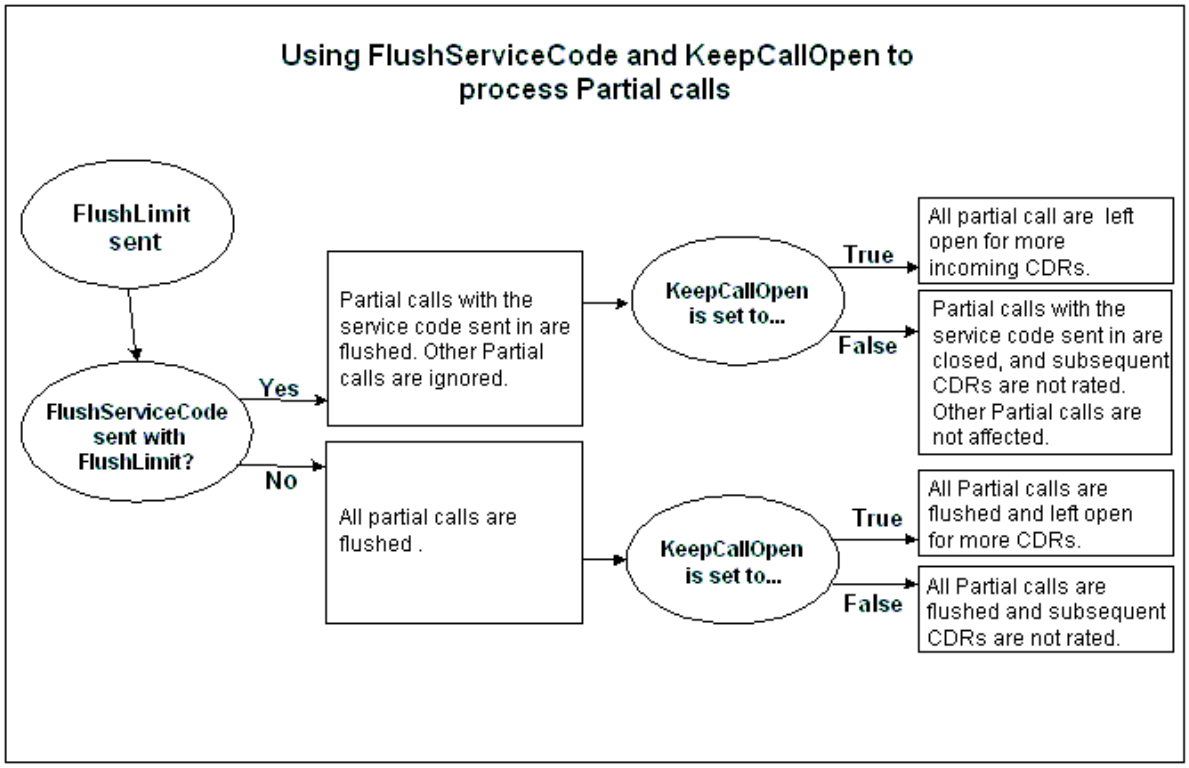
```
{

  FlushLimit=0
  FlushServiceCode = dat

}
```

Figure 5–1 shows how the **FlushServiceCode** and **KeepCallOpen** semaphore registry entries interact.

Figure 5–1 FlushServiceCode and KeepCallOpen Semaphore Interaction



Capturing Fields From the Last Call Record

Use the **AssembledEDR** startup registry entry to capture data from **Basic Detail Record** fields for the **L** call segment.

The final record of a call has a long duration indicator of **L** (last), however the **L** call record may not be the last to arrive at the pipeline. In some cases it is important that information be captured from the **L** call record. For example, the number that terminated the call can only come from the **L** call record.

If **AssembledEDR** is used, the pipeline captures **Basic Detail Record** data from the **L** call record and adds it to the EDR emitted, regardless of whether it was actually the last call record received.

List any **Basic Detail Record** fields to capture data from in the **AssembledEDR** entry, including any custom fields that your business requires. The pipeline will then include the data from those fields on the EDR that is compiled and processed. See "Sample Registry" for an example.

Tracking the Status of Assembled Calls

You can use a semaphore command to receive a report about the status of calls currently being assembled. The report provides the following information:

- The number of partially assembled calls.
- The number of EDRs currently waiting to be assembled.
- The number of late EDRs that are parts of calls that are no longer being assembled, for one of the following reasons:

- The call was assembled and sent to the pipeline. In this case, the first and last portion of a call was processed, and an intermediate part arrived after the EDR had been sent into the pipeline.
- The call was flushed by a **FlushLimit** semaphore, and another portion arrived after the call was flushed.

The report includes the following data about late EDRs:

- The number of late-arriving EDRs for flushed calls.
- The number of late-arriving EDRs for assembled calls.
- The total number of late-arriving EDRs.

In addition to creating the report, each assembled EDR includes a NUMBER_OF_CDRS field that stores the number of CDRs that were included in the EDR. You can use this data in an aggregation scenario to gather additional data about assembled calls.

Note: For EDRs that are not part of split calls, the FCT_CallAssembling module enters **1** in the NUMBER_OF_CDRS field.

Migrating Call Assembling Data Between Releases and Pipelines

When you upgrade BRM from one release to another or apply patches, you need to migrate the call data in your **.dat** files to the new format by using the XML support provided with the FCT_CallAssembling module.

For information on migrating EDR files, see "Upgrading Incomplete Calls to the New Container Description" in *BRM System Administrator's Guide*.

You use the following semaphore registry entries to change the format of the data from one release or pipeline to another:

- **ExportDataToXml**
- **ImportDataFromXml**

For more information, see the semaphore file entries in "FCT_CallAssembling".

To migrate call assembly data, perform the following steps:

1. Export the data from your existing data files to an XML file by using a semaphore registry file with the **ExportDataToXml** entry:

```
ExportDataToXml
{
   CallsPerFile = Value
}
```

2. Import the data from the XML file into the data file with the new format by using the **ImportDataFromXml** entry:

```
ImportDataFromXml
{
   FileName = filename.xml
}
```

Assembling Calls with Multiple Pipelines

When you use the FCT_CallAssembling module, you must process the EDRs for each account in the same pipeline. If you assemble calls in multiple pipelines for the same account, the call segments may get processed by different pipelines and cannot be assembled.

Discarding and Skipping EDRs

You can use the FCT_Discard module to skip or discard EDRs:

- Skipping an EDR removes it from the pipeline.
- Discarding an EDR sends it to a different output stream. You will probably want to audit the information in EDRs with a LONG_DURATION_INDICATOR of **XC**, **XO**, or **XP** before you discard them.

In both the cases the state of the EDR becomes invalid. (To indicate a discarded EDR, a value is entered in the DETAIL.DISCARDING field.)

For example, you can filter the following EDRs:

- Discard EDRs that are older than three days and have a B_NUMBER that begins with 0049.
- Discard EDRs that have a RECORD_TYPE that begins with a 9 followed by two digits, an INTERN_SERVICE_CODE that ends with a 2, and a WHOLESALE_CHARGED_AMOUNT_VALUE of 0.

For information on the regular expressions you can use, see "About Using Regular Expressions when Specifying the Data to Extract".

To create a valid mapping, the data in the EDR must match with all of the mapping data.

Configuring EDR Discarding

To configure EDR discarding:

1. Specify which EDRs to discard or skip. See "About Configuring Discard and Skip Expressions".
2. Configure the OUT_DevNull module as the output stream for discarded EDRs. See "Configuring Output of Discarded EDRs".
3. Configure the FCT_Discard module. See "FCT_Discard".
   - Use the FCT_Discard module **StreamName** registry entry to specify the output stream.
   - To send EDRs to different output streams without making them invalid, use the FCT_EnhancedSplitting module. To process EDRs that belong to accounts in different BRM database schemas, see "FCT_AccountRouter".

About Configuring Discard and Skip Expressions

To specify which EDRs to discard or skip, you create a set of regular expressions using a set of data fields. These fields are not mandatory so if they do not exist in the EDR description it does not matter. If all expressions match the available fields, the EDR will be removed.

You can create different discard rules for separate pipelines.

You define discard rules in Pricing Center. The discard rules are stored in the IFW_DISCARDING table.

Use the **Reload** semaphore file entry to reload regular expression patterns after you change them.

You can discard or skip EDRs based on the following:

- Rank. This specifies the order in which to evaluate the rules.
- Record type.
- Source network.
- Destination network.
- Call complete indicator.
- Long duration indicator.
- Usage class.
- Internal service code.
- GSM switch or GPRS switch.
- Tariff class.
- Tariff subclass.
- Connection type.
- Connection subtype.
- B number.
- The age of the EDR.
- If the wholesale charge should be zero.

For information on the regular expressions you can use, see "About Using Regular Expressions when Specifying the Data to Extract".

To configure the FCT_Discard module. See "FCT_Discard".

Configuring Output of Discarded EDRs

Use the OUT_DevNull module to handle EDRs that should be discarded from a pipeline.

To discard EDRs, do the following:

- Configure the FCT_Discard module. See "Discarding and Skipping EDRs".
- In the pipeline, configure the OUT_DevNull module. See "OUT_DevNull".

OUT_DevNull is a submodule of the Output Collection module. All registry parameters and error messages are handled by the Output Collection module. See "Output Collection".

Generating Multiple TAP MOC and MTC Records

When you process TAP files, you use the ISC_TapSplitting iScript to splits mobile originating and terminating EDRs into multiple EDRs when the CDR contains more than one basic service. The ISC_TapSplitting creates a new EDR for each additional basic service.

Splitting mobile originating and terminating EDRs enables CDR rejection when a basic service record associated with a CDR is in error. It also permits custom validations to be added prior to splitting.

Service information for all secondary services (supplementary, VAS, CAMEL), which are part of a basic service EDR, are added to the last basic service EDR. Charge information for these secondary services are added to the last charge breakdown record of the last basic service EDR.

EDR splitting is performed after TAP validation. TAP files and EDRs that are rejected due to errors are not split. After the new EDRs are created, the original EDR is deleted.

To configure ISC_TapSplitting, see "ISC_TapSplitting".

Using Rules to Send EDRs to Different Output Streams

Note: To send EDRs to different output streams for the Rated Event (RE) Loader, use the IRL_EventTypeSplitting iScript. See "Sending EDRs to Pipeline Output Streams".

Use the FCT_EnhancedSplitting module to specify different output streams for EDRs based on rules. For example:

- You can split EDRs for different service types into different output streams.
- You can split EDRs from roaming outcollects and incollects into different streams. See "About Rating Roaming Events" in *BRM Configuring Roaming in Pipeline Manager*.

To send EDRs to different output streams, you define a set of rules. For example, you can send all telephony EDRs from a specific network to a different output stream.

You can use the following data in the rules:

- Record type.
- Service code.
- Usage class.
- Source and destination network.
- Switch.
- Trunk in/out.
- A number and B number area code.
- Normalized C number area code (forwarded or routed number).

This example assigns all telephony EDRs with an A number starting with 49 to the Out49 system brand:

```
Service code :   TEL
A number:        0049.*
System brand:    Out49
```

In addition you can specify the order in which to evaluate the rules, and the dates when the rule is valid.

The FCT_EnhancedSplitting module evaluates each EDR against the rules. The first rule that matches the criteria defines the system brand to use. The system brand is

identified by a code. You use that code to map the system brand to an output stream in the registry. In this example, system brand **Out49** is mapped to the **EdrOutputOut49** output:

```
SystemBrands
{
Out49 = EdrOutputOut49
}
```

You can create separate splitting rules for different pipelines.

You can use a semaphore file entry to reload a new set of rules.

Configuring Enhanced Splitting

To configure enhanced splitting, you do the following:

1. Configure system brands. Each system brand is mapped to an output stream. See "Creating Brands" in *BRM Managing Customers*.
2. Use Pricing Center to configure the splitting rules. Each rule is associated with a system brand. If an EDR matches a rule, it uses the system brand defined in the rule.

   To create a valid mapping, the data in the EDR must match with all of the mapping data.

   For the data fields, you use regular expressions. For information on the regular expressions you can use, see "About Using Regular Expressions when Specifying the Data to Extract".
3. Configure an output stream for each system brand. See "Configuring EDR Output Processing".
4. Configure the FCT_EnhancedSplitting module. This defines the output stream for each system brand. See "FCT_EnhancedSplitting".

Sending EDRs to Pipeline Output Streams

The way you configure the pipeline to split EDRs into separate output streams depends on your system configuration:

- If your system does not include logical partitions, see "Sending EDRs to an Output Stream Based on Service Code".
- If you have an IMDB Cache-enabled system with multiple logical partitions, see "Sending EDRs to an Output Stream Based on Logical Partition and Service Code".

Sending EDRs to an Output Stream Based on Service Code

When you load rated events in the BRM database, RE Loader loads events for separate services from separate directories. Therefore, your pipeline needs to send EDRs to separate output streams for each internal service code. To do so, you use the IRL_EventTypeSplitting iScript.

To specify the output stream, you edit the **IRL_EventTypeSplitting.data** file. This file maps service codes to output streams. For example, this entry maps the TEL service code to the TelOutput stream:

```
TEL;TelOutput
```

> **Note:** You can also use the FCT_EnhancedSplitting module to send EDRs to different output streams based on EDR content. See "Using Rules to Send EDRs to Different Output Streams".

The IRL_EventTypeSplitting iScript must run after the FCT_ServiceCodeMap module and before FCT_Reject module.

To configure the IRL_EventTypeSplitting iScript, you configure it as part of the FCT_IRules module. See "FCT_IRules".

For more information, see "IRL_EventTypeSplitting".

Sending EDRs to an Output Stream Based on Logical Partition and Service Code

In an IMDB Cache system with multiple logical partitions, RE Loader must load pre-rated and re-rated events into the correct logical partition. However, RE Loader does not process logical partition information. To ensure that events are loaded into the correct logical partition, you must configure your pipelines to send EDRs to separate output streams based on the logical partition and service type.

For example, if your IMDB Cache system contains two logical partitions that each support the GSM and GPRS services, the pipeline must split EDRs into four output streams:

- GSM services from logical partition 0 to GSM output stream 1
- GSM services from logical partition 1 to GSM output stream 2
- GPRS services from logical partition 0 to GPRS output stream 1
- GPRS services from logical partition 1 to GPRS output stream 2

To do so, you must configure the FCT_AccountLPRouter module and IRL_EventTypeSplitting_tt iScript.

To specify the output stream, you must:

- Set the FCT_AccountLPRouter **Active** registry entry to **True**.
- Edit the **IRL_EventTypeSplitting_tt.data** file. This file maps logical partitions and service codes to output streams.

  For example, the following entries map the TEL service code from logical partition 0 to the TELOutput1 stream, and the TEL service code from logical partition 1 to the TELOutput2 stream:

```
TEL;0;TELOutput1
TEL;1;TELOutput2
```

> **Note:** You can also use the FCT_EnhancedSplitting module to send EDRs to different output streams based on EDR content. See "Using Rules to Send EDRs to Different Output Streams".

The IRL_EventTypeSplitting_tt iScript must run after the FCT_ServiceCodeMap module and before the FCT_Reject module.

To configure the IRL_EventTypeSplitting_tt iScript, you configure it as part of the FCT_IRules module. See "FCT_IRules".

For more information, see "IRL_EventTypeSplitting_tt".

Using Pipeline Manager with Multiple Database Schemas

In a multischema environment, you start a separate instance of Pipeline Manager for each BRM database schema.

- Use the FCT_AccountRouter module to find the account and send the EDR to the correct instance of Pipeline Manager. The FCT_AccountRouter runs in its own instance of Pipeline Manager.
- Use the DAT_AccountBatch module to supply data to the FCT_AccountRouter module. It runs in the same instance of Pipeline Manager as the FCT_AccountRouter module. See "DAT_AccountBatch".

To find the A number customer and the B number customer, the FCT_AccountRouter module does the following:

1. The module looks for the following data in the EDR:
   - The internal service code that indicates which data can be used to identify the account. For example, if the internal service code is a telephony service, the identifying data is the A number. A different service might use the IMSI as the identifier.

     You identify which data to use by using the Pricing Center. See "Specifying Which Data is Used for Identifying Accounts" in *BRM Setting Up Pricing and Rating*.
   - The timestamp for the EDR. The timestamp is important because telephone numbers can be used by different accounts at different times.
2. The module uses the DAT_AccountBatch module to look up the account. See "DAT_AccountBatch".

   Note:

   - If no A customer is found, the EDR is rejected. If the B customer is missing, no error is generated.
   - Because phone numbers can be recycled, the search is made on data from BRM audit objects.
3. The DAT_AccountBatch module returns the database number.
4. The FCT_AccountRouter sends the EDR to the correct pipeline, using the configuration defined in the registry.

Setting Up Account Identification in Multischema Systems

To set up account identification in a multischema system, do the following:

1. Configure service and account data in the Pipeline Manager database:
   - Configure internal service codes. The DAT_AccountBatch module retrieves account information based on which services are rated. See "About Mapping Services" in *BRM Setting Up Pricing and Rating*.
   - Configure how the FCT_AccountRouter module looks up accounts (for example, by telephone number or IMSI). You specify which data is used for identifying accounts when you configure the FCT_Account module. See "Specifying Which Data is Used for Identifying Accounts" in *BRM Setting Up Pricing and Rating*.

- (Optional) Configure account ID prefixes to use for handling duplicate telephone numbers. You configure this when you configure the FCT_Account module. See "Configuring Account ID Prefixes" in *BRM Setting Up Pricing and Rating*.

2. Configure the FCT_AccountRouter. See "FCT_AccountRouter".

   Note: Since the FCT_AccountRouter module needs the internal service code, the FCT_AccountRouter module must be placed after the FCT_ServiceCodeMap module in the pipeline.

3. Configure the DAT_AccountBatch module **UseAsRouter** registry entry.

   If set to **True,** the module is used by the FCT_AccountRouter module to route EDRs to separate Pipeline Manager instances. See "FCT_AccountRouter".

   If set to **False** (the default), the module is used by the FCT_Account module to get account data.

   See "DAT_AccountBatch".

6

Setting Up EDR Enrichment

This chapter describes ways to enrich event data record (EDR) data for rating by the Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager.

For information about pipeline rating, see "About Pipeline Rating".

Identifying the Network Operator/Service Provider

If your network supports multiple network operator/service providers (NO/SPs), you can use the FCT_NOSP module to identify the various NO/SPs. This module uses the source and destination codes and A number prefix in the EDR to assign a new source and destination code for the NO/SP.

Use this module when you need to identify the NO/SP and the NO/SP information is not available in the EDR; for example, when mobile networks are separated by means of the A number and you need to segment calls.

To map NO/SP data, you do the following:

1. Use Pricing Center to create NO/SP maps. See "Creating an NO/SP Map".

   You can also use a file to configure the NO/SP map.

2. Configure the FCT_NOSP module. See "FCT_NOSP".

3. Configure the DAT_NOSP module. See "DAT_NOSP".

   If you store NO/SP data in a file, see "Creating an NO/SP Data File".

Creating an NO/SP Map

To create NO/SP mappings, you use Pricing Center or a text file to set up the data that specifies how to map NO/SPs.

The mappings are based on the following data:

- Map group.
- The order in which mappings are applied. The first NO/SP map that matches is used.
- The source network and network destination recorded in the CDR.
- The phone number prefix to match.
- The new network source and destination to use in the EDR.

For information on the regular expressions you can use, see "About Using Regular Expressions when Specifying the Data to Extract".

Pricing Center stores the mappings in the IFW_NOSP database table.

For information on using a file to specify NO/SP maps, see "Creating an NO/SP Data File".

Creating an NO/SP Data File

You can store data for the DAT_NOSP module in the Pipeline Manager database or in a text file.

The configuration file must be an ASCII file. Each row defines one mapping rule. All fields in a row are separated by a semicolon (;). The fields in one row have the following order:

1. MAPGROUP (NOT NULL)
2. RANK (NOT NULL)
3. OLD_SOURCE
4. OLD_DESTINATION
5. A_PREFIX
6. NEW_SOURCE (NOT NULL)
7. NEW_DESTINATION (NOT NULL)

Setting Up Social Numbers

In some cases, customers want a B number to not appear on an invoice. For example, some countries require that certain called numbers remain anonymous, such as the number for a treatment center. You can set up social numbers to hide specific B numbers.

The social flag functionality is executed by the FCT_SocialNo module. When it finds a social number, the module sets a flag in the EDR B_MODIFICATION_INDICATOR field. You can use this flag to customize how to handle the number; for example, remove the last three digits or not allow the EDR to be included in an invoice.

To set up social numbers:

1. Use Pricing Center to specify social numbers.

   To specify social number, you specify the number that identifies it internally and give the number a descriptive name.

   The FCT_SocialNo module looks for an exact match of this number in the EDR.

   You can also use a file to specify social numbers. See "Creating a Social Number Data File".

2. Configure the FCT_SocialNo module. See "FCT_SocialNo".

Creating a Social Number Data File

You can specify social numbers in Pricing Center or in a file.

The social number data file uses the following syntax:

```
number1
number2
...
```

The number uses the same format as the normalized B number in the EDR.

International-access-code c\Country-code National-destination-code Access-code

For example:

```
0014085555555
```

Creating Call Destination Descriptions

You can set up descriptions for call destination area codes. For example, a prefix of 001408 can be described as "San Jose, California." The description is displayed on the customer's bill.

Prefix/description mapping is performed by the FCT_PrefixDesc module. You set up a mapping between prefixes and descriptions. The module finds the best match for the prefix and adds the description to the EDR DESCRIPTION field.

To set up prefix/description mapping:

1. Map prefixes and descriptions in Pricing Center.

   Note: You can also use a text file. See "Creating a Prefix/Description Data File".

2. Configure the FCT_PrefixDesc module. See "FCT_PrefixDesc".
3. Configure the DAT_PrefixDesc module. See "DAT_PrefixDesc".

Setting Up Prefix/Description Mapping in Pricing Center

To set up prefix/description mapping in Pricing Center, enter the following:

- The area code prefix. The prefix must have the same format that is used for normalization within a pipeline. The FCT_PrefixDesc module does not normalize numbers.
- The prefix type:
  - National
  - International
  - Special

  Note: If you use a file to store the mappings, you do not enter any prefix type. See "Creating a Prefix/Description Data File".

- The description to use, for example, "New York City."

  Important: All prefix/description pairs must be unique.

Creating a Prefix/Description Data File

You can define prefix/description mappings in the Pipeline Manager database, or in a file.

The mapping file has the following structure:

```
Prefix1; Description1
Prefix2; Description2
```

Mapping Multiple Phone Numbers to a Single Number

Customers with more than one telephone number sometimes want to get only one bill for all their numbers. You can map multiple telephone numbers to one number. This mapping is performed by the FCT_CliMapping module. The module uses the A number to search for a mapping entry. If there is a mapping entry, the A number is replaced by the Caller Line Identification (CLI) number

For example, a customer with a primary number 14085722000 could have five extra telephone numbers but would like to be billed as if they originated from the same number. To accomplish this for all the calls originating from the other five numbers, the A number is mapped to 14085722000 in the EDR.

To configure CLI mapping:

1. Define the mapping in a file. See "Creating a CLI Mapping File".
2. Configure the FCT_CliMapping module. See "FCT_CliMapping".

Creating a CLI Mapping File

As shown in Table 6–1, the FCT_CliMapping module requires an ASCII mapping file that contains the numbers to map to a single number.

- Every new line defines a mapping.
- Fields are separated by semicolons (;).
- There should be no semicolon at the end of a line.

Table 6–1 FCT_CliMapping Module File Structure

Column	Name	Position	Length	Format	Description
1	CLI_FROM	1	25	X(25)	Start CLI
2	SERVICE_CODE	27	5	X(5)	Service Code
3	MAPPING_CLI	33	25	X(25)	Mapping CLI
4	CUST_NUMBER	59	10	X(10)	Customer number
5	SUBSC_NUMBER	70	20	X(20)	Subscriber number
6	ISDN_RANK	91	1	X(1)	Rank of the ISDN number

Managing Number Portability

Customers may want to change network operators but retain their existing phone number. You can maintain number portability data to keep a track of the network operator a customer uses.

For maintaining number portability data, the DAT_NumberPortability module loads the data from the number portability file into the memory (pipeline's run-time memory).

The FCT_NumberPortability gets the number portability data from the DAT_NumberPortability module. The data includes the CLI, the new network operator ID, and the date that the customer changed network operators. The FCT_NumberPortability module uses the date of the event to determine which network

operator applies. If the new network operator applies, FCT_NumberPortability module updates the new network operator ID in the EDR.

BRM's number portability feature support batch pipeline (offline mode). See "Number Portability for the Batch Pipeline".

For batch pipeline rating, the FCT_NumberPortability module updates the EDR with the appropriate network operator ID based on the time stamp when the network operator changed.

For real-time pipeline rating, the FCT_NumberPortability module updates the EDR with the appropriate network operator ID based on the time stamp when the network operator changed. The FCT_Opcode module creates an opcode input flist containing the new network operator information. The flist is used for rating the calls in real time.

Number Portability for the Batch Pipeline

BRM uses semaphores to load the number portability data file using the batch process in the pipeline.

The number portability process for the batch pipeline makes the system inactive when the new number portability records are updated in the system.

About Number Portability Files

The number portability file is an ASCII file that stores the number portability data. The data from these files is loaded into the DAT_NumberPortability module.

You enter the following information in the number portability file:

- CLIs.
- The time stamp when CLIs are ported to a new network operator.
- The new network operator ID.

See "Creating a Number Portability Data File".

The following example shows a sample of number portability records in a number portability file:

```
Call Line ID            Date              Network Operator ID
408555                  20080101000000    D030
408555                  20080110000000    D010
```

The records in the example show that the subscriber changes the network operator to D030 on January 01, 2008. From January 01 to January 1, the subscriber is with the network operator D030. On January 10, the subscriber changes the network operator to D010.

BRM uses the following number portability files:

- Primary number portability file: Contains the primary number portability records that already exist.
- Delta number portability file: Contains the additional number portability records that need to be added to the memory and the primary number portability file.

You use the primary number portability file when you use the **reload** probe or the **Reload** semaphore to reload number portability data.

You use the delta number portability file when you use the **deltaLoad** probe to update the primary number portability file.

To manage number portability data, you can do the following:

- Purge data from the primary number portability file. To purge data from the file, use the purge utility (**purge.np_data.pl**). See "Purging and Reloading the Memory Records".
- Reload the data from the primary number portability file to the memory. See "Purging and Reloading the Memory Records".
- Append the data from the delta number portability file to the memory and subsequently to the primary number portability file. See "Appending Additional Number Portability Records".

Creating a Number Portability Data File

The DAT_NumberPortability module reads the required data from a reverse-sorted ASCII file.

Each row of the number portability file contains a CLI number, a date and time (when the numbers are ported), and a network operator ID as shown in Table 6–2.

Table 6–2 Input File for DAT NumberPortability module

Column	Format	Description
Call Line Identity Number	String	Specifies the telephone number or a part of the telephone number.
Portation date and time	*YYYYMMDDhhmmss*	Specifies the date and time of the number portation.
Network Operator ID	D*xxx*	Specifies the new telephone carrier.

An example of a sample file:

```
089761   20020910101230   D030
089761   20020912101230   D018
089545   20020820084000   D017
089545   20020920230010   D030
```

Important: The columns are separated by spaces.

Use the DAT_NumberPortability FileName registry entry to specify the file name for the number portability data file.

Purging and Reloading the Memory Records

The purge utility is used to delete existing records (from the number portability file) that are older than a time stamp specified in the utility. You can purge the data from the memory by using the purge utility. For purging the records from the memory, you must first purge records from the number portability file and then use the **reload** probe to ensure that the records in the number portability file and the memory are in sync.

To purge and reload number portability records:

1. Run the **purge_np_data.pl** utility to purge the primary number portability data file:

   purge_np_data.pl *NP_FileName TimeStamp* [**-b** *backup_filename*] [**-n**]

where,

- *NP_FileName* specifies the name of the primary number portability file. For example, **Primary_NP_Data.data**.
- *TimeStamp* specifies the date prior to which all the number portability records are purged. For example, 20080105000000.
- **-b** *backup_filename* specifies the name of the backup file that will contain the unpurged number portability records. For example, **Primary_NP_Data.bak**.

This command takes a backup of the existing number portability records in the backup file and deletes all number portability records prior to the date (timestamp) specified from the primary number portability file.

2. Initiate the **reload** probe or the Reload semaphore to reload the purged number portability data into the memory.

   This ensures that the records in the number portability file and the memory are in sync.

   The syntax for the **reload** probe is as follows:

   ```
   snmpSet Host_name 1.3.6.1.4.1.3512.1.7.3.1.1.1.1.47.0 -pPort_No
   ```

   Value: **True**/**False**

   where,

   - *Host_name* specifies the name of the host where you run the SNMP server.
   - *Port_No* specifies the SNMP port number configured in the pipeline registry configuration.

   Note: Use the **Reload** semaphore only for the batch pipeline. For the sample entry of **Reload** semaphore, see "Sample Semaphore File Entry".

   If the reload operation fails, the memory data will contain all the unpurged data. The primary number portability data file is moved to a *Pipeline_Home***/npdata/error** directory. *Pipeline_Home* is the directory where you installed Pipeline Manager.

Appending Additional Number Portability Records

You can use the delta number portability file to append additional or newly ported records to the primary number portability file. The delta number portability file contains the additional records that are in the same format as the primary number portability file. By default, the delta number portability file is stored in the *Pipeline_Home* directory. You can also manually add the additional records in the primary number portability file but this process is cumbersome when you need to add many records.

To append additional number portability records:

1. Initiate the **deltaLoad** probe or the **AdditionalNumPortData** semaphore with the delta number portability file name as a value. The delta number portability file contains additional entries in the same format as the number portability file.

   The syntax for the **deltaLoad** probe is follows:

   ```
   snmpSet Host_name 1.3.6.1.4.1.3512.1.7.3.1.1.2.1.47.0 -pPort_No
   ```

Value: *file name*

where:

- *Host_name* specifies the name of the host where you run the SNMP server.
- *Port_No* is the SNMP port number configured in the pipeline registry configuration.
- *File_name* specifies the name of the delta number portability file.

The additional entries are first added to the memory and then the memory data is dumped into the primary number portability file so that the records in the memory and the primary number portability file are in sync.

Note: Use the **AdditionalNumPortData** semaphore only for the batch pipeline. For the sample entry of **AdditionalNumPortData** semaphore, see "Sample Semaphore File Entry".

2. Initiate a **printData** probe or a **PrintData** semaphore to print the records to a file.

   The syntax for the **printData** probe is as follows:

   ```
   snmpSet HOST_NAME 1.3.6.1.4.1.3512.1.7.3.1.1.3.1.47.0 -PPORT_NO
   ```

   Value: *File name* or **NULL** (Upper or Lowercase)

   where:

   - *Host_name* specifies the name of the host where you run the SNMP server.
   - *Port_No* specifies the SNMP port number configured in the pipeline registry configuration.

   All number portability data in the memory is copied to the file. By default, this file is created in the *Pipeline_Home* directory.

   Note: Use the **PrintData** semaphore only for the batch pipeline. For the sample entry of the **PrintData** semaphore, see "Sample Semaphore File Entry".

Setting Up Number Portability

You must set up number portability to maintain the number portability data to keep track of the network operator a customer is using. You can configure number portability for the following:

- Batch pipeline
- Real-time pipeline

Setting Up Number Portability for Batch Pipeline

To set up number portability for the batch pipeline:

1. Configure the FCT_NumberPortability module. See "FCT_NumberPortability".
2. Configure the DAT_NumberPortability module. See "DAT_NumberPortability".

   When you configure the DAT_NumberPortability module, you specify the following:

- The search method. See "Configuring Number Portability Search".
- Normalization. See "Configuring Normalization for Number Portability".
- The number portability data file. See "Creating a Number Portability Data File".

Setting Up Number Portability for Real-Time Pipeline

To set up number portability for the real-time pipeline:

1. Configure the FCT_NumberPortability module. See "FCT_NumberPortability".
2. Configure the DAT_NumberPortability module. See "DAT_NumberPortability".

   When you configure the DAT_NumberPortability module, you specify the following:

   - The search method. See "Configuring Number Portability Search".
   - Normalization. See "Configuring Normalization for Number Portability".
   - The number portability data file. See "Creating a Number Portability Data File".
3. Configure the ISC_PopulateOpcodeAndUtilBlock_Diameter iScript in the registry file and place it in the processing pipeline after the FCT_NumberPortability module. See "ISC_PopulateOpcodeandUtilBlock_Diameter".
4. Configure the ISC_MapNetworkOperatorInfo iScript and place it after the ISC_PopulateOpcodeAndUtilBlock_Diameter iScript in the registry file. See "ISC_MapNetworkOperatorInfo".
5. Configure the FCT_Opcode module and place it in the processing pipeline after the ISC_PopulateOpcodeAndUtilBlock_Diameter iScript. See "FCT_Opcode".

Configuring Number Portability Search

You can configure which of the following search methods the DAT_NumberPortability module uses to find a phone number's current network operator:

- **Best match** searches the number portability file for objects with the best combination of matching phone number prefix and recent port date. For example, a number portability file includes the following entries and a pipeline module receives a CDR with a date of November 2 and the phone number 4085551234:

  ```
  Call Line ID      Date              Network Operator ID
  408               20081001000000    D030
  408555            20080801000000    D029
  408555            20080708000000    D028
  ```

  The DAT_NumberPortability module returns network operator D029 because the entry contains the most recent port date with the most matching prefix numbers.
- **Exact match** searches for the first object that exactly matches a given phone number.
- **Any prefix match** searches for the first object with a matching prefix.

To configure the search method, use the DAT_NumberPortability **SearchMethod** registry entry. See "DAT_NumberPortability".

Configuring Normalization for Number Portability

Use the following DAT_NumberPortability registry entries to specify number normalization data:

- CountryCode
- NationalAccessCode
- InternationalAccessCode
- InternationalAccessCodeSign

For example, you can normalize this phone number:

04106760279

so that it becomes:

00494106760279

See "DAT_NumberPortability".

7

Setting Up Pipeline Aggregation

This chapter describes the Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager aggregation feature.

For procedural information, see "Creating Aggregation Scenarios".

For information about pipeline rating, see "About Pipeline Rating".

About Aggregation

You use the aggregation feature to compile data from event data records (EDRs). Aggregation is typically used for the following:

- To filter and summarize data for compiled statistics about service usage, customer activity, network activity, dealer commissions, and so forth. For example, you can compile the sum of charges per customer per month.
- To aggregate data and use for rating. For example, you can compile GPRS usage records and rate the aggregated amount.
- To output a back-out file used as input for rerating. See "About Rerating Pipeline-Rated Events" in *BRM Setting Up Pricing and Rating*.

About Setting Up Aggregation Pipelines

Because the results of aggregation are typically not used by other modules, and because the aggregation process uses more resources than rating, aggregation is typically performed by a separate pipeline. You can run an aggregation pipeline parallel to a rating pipeline, so that the same input files will be processed by both pipelines.

The aggregation pipeline processes EDRs in transactions, typically one transaction per input file. When the transaction is finished, the data is written from memory to a file.

After every transaction, the aggregated results are written to a file that contains the transaction ID. For every transaction and aggregation scenario a result and a control file is created. The control files are used by the Database (DB) Loader utility to load the results into the database. See "Creating Aggregation Scenarios".

You define the control file in the FCT_AggreGate registry entries. After the data is processed by the pipeline, you use DB Loader to load the data from the file to a database. This can be the same database used by Pipeline Manager or a different database. You can run the utility automatically or manually. See "Database Loader".

See "Creating Aggregation Scenarios".

See "FCT_AggreGate", "DAT_ScenarioReader".

Important: If you configure aggregation for rerating, ensure that the FCT_AggreGate module is working in back-out mode. See "Configuring rerating in Pipeline Manager" in *BRM Setting Up Pricing and Rating*.

About Aggregation Scenarios

You aggregate data by creating aggregation scenarios (see "Creating Aggregation Scenarios"). An aggregation scenario specifies which data to use and how to process the data. You use the following criteria when setting up aggregation scenarios:

- **Filter conditions**: Allows you to choose which EDRs to aggregate; for example, you can specify all calls for a single customer, service, time period, or rate plan.

  Note: Conditional filters for combined attributes are not supported. See "Defining Filter Conditions" in *BRM Collecting Revenue Assurance Data*.

- **Grouping rules:** Allows you to group calls according to various criteria; for example, you can aggregate sums based on levels of duration, as follows:
  - All calls less than one minute.
  - All calls from one to five minutes.
  - All calls over five minutes.
- **Aggregation functions:** You can do the following with the data:
  - Count the number of selected events.
  - Sum the values of selected events.
  - Generate the square sum from values of selected events.

You can use any EDR field for filtering, grouping, or aggregating. However, aggregation criteria are limited to the following:

- The data must be in the EDR.
- The data must be available for an overall aggregation, typically of a NUMBER type.

Note: An aggregation scenario is analogous to an SQL SELECT statement with a GROUP BY claus.

For example, select all EDRs where the time period is peak time, then group the calls by levels of duration (for example, 0-1 minute, 1-5 minutes), and aggregate the charges for each level of duration.

Creating Aggregation Scenarios

When you create an aggregation scenario, you define filter, grouping, and aggregation functions. When you run an aggregation pipeline, you specify which scenario to use in the FCT_AggreGate registry. See "Creating Aggregation Scenarios".

Defining Filter Criteria

Define the following when defining filter criteria for a scenario:

- The status of the filter criteria (active or inactive).
- The EDR field that the filter is applied to.
- The data type to use; for example, string, number, or date/time.
- The value to filter with.
- The function used for aggregating:
  - Count
  - Sum
  - Square sum
- The decimal precision for the aggregated result.

See "Defining Filter Conditions" in *BRM Collecting Revenue Assurance Data*.

Specifying Scenario Attributes

In addition to defining the filters, groups, and aggregation functions, you can use the registry to define the following attributes for each scenario:

- The EDR container description for the scenario.

  Note: You can only set up filters and groups for fields included in the EDR container description.

- A default table name where the aggregation results are stored.
- The maximum number of aggregations held in memory before writing the data to a file. Enter **0** to specify an infinite number.
- Default directories for the following:
  - Where to store the aggregation results file (temporary and finished data).
  - Database (DB) Loader control files.
- A default delimiter for the single values of grouping and aggregation fields.

About Creating Groups

You use grouping rules to group and order aggregation results. Grouping rules are applied to EDR container fields and associated with a rank. For example, aggregation results can be grouped first by the event originator, then by zones, and then by time periods, or by duration, area code, or geographic zone.

When you define groups, you define the database columns that contain the aggregated data. For example, if you group aggregation results by duration, a column for that value is created in the table in the results file.

A group includes the following attributes:

- The EDR field for which the values are grouped; for example, time period or duration.

- The column name where the aggregations for the group are written in the results file.
- The data type for the grouped data:
  - String
  - Number
  - Date/time
- If applicable, the output format for the data. This can be date/time values or string format; for example UPPER, LOWER or SUBSTR(x,y).

About Creating Classes for Groups

You create classes to group results for a type of data. For example, to group aggregation results by the duration of events, you can create the duration classes shown in Table 7–1:

Table 7–1 Duration Classes

Class	Value to group by
Duration Class A	0 to 60 seconds
Duration Class B	61 to 300 seconds
Duration Class C	301 to 600 seconds
Duration Class D	More than 600 seconds

When creating a class, you define the following categories:

- The name; for example, duration.
- The data type:
  - String
  - Number
  - Date/time
- Class items. For each class item you define the following:
  - The name; for example, duration.
  - The data type (string, number, or date/time)
  - The value; the following is an example duration class item:

    D=1-5 & HH=08-20

    You specify values by using regular expressions. Use the same expressions defined when "Defining Filter Criteria".

    Tip: For values that do not match any other class items, specify an "undefined" class item for every class.

About linked classes

You can create multiple linked classes to group results by more than one category. For example, if you create groups for time period and duration, you can group results as follows:

- Peak time: 0-1 minute
- Peak time: 1-5 minutes
- Peak time: over 5 minutes
- Off-peak time: 0-1 minute
- Off-peak time: 1-5 minutes
- Off-peak time: over 5 minutes

About Defining Class Dependencies

During aggregation, class groupings are processed in order as defined in a tree structure. To define this order, you assign classes to nodes in the tree structure. For example, to aggregate data by time and duration, data is aggregated first by the time class, and then by the duration class.

8

Migrating Pipeline Manager Data between Test and Production Systems

This chapter explains the Pipeline Manager data migration features of Oracle Communications Billing and Revenue Management (BRM) Pricing Center. It provides conceptual information about Pipeline Manager data migration and operational information about implementing data migration features in your business. For step-by-step instructions about using the data migration features in Pricing Center, see Pricing Center Help.

Before reading this chapter, you should be familiar with pipeline rating concepts and using Pricing Center to create pipeline rating data. See "About Pipeline Rating". You should also be familiar with your business's internal policies for creating, testing, and deploying pipeline rating data.

Note: This document covers data migration for pipeline pricing data only. Real-time pricing data is exported and imported using different procedures. See "About Price Lists" in *BRM Concepts*.

About Pipeline Manager Data Migration

You use the Pipeline Manager data migration feature to create, test, and modify pipeline rating data in a development environment, then deploy it to your production environment. This capability provides increased flexibility and security by isolating development and testing activities from production systems.

The data migration features of Pricing Center are optional. You enable them during the Pricing Center installation or by modifying the Pricing Center configuration file. See "Enabling Data Migration in Pricing Center".

This is the basic workflow of the Pipeline Manager data migration process:

1. **Set up identical development and production systems.**

   To ensure data integrity and testing validity, the pipeline rating data in the development system must be exactly the same as the data in the production system. The two systems diverge as you make changes to the development environment, but they converge again when you export these changes and import them into the production system.

   See "Setting Up Development and Production Environments".

2. **Create and modify rate plans and discount models on the development system.**

   Using standard Pricing Center procedures with some modifications, you create and modify pipeline pricing data such as rate plans, discount models, and price

models. You use change sets to organize your work. Change sets are groups of related changes that are managed and exported as a whole. Change sets ensure data integrity by locking objects when necessary.

See "Understanding Change Sets"and "Understanding Locks and Associations".

3. **Test your changes on the development system.**

   You should test your changes thoroughly to ensure that they work as you expect. The data migration features in Pricing Center guard against major errors such as referring to objects that don't exist, but you must test the business content of your changes to make sure that you achieve the results that you want.

   See "Testing Change Sets".

4. **Export change sets from the development system.**

   After testing, you can export a single change set or a group of change sets to a file. The file contains the information required to recreate the modified objects in the production database. If you export a group of change sets to a single file, you can specify the order in which the change sets are exported.

   See "Planning the Export Process".

5. **Import change set files into the production environment.**

   The import process reads the file created during export from the development system. All changes are validated to ensure data integrity. For example, the import process checks that all objects referred to by objects in the change set exist in the production database. If errors are found, the entire file is rejected and no changes are made.

   See "Planning the Import Process".

Understanding Change Sets

Change sets are the basis of the Pipeline Manager data migration features in Pricing Center. Change sets track the changes you make and make it possible to treat those changes as a whole. They also enforce rules about which objects can be added, changed, or deleted. See "Understanding Locks and Associations".

For example, suppose a change in policies at your business makes it necessary to modify several price models. You can make all these modifications as part of single change set that you export from the development system and import into the production system as a whole. Using a change set guarantees that the production system receives exactly the same changes that you made in the development system.

The content of a change set is determined by the changes you make while that change set is *active*. When data migration is enabled, you must you have an active change set before you make any changes to rating and pricing data. You can have only one active change set at a time and a change set can be active for only one user at a time. Every change you make is part of the active change set.

You can activate, inactivate, and reactivate change sets freely. For example, you can activate one change set to make a change to a screen, then switch to another change set to make a different change. When you exit from Pricing Center, the active change set is automatically inactivated to make it available to other users.

You can create as many change sets as you need. In some cases you may use only a single change set to include all the changes associated with a particular pricing change. In other cases, you may need to set up a number of different change sets for various parts of the project. See "Organizing Work into Change Sets".

You use the Change Set Manager, a screen in the Pricing Center application, to create and manage change sets. See "About the Change Set Manager".

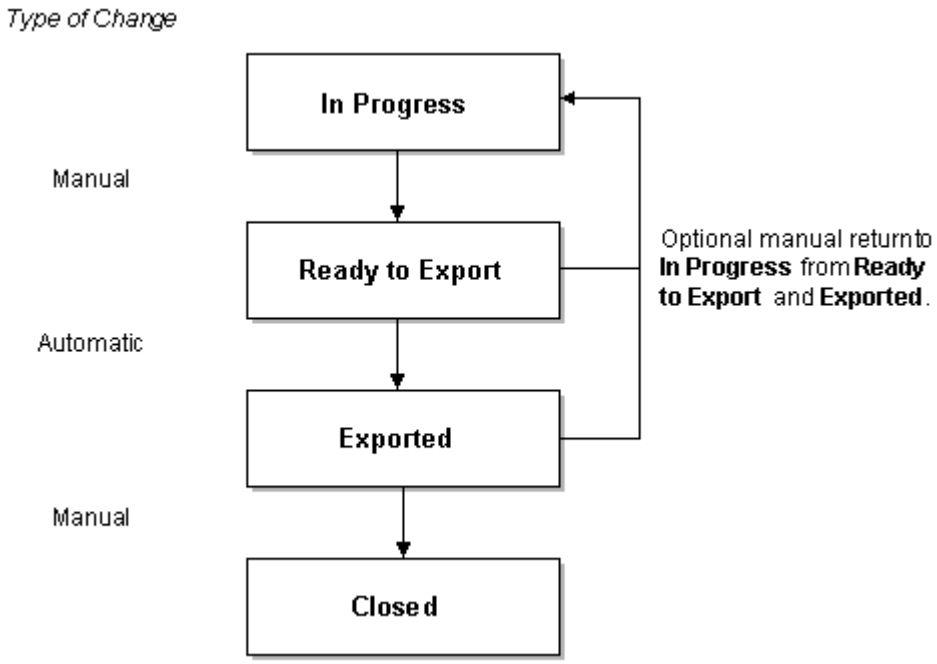
Understanding the Change Set Life Cycle

Change sets follow a life cycle that dictates what you can and can't do. This life cycle is comprised of change set states through which change sets move. By default, the life cycle includes four states. You can add additional states to conform to your business practices.

> **Important:** Managing the change set life cycle is a sensitive task. It is possible to invalidate testing scenarios and to create discrepancies between the test and production environments if you do not monitor and manage change set states carefully.

Figure 8–1 illustrates the default change set life cycle:

Figure 8–1 Default Change Set Life Cycle



There are four default change set states:

- **In Progress**. When you create a new change set, it is in the In Progress state, the working state for change sets. When an In Progress change set is active, you can make any modifications to the pipeline rating data that you need. From In Progress, you can manually change the state to Ready to Export or to a custom states that you define.
- **Ready to Export.** You change the state to Ready to Export when you confirm that all your data is complete and correct. You should test your data before switching to this state.

  In the Ready to Export state, you cannot make any changes to the data in the change set. If you need to make additional changes, you must manually switch back to In Progress.

The normal next step is to export the change set to a file. When you export a change set, its state automatically switches to Exported.

- **Exported**. This state shows that the change set has been exported, but has not yet been imported into the production database. You cannot make any changes to the data in a change set in the Exported state.

  You must manually change the state to Closed when the file is imported.

  Important: Changing to Closed state from Exported is a vital step. If you do not close the change set, all locks and associations remain active, potentially blocking the completion of other change sets.

  If there is a problem importing the change set, you can manually return the change set to the In Progress state to make necessary changes. This should be a carefully managed task to avoid confusion about which change set file contains the correct data. See "Planning the Export Process".

- **Closed**. This is the end state for change sets. The Closed state indicates that a change set is complete and in production. The change set is no longer available for use and cannot be reactivated. Its locks and associations are released. You can view information about the modifications made in this change set, but can no longer back them out.

For more information, see "Customizing Change Set States".

Understanding Locks and Associations

Locks and associations are used by change sets to ensure data integrity, prevent contradictory changes, and maintain information about what data has changed or been affected.

- A *lock* prevents a data object from being modified or deleted by a change set other than the one that established the lock.
- An *association* indicates that a data object is referred to by a locked object. While an object can have only a single lock, it can have multiple associations. It can also have a lock and associations at the same time. An object with an association cannot be deleted until the association is removed.

When data migration is enabled, locks and associations are automatically implemented by Pricing Center, preventing you from making changes that violate the locking rules. For example, if a change set has locked a price model, you can't modify that price model with any other change set.

Even though locks and associations are enforced automatically, you should understand the rules that are used to enforce them. This knowledge will help you and your colleagues work efficiently and smoothly. For example, if you create a new calendar in a particular change set, locking prevents that calendar from being visible to other change sets until the change set that created the calendar is closed. You need to plan your work accordingly. See "Organizing Work into Change Sets" for more information.

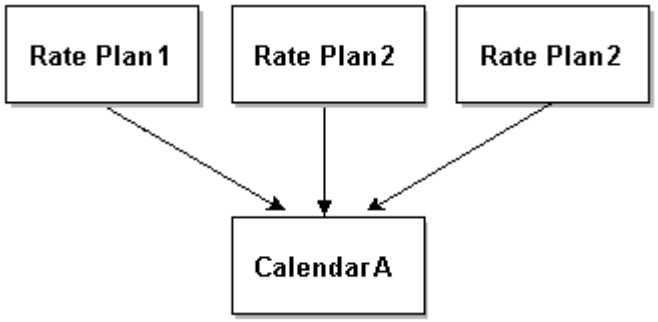
The next section, "Understanding the Pricing Data Model", provides background information about how pipeline rating data is stored in the database. "Locking and Association Rules" provides information about the rules governing locks and associations.

Understanding the Pricing Data Model

When you create a rate plan, a price model, or another element of pipeline rating data, it is stored in a table in the Pipeline Manager database, but for the purposes of clarity, we can think of Pipeline Manager data as independent objects.

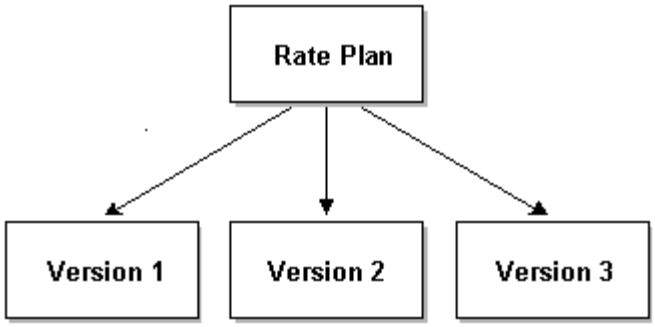
These objects have different kinds of relationships with each other. Many objects are *reusable*. They are elements such as calendars, time models, price models, and zone models that can be referred to by many other objects. For example, a single calendar can be used by many rate plans as shown in Figure 8–2:

Figure 8–2 Calendar A Reuse by Multiple Rate Plans



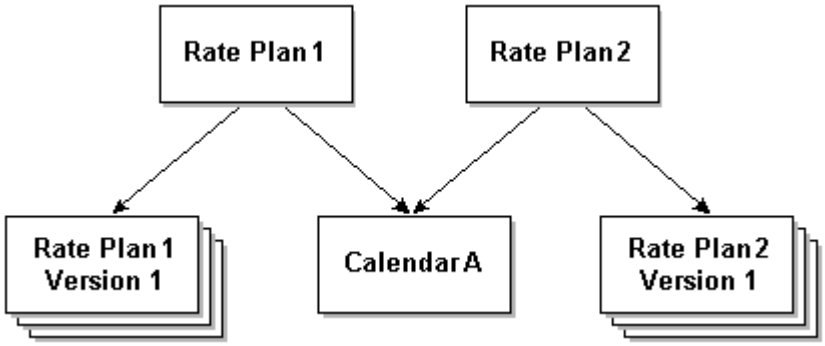
Other objects have a *parent-child* relationship. One parent object refers to many different occurrences of the same type of child object. For example, a single rate plan can contain an unlimited number of rate plan versions as shown in Figure 8–3:

Figure 8–3 Rate Plan Versions



The same object can have both parent-child relationships and references to reusable objects. For example, a rate plan can refer to its own rate plan versions as well as a calendar that is referred to by other rate plans as shown in Figure 8–4.

Figure 8–4 Parent-Child Relationships and Reusable Objects



Locking and Association Rules

When an object is modified, added, or deleted while a change set is active, that change set has a lock on the object. For example, if you make a change to a rate plan under Change Set 1, that rate plan is locked by Change Set 1 and cannot be modified or deleted by another change set until the lock is released.

These are the three most basic locking rules:

- An object can have only one lock. In other words, when an object is locked by one change set, it cannot be locked by another.
- An object that is locked by a change set cannot be modified or deleted by another change set.
- A newly created object is locked by the change set that created it.

These rules prevent change sets from making contradictory changes.

In addition to these basic locking rules, additional rules govern the objects related to locked objects. These rules work differently depending on whether the object is reusable or part of a parent-child relationship.

Rules for reusable objects

When a change set locks an object that directly refers to a reusable object, the change set creates an association to the reusable object. For example, when you modify a rate plan, the change set creates a lock on the rate plan and an association to the calendar referred to by the rate plan.

Associations are used to keep track of data that is potentially affected by changes made in a change set. They are also used to guard against deletion of objects that might cause data integrity problems.

These are the rules governing locking and associations for reusable objects:

- Objects with associations cannot be deleted until all associations have been removed. The deletion of associated objects is prohibited because it would cause data integrity problems; the locked object would have a reference to an object that no longer exists.
- Multiple change sets can create associations to the same object. For example, suppose Rate Plan A and Rate B both refer to Calendar Z. Change Set 1 modifies Rate Plan A, thereby creating an association to Calendar Z. Change Set 2 then modifies Rate Plan B, which creates an additional association to Calendar Z.
- A change set can create an association to a locked object, except if that object is newly added by another change set. For example, if Change Set 1 has modified

Price Model A and therefore locked it, Change Set 2 can still make a change that results in an association to that price model. However, if Change Set 1 has *added* Price Model B, the new price model is not visible to other change sets. No associations can be created to it by other change sets.

- A change set can obtain a lock on an associated object to make modifications. (The object cannot be deleted, however.) For example, suppose Change Set 1 modifies Rate Plan A, which locks the rate plan and creates an association to Calendar Z. Change Set 2 can still lock the calendar for modification. It cannot delete the calendar because that would violate the rule about deleting associated objects.
- Associations are created only for objects directly referred to by a particular locked object. When you lock a rate plan, you create an association to the calendar it refers to, but you do not create an association to any objects referred to by the calendar.

Rules for parent-child relationships

There is one main locking rule for parent-child relationships: a child object is locked when its immediate parent object is locked. For example, when you modify a rate plan, the rate plan and all its rate plan versions are locked.

Unlike associations, parent-child locking is recursive. In other words, not only the children of the parent object, but the children's children, are locked. For example, when you lock a rate plan, its versions are locked, which in turn causes the version's rate plan configurations, rate adjustments, and special day links also to be locked.

Because of the recursive nature of parent-child locks, you need to use some special techniques to minimize their impact. For example, to prevent all of a rate plan's children from being locked when you want to modify only a particular version, you can open the rate plan in read-only mode, then open the version you want to edit. Only that version and its children are locked, making it possible for other change sets to change other parts of the rate plan.

Important: You cannot *delete* a child object when you open its parent in read-only mode. For example, if you open a rate plan in read-only mode, then try to delete a rate plan version, you see an error dialog box. You must open and lock the parent object to delete a child object.

About the Change Set Manager

You use the Change Set Manager in Pricing Center to create and work with change sets. The Change Set Manager has two main areas:

- The navigation panel on the left enables you to open and create change sets. It lists all the available change sets in two sections: **Non-Exported** and **Exported**. You also search for closed change sets in the navigation panel.
- The Change Set Manager window displays the details of the open change set, including the name, state, description., the data modified by this change set, and the data associated with it. You can also activate a change set in the Change Set Manager window.

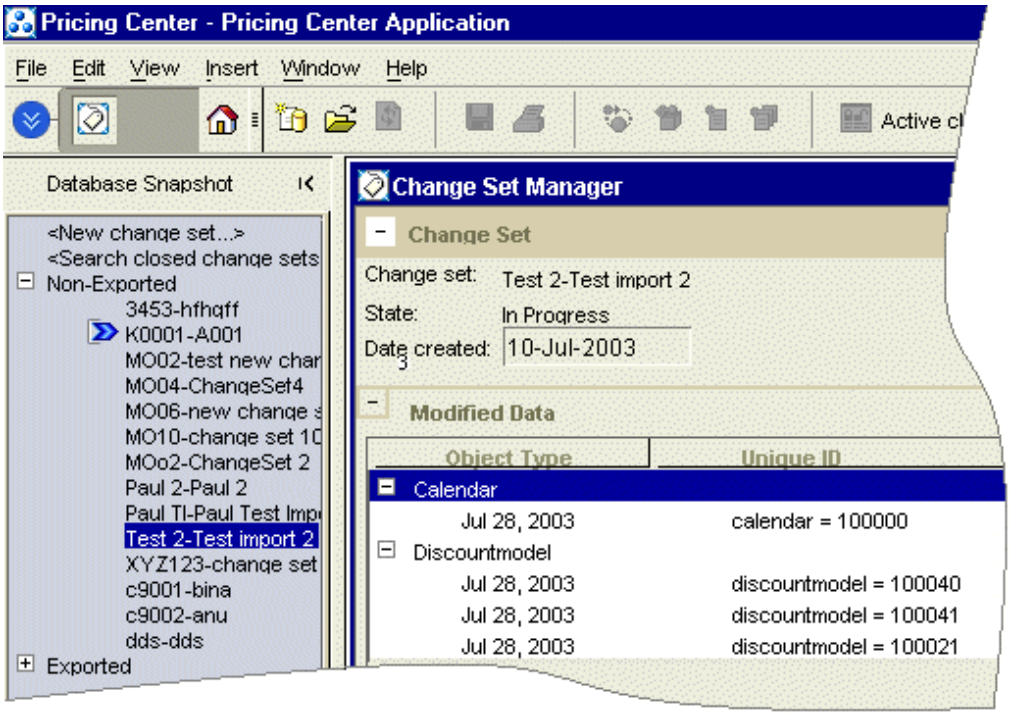
To open the Change Set Manager:

- Click the Change Set Manager button in the toolbar.

Alternatively, choose **View - Change Set Manager**.

The Change Set Manager opens. The navigation panel on the left side of the screen shows the available change sets as shown in Figure 8–5.

Figure 8–5 Change Set Manager



In the Change Set Manager, you can do the following:

- Create new change sets
- Activate change sets
- View change set data
- Work with change set states
- Back out change sets
- Export change sets
- Import change sets

Using Pipeline Manager Data Migration Features in Your Business

The Pipeline Manager data migration features in Pricing Center provide a framework that you can use to create and test new pricing data for your business. Because every business is different, you need to develop procedures that meet your needs. This section provides guidance about integrating Pricing Center data migration features into your business.

Important: The Pricing Center data migration features guard against problems in data integrity, but don't validate content. You must be careful and methodical to ensure that your business content is correct and testable.

Setting Up Development and Production Environments

To ensure accurate testing, your development and production data models must be identical to begin with. Each database must contain exactly the same objects with exactly the same contents.

When you create new pricing data on the development system, the two databases diverge, but due to the data that you have introduced. You can therefore be confident that your testing will reveal only issues introduced by the new data. Later, when all your changes have been exported from your development system to your production system, the data models will again be identical. See "Copying Production Data to the Development System".

You must also enable data migration for the two systems. When you install Pricing Center, you can choose not to enable data migration, to enable only import capability, or to enable the entire feature. You can also enable or disable data migration after Pricing Center has been installed. See "Enabling Data Migration in Pricing Center".

Until you enable data migration, you cannot import or export data to or from either system.

> **Important:** If you enable data migration for one instance of Pricing Center, you should also enable data migration for *all* other instances of Pricing Center that have access to the same test and production databases. Instances of Pricing Center without data migration enabled can make changes outside the scope of change sets, thereby causing inconsistencies in the data.

Another configuration step is setting up the change set life cycle that you want to use. The change set life cycle is based on your business process and reflects the stages that a change goes through from development to production. Depending on your business process, you may want to add change set states to the default life cycle. For example, you may want to add states called Testing and Approval. See "Understanding the Change Set Life Cycle" and "Customizing Change Set States".

Planning Your Work

You should plan your pricing data projects offline before using Pricing Center to implement them on the development system. You should know exactly which new pricing objects you need to introduce, which objects need to be changed, and which need to be deleted. You should also make sure to identify all reusable object that will be modified so that you can test all the objects that refer to them.

The exact planning process you use depends on your business needs and the complexity of the data model you are implementing. Whatever process you use, make sure there is a solid connection between your offline process and your work in Pricing Center. Use a consistent naming policy so that you can track a change from its inception in the planning process to its implementation in Pricing Center. For example, if your marketing department initiates a change through a formal change request, you can incorporate information from the change request into the Change Set ID, name, and description in Pricing Center.

Organizing Work into Change Sets

The way you organize change sets on your development system; how many change sets you use for a particular project and how changes are divided among them; can

have a large impact on how efficiently you complete your work. Here are two reasons why you need to think carefully about change set organization:

- Change sets can be dependent on each other. For example, if you are creating a new reusable object in one change set, other change sets cannot view or reference that object until the first change set is closed. Dependencies can also be based on content. If you make changes to the content of a price model in one change set, for example, any changes you make in other change sets that rely on this new content create a dependency.
- Locks created by one change set can prevent other change sets from accessing objects. For example, suppose two people are implementing two separate groups of changes, each in its own change set, and each change set requires the modification of the same price model. When one user modifies the price model, the other user is blocked until the first change set is closed.

You should use the information from the planning process to decide how many change sets to create and what to include in them. From your planning, you should develop a clear picture of the specific changes you need to make.

These are some of the factors to consider when organizing change sets:

- **The number and complexity of changes.** If you need to make only a few, simple changes, you can make them all in single change set. On the other, if you are working on a major overhaul of your pricing data, you should organize your work into multiple change sets.
- **The types of changes you are making.** Reusable objects can be referenced by many different objects, so changes to them can have a broad impact. To prevent one change set's locks from causing problems for other change sets, you can make all your reusable object changes in one change set that you export and import separately before other changes.
- **How many users are involved.** The larger the number of users involved in the implementation of a pricing data project, the more important it is to be very careful about planning and organizing the work. You can use change sets to divide work among users.

Testing Change Sets

Testing is vital to ensure that the pricing data you create is valid. Pricing Center prevents you from making errors that cause data integrity problems such as references to objects that no longer exist. But it is your responsibility to ensure that the *content* of your pricing data is valid: Pricing Center can ensure that a price model exists, but it can't verify that it contains the correct information for your business.

Pricing Center does provide warnings in situations where your actions might cause problems. For example, when you open a Pricing Center screen for an object that has an association to another change set, you can get information about which change set created the association.

You should test the data as realistically as possible by using the pricing data in your development system to rate CDRs in an environment that closely resembles your production environment.

These are some guidelines for helping you decide what to test:

- When you modify a reusable object, test to ensure that the objects that reference it are still valid. For example, if you make a change to a calendar, you should test all the rate plans that refer to that calendar to make sure that the change doesn't cause an unexpected result.

- Keep in mind the possible effects of multiple successive change sets modifying the same object. Locking prevents more than one change set from modifying an object at the same time. After a change set is closed and its locks are released, however, another change set can modify previously locked objects.
- Coordinate the activities of all users and all change sets to ensure that you are testing exactly what will be exported and imported. For example, another change set can modify a price model to which a rate plan in your change set has an association. Such a change doesn't cause a referential integrity problem; the price model still exists; but may cause unexpected results during rating. Ideally, when you are testing a change set or group of change sets, there is no other development activity that might affect the tests.

You should also make sure to test any real-time pricing data that is associated with your pipeline pricing data. For example, if you have introduced a new rate plan that is used to rate events associated with a new product, you should make sure that your testing includes both real-time and Pipeline Manager components. For information about testing real-time pricing data, see "Testing Your Price List" in *BRM Setting Up Pricing and Rating*.

Planning the Export Process

When a change set or group of change sets is complete, you export it to a file that can be imported into the target system.

Follow these guidelines for exporting change sets:

- Before you export, make sure that you understand any dependencies between change sets.

  Some dependencies are determined by locking rules. For example, if a new reusable object is introduced in one change set, that change set must be exported, imported, and closed before another change set can refer to the new object.

  Other dependencies are based on content. For example, if you use a change set to modify an existing calendar, you should make sure to export and import that change set before any change sets that rely on the modification.
- To ensure that change sets are exported and imported in the proper order, you can include multiple change sets in the same file and specify the sequence in which they are processed.
- Export change sets only when you are sure that they are complete and ready to be imported. There is no point in having incomplete export files in your system. They serve no purpose and there is some risk that they might be imported accidentally.

Managing Change Set Files

Change set files contain sensitive data, so you should manage them carefully. Three tasks are particularly vital:

- **Ensuring file security.** Once a file has been exported, it should be tracked carefully to ensure that it is not lost or corrupted. You should make sure there is no confusion about file names, which files are waiting to be imported, and similar matters.
- **Keeping track of file versions.** It's possible to have more than one version of a change set file. For example, if you export a file and then discover a problem with a change set in the file, you may need to make corrections and export the changes

again. You should be very careful to keep track of the file versions to ensure that you are importing the correct one.

- **Keeping track of the file sequence.** In some cases, files must be exported and imported in a particular order. For example, if you modify a calendar in one change set, you should export and import that change set before other change sets that rely on the modified calendar.

Depending on the complexity of your data model, a version control system may be useful for managing change set files.

You can specify the default locations to which change set files are exported and from which they are imported. For example, you can choose to export files to a location known to your version control system.

Planning the Import Process

Importing data into your production system is obviously a critical task. The Pricing Center import process checks every change to ensure that it is valid. If there are any errors during importation, the entire file is rejected, no changes are made to the target data, and the file is moved to an error directory.

In addition, you can take these steps to ensure that data is imported successfully:

- **Import files in the correct order.** If there are content dependencies among change set files, make sure to import them in the required order. For example, a change set may include a modification that another change relies on. The locking rules and other safeguards prevent data integrity errors such as references to non-existent objects, but you must keep track of dependencies based on business content.
- **Ensure that files reflect final data.** You should check that you are importing files that contain the final versions of your pricing data. If you accidentally import a file that is incomplete, you have to remove or modify the data manually; importing a file cannot be reversed. Careful planning and file management will prevent these problems.

Coordinating Real-Time Rating Data Migration and Pipeline Data Migration

BRM price plans contain a mixture of real-time and pipeline data. For example, when you define products, you can map some events to real-time rate plans and other events to pipeline rate plans.

Real-time and pipeline pricing data are exported and imported separately using different procedures. This document covers data migration for pipeline pricing data only. See "About Price Lists" in *BRM Concepts*.

You must manually coordinate the real-time and pipeline migration procedures. For example, if you added new pipeline pricing data associated with rating a new product, you must migrate both the new product (real-time data) and the new pricing data (pipeline data).

Configuring Pipeline Manager Data Migration Features

There are a number of configuration task for Pipeline Manager data migration that you accomplish outside the Pricing Center application, including enabling data migration, copying data to ensure that the development and production systems are identical, and optionally customizing the change set life cycle.

Enabling Data Migration in Pricing Center

The Pipeline Manager data migration features are optional. They are visible only if you enable them. You can enable the ability to import change set files without enabling the full set of data migration features. Import-only data migration is useful for production systems where you want to reduce the risk of accidental data changes.

You normally enable data migration during the Pricing Center installation process. You can also enable data migration after Pricing Center has been installed by making a change to the Pricing Center configuration file.

> **Important:** If you enable data migration for one instance of Pricing Center, you should also enable data migration for all other instances of Pricing Center that have access to the same test and production databases. Instances of Pricing Center without data migration enabled can make changes outside the scope of change sets, thereby causing inconsistencies in the data.

To enable data migration after installation:

1. Exit Pricing Center.
2. Open the Pricing Center configuration file (**custom.properties**) in a text editor. This file is located in the **\lib** subdirectory of the installation directory, normally **C:\Program Files\Portal\PricingCenter**.
3. Change the value for the **pipeline.datamigration** parameter to **2** (for full data migration functionality) or **1** (for import capability only).
4. Save the file.
5. Start Pricing Center.

Copying Production Data to the Development System

When you set up your development environment, you start out with an exact duplicate of the production database.

> **Important:** The test and production databases must be completely identical, including sequence IDs, for data migration to work reliably.

Use the replication tools provided with your database to copy the production database. See your system's documentation for instructions.

Customizing Change Set States

You can customize change set states to define a workflow for your projects. Your need for this customization depends on the complexity of your work. If there are only one or two change sets in progress at any one time and they contain simple changes, it may not be necessary to customize. On the other hand, if you have a team of planners working on multiple change sets in varying states of completion, you should customize the life cycle to reflect your process. See "Understanding the Change Set Life Cycle".

To customize change set states, you modify the **state.config** file, then load it by using the **stateconfigtool** utility. The **state.config** file lists each valid state transition. In other words, it defines all the states that it is valid to move to from any given state. It also

lists whether the transition is manual (accomplished by the user in the Change Set State dialog box) or automatic (accomplished by the BRM).

These are the contents of the default **state.config** file, which defines the standard change set life cycle:

```
# State Transition for Changeset
# currentState,nextState,Action

inprogress,readytoexport,manual
readytoexport,exported,automatic
readytoexport,inprogress,manual
exported,inprogress,manual
exported,closed,manual
```

Important: You can define additional states, but you cannot delete any default states. The custom states you define must come between In Progress and Ready to Export. All customized states must have a manual transition.

For example, the following file defines a new state called Testing. You can switch to Testing from In Progress and can switch from Testing to Ready to Export or back to In Progress. Note that you can't switch from Ready to Export back to Testing.

```
# State Transition for Changeset
# currentState,nextState,Action

inprogress,readytoexport,manual
inprogress,testing,manual
testing,readytoexport,manual
testing,inprogress,manual
readytoexport,exported,automatic
readytoexport,inprogress,manual
exported,inprogress,manual
exported,closed,manual
```

You load the change set configuration into the database by using the **stateconfigtool** utility. When you run this utility, you specify the file name, the database type, the host, the port number, the database instance, a login user name, and a login password.

To customize change set states:

1. Use a text editor to open the **state.config** file located in the Pricing Center directory, typically **Program Files\Portal Software\Pricing Center**.
2. Add new states, being careful to account for all the transitions necessary.

   Important: Don't make any changes to the default entries in the **state.config** file. Doing so will cause an error when you load the file.

3. Save the file under a new name. Saving the file under a different name preserves the original file in case you want to return to the default configuration.
4. From the *Pipeline_Home***tools\StateConfigTool** directory, run the **stateconfigtool** utility to load the file. *Pipeline_Home* is the directory where you installed Pipeline Manager. Use this syntax:

stateconfigtool -f *file_name* **-d** *database_type* **-h** *host* **-n** *port* **-u** *user_name* **-p** *password* **-i** *database_id*

For example:

```
stateconfigtool -f Pipeline_Home/tools/StateConfigTool/state.config -d oracle
-h sample_host -n 1521 -u sample_user -p sample_pwd -i pindb
```

Exporting and Importing Change Sets by Using the loadchangesets Utility

Under certain circumstances, importing and exporting change sets by using Pricing Center may be inconvenient or undesirable. For example, you may prefer not to allow Pricing Center access to your production system to prevent unauthorized or accidental changes to your production pricing data.

In these cases, you can use the **loadchangesets** command-line utility to export change sets from your test environment and import them into your production database.

> **Note:** Exporting and importing change sets by using the **loadchangesets** utility changes only the final stages of the entire pipeline pricing data migration process. You still use Pricing Center to create and manage change sets.

The general process for exporting and importing change sets is similar whether you use Pricing Center or **loadchangesets**. See Exporting change sets and Importing change sets in Pricing Center Help for more information.

The **loadchangesets** utility has two modes: interactive and non-interactive. They both enable you to import and export change sets, but work somewhat differently. See "Working in Interactive and Non-Interactive Mode".

Unlike Pricing Center, where you can choose which change sets to export, **loadchangesets** exports all the change sets that are in Ready to Export state. You should therefore be careful to monitor the life cycles of your change sets to ensure that you are exporting all the changes sets you want and none that you don't want. For more information about change set states, see "Understanding the Change Set Life Cycle" and the discussion about working with change set states in Pricing Center Help.

Specifying BRM Servers for the loadchangesets Utility

Before you can use the **loadchangesets** utility, you must specify the BRM servers to export from and import to. You enter the host names (or IP addresses) and port numbers in a configuration file.

To specify BRM servers for import and export:

1. Exit Pricing Center if necessary.
2. Open the Pricing Center configuration file (**custom.properties**) in a text editor. This file is located in the **\lib** subdirectory of the installation directory, normally **C:\Program Files\Portal\PricingCenter**.
3. To specify the server from which to export pricing data, enter the host name (or IP address) and port number in the **pipeline.datamigration.utility.export.environment.host** and **pipeline.datamigration.utility.export.environment.port** entries.

   For example, to export from TestPricingServer, port number 11960, enter the following:

```
pipeline.datamigration.utility.export.environment.host=TestPricingServer
pipeline.datamigration.utility.export.environment.port=11960
```

4. To specify the server to which to import pricing data, enter the host name (or IP address) and port number in the **pipeline.datamigration.utility.import.environment.host** and **pipeline.datamigration.utility.import.environment.port** entries.

   For example, to import to ProductionPricingServer, port number 56968, enter the following:

   ```
   pipeline.datamigration.utility.import.environment.host=ProductionPricingServer
   pipeline.datamigration.utility.import.environment.port=56968
   ```

5. Save the file.

Working in Interactive and Non-Interactive Mode

You can use the **loadchangesets** utility in interactive or non-interactive mode:

- In interactive mode, you issue a command for each step in the process of importing or exporting. After you enter interactive mode, the prompt changes to an angle bracket and commands are single words for performing particular actions. You can view a list of the change sets that will be imported and exported
- In non-interactive mode, you use commands that batch several related parts of the import or export process. You must enter a full command, including the utility name for each set of actions.

For more information about interactive and non-interactive modes, see "loadchangesets" in *BRM Setting Up Pricing and Rating*.

Exporting and Importing Change Sets in Interactive Mode

The following procedures describe exporting and importing change sets by using **loadchangesets** in interactive mode. See "loadchangesets" in *BRM Setting Up Pricing and Rating* for detailed information about syntax.

To export and import change sets by using **loadchangesets** in interactive mode:

1. Make sure the change sets that you want to export are complete and that they, and no others, are in Ready to Export state.
2. Go the **/lib** directory in the Pricing Center installation directory.
3. To switch to interactive mode, enter the following command:

   ```
   loadchangesets -i
   ```

   The prompt changes to **==>.**
4. To load the change set data from the export database into memory, enter the following command:

   ```
   export
   ```

5. To write the change set data from memory to a change set file, enter the following command. The file name must include the .**exp** file name extension.

   write *change_set_file*

The change sets are exported to the specified file in the **/export/done** subdirectory in the Pricing Center installation directory. This directory is created automatically if it doesn't exist when you run the utility.

6. Move the change set file that you created into the **/import** subdirectory in the PricingCenter install directory.
7. To read the change set data from the change set file into memory, enter the following command. The file name must include the **.exp** extension.

   read *change_set_file*

8. (Optional) Enter the following command to view the names of the change sets you loaded into memory:

   list

9. To load the change set data from memory into the import database, enter the following command:

   import

Exporting and Importing Change Sets in Non-Interactive Mode

The following procedures describe exporting and importing change sets by using **loadchangesets** in non-interactive mode. See "loadchangesets" in *BRM Setting Up Pricing and Rating* for detailed information about syntax.

To export change sets by using **loadchangesets** in non-interactive mode:

1. Make sure the change sets that you want to export are complete and that they, and no others, are in Ready to Export state.
2. Go the **/lib** directory in the Pricing Center installation directory.
3. Enter the following command, where *change_set_file* is the name of the file into which you want to export the change sets. The file name must include the **.exp** file name extension.

   loadchangesets -fx *change_set_file*

   The change sets are exported to the specified file in the **/export/done** subdirectory in the Pricing Center installation directory. This directory is created automatically if it doesn't exist when you run the utility.

To import change sets by using **loadchangesets** in non-interactive mode:

1. Move the change set file that you want to import into the **/import** subdirectory in the PricingCenter install directory.
2. Go the **/lib** directory in the Pricing Center installation directory.
3. Enter the following command, where *change_set_file* is the name of the file that you want to import. The file name must include the **.exp** file name extension.

   loadchangesets -fi *change_set_file*

9

Transferring Data Between Pipeline Manager Databases

This chapter describes how to transfer data from one Pipeline Manager database to another, such as from a test database to a production database, by using the Oracle Communications Billing and Revenue Management (BRM) **LoadIfwConfig** utility.

About Transferring Data

You transfer data by extracting data from a *source* Pipeline Manager database and then loading the data into a *destination* Pipeline Manager database. You specify which data to extract at the command line or by using an XML file. The **LoadIfwConfig** utility extracts the specified data from the source database to an output XML file. The utility can then load the output XML file directly into the destination database.

About Specifying the Data to Extract

You can specify to extract:

- All data in the Pipeline Manager database. You specify to extract all data by using only a utility command. For instructions, see "Extracting All Database Objects with LoadIfwConfig".
- All Pipeline Manager data that has been modified after a specified date and time. You specify to extract data based on the modification date and time by using utility commands. For instructions, see "Extracting All Database Objects Modified after a Specific Time".
- A subset of the data, based on the objects' attributes and modification time. You specify to extract a subset of the data by using an input XML file with the **LoadIfwConfig** utility. See "About Creating an Input XML File to Extract Data".

About Creating an Input XML File to Extract Data

If you are extracting a subset of data, you must create an input XML file that specifies the table from which to extract the data and, optionally, the criteria that the data must meet. The criteria consists of fields and their required values. For example, you can specify to extract from a specific table only those objects that have the SAMPLE field set to 100. The utility would then extract all objects with a SAMPLE field set to 100 as well as any child objects and any dependent objects. See "About Specifying to Extract Child and Dependent Objects".

When the input XML file specifies a table only, the utility extracts objects from the entire table as well as from any child and dependent objects. When the input XML file

specifies a table and required field values, the utility extracts from the table only those objects that contain the matching field values plus any child and dependent objects.

The syntax for the input XML file is shown below:

```
<RecordSet>
  <TableName [FieldName1 ="Value1"] [FieldName2 ="Value2"] .../>
</RecordSet>
```

where:

- *TableName* is the name of the table from which to extract objects.
- *FieldNameX* is the name of the table field that must match the specified value. You can list multiple field-value pairs.
- *ValueX* specifies the required field value. To be able to list multiple values for each field, see "About Using Regular Expressions when Specifying the Data to Extract".

For example, the following input XML file specifies to retrieve all objects from the IFW_RUM table that match all of the criteria below:

- Have their NAME field set to **Duration**.
- Have their RUM field set to **DUR**.
- Have their TYPE field set to **D**.

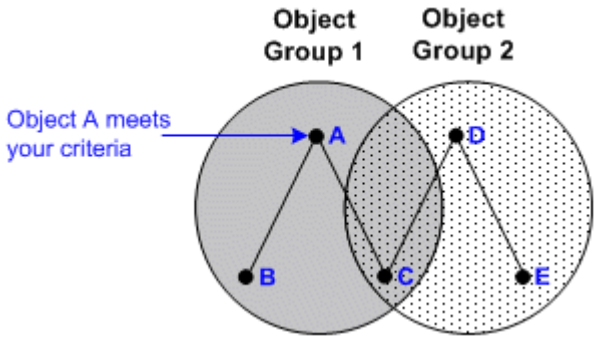
```
<RecordSet>
   <IFW_RUM NAME ="Duration" RUM ="DUR" TYPE ="D" />
</RecordSet>
```

About Specifying to Extract Child and Dependent Objects

When extracting a subset of data, the **LoadIfwConfig** utility automatically extracts all objects that meet your specified criteria as well as any related child objects. For example, when the utility extracts a rate plan object, the utility automatically extracts all child objects of the rate plan, such as the rate plan version and rate plan configuration objects.

If one of the child objects is also related to other objects, you can configure the utility to extract the other objects as well.

Figure 9–1 Object Relationships and Dependencies



For example, in Figure 9–1 above, object A meets your specified criteria. Because object A is a parent object, the utility automatically extracts object A and its two child objects, B and C. Because object C is also related to object D, extracting only the objects in object group 1 would break the relationships in object group 2.

To prevent this from happening, you can configure the utility to extract object D whenever object C is extracted. You do this by making object D a dependent of object C. In this case, whenever object C is extracted, the utility would also extract:

- Object D, because it is a dependent of object C.
- Object E, because it is a child of object D.

The utility determines the object dependencies by using the *Pipeline_Home***/tools/XmlLoader/LoadIfwConfig.xsd** file. *Pipeline_Home* is the directory where you installed Pipeline Manager. You customize the dependencies by using the *Pipeline_Home***/tools/XmlLoader/CustomConfig.xml** file.

Note: The settings in the **CustomConfig.xml** file override the settings in the **LoadIfwConfig.xsd** file.

The syntax for the **CustomConfig.xml** file is shown below:

```
<TableName AddDependingTable="AddTable" AddDependingTableMapping="Field1:Field2"/>
<TableName DependingTableNames="DependingTable"/>
```

where:

- *TableName* specifies the table that has dependent objects in another table.
- *AddTable* specifies the dependent table. The utility extracts data using the provided mapping whenever data from *TableName* is extracted.
- *Field1* is the field from *TableName* that is related to a field in *AddTable*.
- *Field2* is the relating field in *AddTable*. The utility extracts any objects from *AddTable* that have matching values whenever data from *TableName* is extracted.
- *DependingTable* specifies the list of dependent tables. Data from these tables must be extracted whenever data is extracted from *TableName*. You can list multiple table names by using the pipe symbol (|) as a delimiter.

Sample **CustomConfig.xml** entries are shown below:

```
<IFW_RATEPLAN_CNF AddDependingTable="IFW_TIMEMODEL"
AddDependingTableMapping="TIMEMODEL:TIMEMODEL"/>
<IFW_RATEPLAN_CNF DependingTableNames="IFW_TIMEMODEL|IFW_PRICEMODEL"/>
```

- The first line specifies that the utility must extract dependent data from IFW_TIMEMODEL, relating the TIMEMODEL field of IFW_RATEPLAN_CNF to the TIMEMODEL field of IFW_TIMEMODEL.
- The second line specifies to extract dependent data from the IFW_TIMEMODEL and IFW_PRICEMODEL tables whenever data is extracted from IFW_RATEPLAN_CNF.

About Using Regular Expressions when Specifying the Data to Extract

By default, you cannot use regular expressions when specifying the data to extract. This means that the input XML file must include a separate line for each required field value, which impacts performance because multiple entries generate multiple SQL queries to the database. For example, to retrieve objects from the IFW_RATEPLAN_CNF table that have an IMPACT_CATEGORY value of FRANCE or SPAIN, the input XML file would contain these lines:

```
<IFW_RATEPLAN_CNF IMPACT_CATEGORY="FRANCE"/>
```

```
<IFW_RATEPLAN_CNF IMPACT_CATEGORY="SPAIN"/>
```

You can configure the utility to accept the following regular expressions when searching defined fields:

- The asterisk (*) symbol for wildcard searches.
- The pipe (|) symbol for the logical OR operation.

You define which fields support regular expressions by configuring the **RegExFields** entry in the *Pipeline_Home*/**tools/XmlLoader/CustomConfig.xml** file.

The syntax for the **RegExFields** entry is shown below:

```
<TableName RegExFields="FieldName"/>
```

where:

- *TableName* is the name of the table that contains the specified field.
- *FieldName* is the name of the field that can be searched with regular expressions. You can list multiple fields by using the pipe symbol (|) as a delimiter.

For example, the following entry specifies that you can use regular expressions when searching for values in the IMPACT_CATEGORY field of the IFW_RATEPLAN_CNF table:

```
<IFW_RATEPLAN_CNF RegExFields="IMPACT_CATEGORY"/>
```

For the above example, you could retrieve records that have their IMPACT_CATEGORY field set to FRANCE or SPAIN by using this input XML entry:

```
<IFW_RATEPLAN_CNF IMPACT_CATEGORY="FRANCE|SPAIN"/>
```

About the LoadIfwConfig Error Messages

The **LoadIfwConfig** utility logs information about any errors it encountered to the log file you specified in the **ProcessLog** section of the **LoadIfwConfig.reg** registry file. Table 9–1 describes the utility's error messages.

> **Note:** The utility will pass through any error messages thrown by the Xerces SAX parser and the Oracle database. For information about these error messages, see the appropriate vendor's documentation.

Table 9–1 LoadIfwConfi Error Messages utility

Error message	Description
ERROR: Connection is not Valid	The utility failed to connect to the database.
ERROR: DataBaseStatus is not Valid	The database credentials or database connect string is not correct.
ERROR: Couldn't get next sequence: *SQLString*	The utility could not generate a sequence number to insert into the database.
ERROR: during insert: *SQLString*	The utility encountered an error during the insert operation.
ERROR: during update: *SQLString*	The utility encountered an error during the update operation.
ERROR: during delete: *SQLString*	The utility encountered an error during the delete operation.

Table 9–1 (Cont.) LoadIfwConfi Error Messages utility

Error message	Description
Exception occurred while executing *SQLString*	The utility encountered an error while running other SQL statements.
Exception from DB: *ErrorMessage*	The error message that was thrown by the Oracle database.
File Not parsed properly or Braces not matched properly	The utility's registry file (**LoadIfwConfig.reg**) contains incorrect entries or the entries are not framed correctly.
DependentFields structure provided in XML for depending table not proper	The **AddDependingTableMapping** entry in the **CustomConfig.xml** file is set incorrectly.
No rows for deletion because rows present in Child table	The utility could not delete the requested row because the row's associated child records still contain data.
Could not find valid Translation for: Table: *TableName* Referred Table: *TableName* Field: *FieldName* CODE Field Value: *FieldValue*	The required table dependencies are not provided in the input XML file.
FATAL ERROR at file: *FileName* line: *LineNumber* char: *Position* Message: *Message*	The input XML file contains mistakes, such as unparseable characters.

Using LoadIfwConfig to Transfer Data between Databases

To transfer data from one Pipeline Manager database to another, perform these steps:

1. Connect **LoadIfwConfig** to the source Pipeline Manager database. See "Connecting LoadIfwConfig to the Pipeline Manager Database".
2. (Optional) Specify the regular expressions and table dependencies that are supported on the source Pipeline Manager system. See "Customizing the Regular Expression and Dependent Table Settings".
3. Extract data from the source Pipeline Manager database. See "Extracting Data from a Pipeline Manager Database".
4. Connect **LoadIfwConfig** to the destination Pipeline Manager database. See "Connecting LoadIfwConfig to the Pipeline Manager Database".
5. Load data into the destination Pipeline Manager database. See "Loading Data into Pipeline Manager Databases".

Connecting LoadIfwConfig to the Pipeline Manager Database

You connect the **LoadIfwConfig** utility to the Pipeline Manager database by using the **LoadIfwConfig.reg** registry file. You can edit this registry file manually, but it is also updated by the **pin_setup** utility during the **LoadIfwConfig** installation process.

> **Note:** If you *upgraded* to Pipeline Manager 7.4, you must create the *Pipeline_Home***/tools/XmlLoader/log** directory before you start the **LoadIfwConfig** utility.

To connect **LoadIfwConfig** to the Pipeline Manager database:

1. Open the *Pipeline_Home***/tools/XmlLoader/LoadIfwConfig.reg registry file in a text editor.**

2. **Edit the registry entries to match your system environment. In particular, pay attention to these entries:**

   - In the **XMLCustomizationFile** entry, specify the location of the optional customization XML file. **LoadIfwConfig** contains a sample customization file (**CustomConfig.xml**) and a schema file (**CustomConfig.xsd**) to which the XML should conform.
   - In the **LoadDataFromDB** entry, specify whether to increase performance by loading the sequence-to-code translation information into memory. If this entry is missing or blank, it defaults to **False**.
   - (Optional) In the **RowFetchSize** entry, specify the number of database rows to retrieve on each trip to the database. Increasing the number of rows can reduce the required number of database fetches and increase the utility's performance. The default is **100**.

   The other entries are standard logging and connection registry entries. For information, see "Configuring Pipeline Manager" in *BRM System Administrator's Guide*.

3. Save and close the file.

A sample **LoadIfwConfig.reg** file is shown below:

```
LoadIfwConfig
{
  LogMessageTable
  {
    MessageFilePath   = ./
    MessageFileSuffix = .msg
  }

  ProcessLog
  {
    FilePath          = ./log
    FileName          = LoadIfwConfig
    FileSuffix        = .log
    WriteMessageKey   = True
  }

  DataPool
  {
    Database
    {
      ModuleName = DbInterface
      Module
      {
        # Common
        DatabaseName = DatabaseName
        UserName     = UserName
        PassWord     = EncryptedPassword
        AccessLib    = oci11g72
      }
    }
  }
  XMLCustomizationFile = CustomConfig.xml
  LoadDataFromDB = True
  RowFetchSize = 200
}
```

Customizing the Regular Expression and Dependent Table Settings

If you want the **LoadIfwConfig** utility to support regular expressions for any fields, or if you want to add table dependencies for any objects, you must modify the **CustomConfig.xml** file.

To customize the regular expression and dependent table settings, perform these steps on the source Pipeline Manager:

1. Open the *Pipeline_Home*/**tools/XmlLoader/CustomConfig.xml** file in a text editor.
2. Specify the table fields that support regular expressions by using the **RegExFields** entry. For example, the following entry specifies that the CODE and NAME fields in the IFW_RATEPLAN table support regular expressions:

   ```
   <IFW_RATEPLAN RegExFields="CODE|NAME"/>
   ```

   For more information, see "About Using Regular Expressions when Specifying the Data to Extract".
3. Specify any object dependencies by using the **AddDependingTable** and **AddDependingTableMapping** entries. For example, the following entry specifies to retrieve dependent data from IFW_TIMEMODEL, relating the TIMEMODEL field of IFW_RATEPLAN_CNF to the TIMEMODEL field of IFW_TIMEMODEL:

   ```
   <IFW_RATEPLAN_CNF AddDependingTable="IFW_TIMEMODEL"
   AddDependingTableMapping="TIMEMODEL:TIMEMODEL"/>
   ```

   For more information, see "About Specifying to Extract Child and Dependent Objects".
4. Specify any table dependencies by using the **DependingTableNames** entry. For example, the following entry specifies to extract dependent data from the IFW_TIMEMODEL table whenever data from IFW_RATEPLAN_CNF is extracted:

   ```
   <IFW_RATEPLAN_CNF DependingTableNames="IFW_TIMEMODEL"/>
   ```

   For more information, see "About Specifying to Extract Child and Dependent Objects".
5. Save and close the file.

Extracting Data from a Pipeline Manager Database

You can use the **LoadIfwConfig** utility to extract:

- All Pipeline Manager database objects. See "Extracting All Database Objects with LoadIfwConfig".
- All Pipeline Manager database objects modified after a certain date and time. See "Extracting All Database Objects Modified after a Specific Time".
- A subset of Pipeline Manager database objects. See "Extracting a Subset of Database Objects with LoadIfwConfig".

Extracting All Database Objects with LoadIfwConfig

The utility extracts all database objects by iterating through each table in the schema in order, selecting all of the rows, and dumping them directly into an XML file.

To extract all Pipeline Manager database objects:

1. Go to the *Pipeline_Home*/**tools/XmlLoader** directory.

2. Enter this command:

   Interactive mode

   ```
   LoadIfwConfig
   write [OutputFile]
   retrieve_all
   ```

   where *OutputFile* specifies the name and location of the file to which to extract the pipeline data. By default, the utility writes the output to a file named **default.out** in the current directory.

   Non-interactive mode

   ```
   LoadIfwConfig  -rall [-o OutputFile]
   ```

   where *OutputFile* specifies the name and location of the file to which to extract the pipeline data. By default, the utility writes the output to a file named **default.out** in the current directory.

   Note: For more information about the utility's syntax, see "LoadIfwConfig".

The utility writes the extracted objects to the output file in XML format. You can now load the output XML file directly into the destination Pipeline Manager database.

Extracting All Database Objects Modified after a Specific Time

To extract all Pipeline Manager database objects that have been modified after a specified date and time:

1. Go to the *Pipeline_Home*/**tools/XmlLoader** directory.
2. Enter this command:

   Interactive mode

   ```
   LoadIfwConfig
   write [OutputFile]
   retrieve_all [-t Modifidate]
   ```

   where:

   - *OutputFile* specifies the name and location of the file to which to extract the pipeline data. By default, the utility writes the output to a file named **default.out** in the current directory.
   - *Modifidate* specifies the timestamp after which to retrieve pricing objects. Enter the timestamp in the ISO-8601 format: *YYYY*-*MM*-*DD*T*hh*:*mm*:*ss* or *YYYY*-*MM*-*DD*, with the server time zone as the default. For example:

     ```
     1997-07-16T19:20:30
     ```

   Non-interactive mode

   ```
   LoadIfwConfig  -rall [-t Modifidate] [-o OutputFile]
   ```

   where:

   - *Modifidate* specifies the timestamp after which to retrieve pricing objects. Enter the timestamp in the ISO-8601 format: *YYYY*-*MM*-*DD*T*hh*:*mm*:*ss* or *YYYY*-*MM*-*DD*, with the server time zone as the default. For example:

```
1997-07-16T19:20:30
```

- *OutputFile* specifies the name and location of the file to which to extract the pipeline data. By default, the utility writes the output to a file named **default.out** in the current directory.

Note: For more information about the utility's syntax, see "LoadIfwConfig".

The utility writes the extracted objects to the output file in XML format. You can now load the output XML file directly into the destination Pipeline Manager database.

Extracting a Subset of Database Objects with LoadIfwConfig

To extract a subset of Pipeline Manager database objects:

1. Create an XML file that lists the objects to extract. The file specifies the table from which to extract the objects and, optionally, the criteria that the objects must meet. You can use the sample input XML file (*Pipeline_Home*/**tools/XmlLoader/samples.xml**) as a starting point.

   See "About Specifying the Data to Extract" for more information.

2. Go to the *Pipeline_Home*/**tools/XmlLoader** directory.

3. Enter the following command:

   Interactive mode

   ```
   LoadIfwConfig
   [-nodep]
   read InputFile
   fetch [-t Modifidate]
   write [OutputFile]
   ```

   where:

   – **-nodep** suppresses the object dependency relationships you configured in the *Pipeline_Home*/**tools/XmlLoader/CustomConfig.xml** file. The utility extracts only the objects specified in *InputFile* and ignores all dependent objects.

   Important: To suppress object dependency relationships in interactive mode, the utility session must start with the **-nodep** parameter.

   – *InputFile* specifies the name and location of the file that lists which objects to retrieve. This is the file that you created in step 1.

   – *Modifidate* specifies to retrieve objects that were modified after the specified timestamp. Enter the timestamp in the ISO-8601 format: *YYYY*-*MM*-*DD*T*hh*:*mm*:*ss* or *YYYY*-*MM*-*DD*, with the server time zone as the default. For example:

   ```
   1997-07-16T19:20:30
   ```

   – *OutputFile* specifies the output file to which the Pipeline Manager data is extracted. By default, the utility writes the output to a file named **default.out** in the current directory.

Non-interactive mode

```
LoadIfwConfig  -i InputFile  -r [-nodep] [-t Modifidate] [-o OutputFile]
```

where:

- *InputFile* specifies the name and location of the file that lists which objects to retrieve. This is the file that you created in step 1.
- **-nodep** suppresses the object dependency relationships you configured in the *Pipeline_Home*/**tools/XmlLoader/CustomConfig.xml** file. The utility extracts only the objects specified in *InputFile* and ignores all dependent objects.
- *Modifidate* specifies to retrieve objects that were modified after the specified timestamp. Enter the timestamp in the ISO-8601 format: *YYYY-MM-DDThh*:*mm*:*ss* or *YYYY-MM-DD*, with the server time zone as the default. For example,

  ```
  1997-07-16T19:20:30
  ```

- *OutputFile* specifies the output file to which the Pipeline Manager data is extracted. By default, the utility writes the output to a file named default.out in the current directory.

The utility writes the extracted objects to the output file in XML format. You can now load the output XML file directly into the destination Pipeline Manager database.

Loading Data into Pipeline Manager Databases

You load data into the destination Pipeline Manager database by using the **LoadIfwConfig** utility's update option or insert option.

- Update option: The utility verifies whether the data provided in the input XML file already exists in the database. If the data already exists, the utility updates the database record with the new information. If the data does not exist, the utility inserts the data into the database. To update data see, "Updating the Pipeline Manager Database".
- Insert option: The utility inserts data into the database without verifying whether the data already exists. To insert data, see "Inserting Data into the Pipeline Manager Database".

Updating the Pipeline Manager Database

To update the data in the Pipeline Manager database:

1. Go to the *Pipeline_Home*/**tools/XmlLoader** directory.
2. Enter this command:

   Interactive mode

   ```
   LoadIfwConfig
   read InputFile
   update
   commit
   ```

   where *InputFile* specifies the name and location of the XML file that contains the extracted pipeline data from the source database. This is the output file you generated in "Extracting Data from a Pipeline Manager Database".

   Non-interactive mode

```
LoadIfwConfig  -u  -c  -i InputFile
```

where *InputFile* specifies the name and location of the XML file that contains the extracted pipeline data from the source database. This is the output file you generated in "Extracting Data from a Pipeline Manager Database".

Note: For more information about the utility's syntax, see "LoadIfwConfig".

The utility loads the data from the XML file into the Pipeline Manager database and commits the data. If there is a failure, the utility rolls back the data and displays an error message.

Inserting Data into the Pipeline Manager Database

To insert data into a Pipeline Manager database:

1. Go to the *Pipeline_Home*/**tools/XmlLoader** directory.
2. Enter this command:

Interactive mode

```
LoadIfwConfig
read InputFile
insert
commit
```

where *InputFile* specifies the name and location of the XML file that contains the extracted data from the source database. This is the output file you generated in "Extracting Data from a Pipeline Manager Database".

Non-interactive mode

```
LoadIfwConfig  -I  -c  -i InputFile
```

where *InputFile* specifies the name and location of the XML file that contains the extracted data from the source database. This is the output file you generated in "Extracting Data from a Pipeline Manager Database".

Note: For more information about the utility's syntax, see "LoadIfwConfig".

The utility loads the data from the XML file into the Pipeline Manager database and commits the data. If there is a failure, the utility rolls back the data and displays an error message.

Deleting Data from a Pipeline Manager Database

Note: Make sure the utility is connected to the Pipeline Manager database. See "Connecting LoadIfwConfig to the Pipeline Manager Database".

To delete data from a Pipeline Manager database:

1. Create an XML file that specifies the data to delete. The file includes the table from which to delete the objects and, optionally, the criteria that the objects must meet. For information, see "About Specifying the Data to Extract".
2. Test the XML file by running the **LoadIfwConfig** utility with the **-r** or **fetch** parameter. Verify that the output file lists the correct objects to delete. See "Extracting a Subset of Database Objects with LoadIfwConfig" for more information.
3. Go to the *Pipeline_Home*/**tools/XmlLoader** directory.
4. Enter the following command:

   Interactive mode

   ```
   LoadIfwConfig
   read InputFile
   delete
   ```

   where:

   - *InputFile* specifies the name and location of the file that lists the objects to delete. This is the file that you created in step 1.

   Non-interactive mode

   ```
   LoadIfwConfig  -p[f] -i InputFile
   ```

   where:

   - **f** turns off the delete confirmation message.
   - *InputFile* specifies the name and location of the file that lists the objects to delete. This is the file that you created in step 1.

   Note: For more information about the utility's syntax, see "LoadIfwConfig".

The utility deletes the specified database objects.

Part II

Configuring Pipeline Discounting

Part II describes how to configure pipeline discounting in Oracle Communications Billing and Revenue Management (BRM). It includes these chapters:

10

About Discounts

This chapter provides an overview of Oracle Communications Billing and Revenue Management (BRM) discounting. You can discount events rated in real time and by pipeline batch rating.

Important: The Batch Discounts (batch discounting in a pipeline) feature is bundled with Advanced Discounting Manager, an optional feature that requires a separate license.

Before reading this chapter, you should be familiar with real-time and batch rating. See "About Real-Time Rate Plans" in *BRM Setting Up Pricing and Rating* and "About Pipeline Rating" for more information.

Note: In addition to the discounting procedures discussed in this chapter, you can create rate adjustments for real-time events in two ways:

- You can reduce purchase, recurring, and usage fees by a flat percentage. These rate adjustments are offered through deals. See "Providing Deal-Level Discounts" in *BRM Setting Up Pricing and Rating* for more information.
- You can use multiple rates and set minimum and maximum quantity values for them. You do this in real-time rate plans. See "About Quantity-Based Rating" in *BRM Setting Up Pricing and Rating* for more information.

For information about setting up specific types of discounts, see "About Implementing Discounts". For information about configuring discounting modules and real-time discounting pipelines, see "Configuring Discounting Modules and Components".

About Discounting

You use discounts to reduce the charges associated with billable events and to grant or consume non-monetary resources such as free minutes or frequent flier miles. You can discount usage charges, purchase fees, and recurring charges. You can discount events rated in real time and by pipeline batch rating.

Note: In real-time rating, non-currency resources can be consumed during rating (by the real-time rating engine), or they can be consumed by the real-time discounting pipeline. In batch pipeline rating, non-currency resources are consumed by your batch discounting module. As such, in batch rating, you use a discount to consume a non-currency resource. In addition, note that you cannot use the pipeline rating module to consume non-currency resources (the pipeline rating module does not have the necessary access to non-currency resource sub-balances).

You can set up offer profiles using the provisioning tags in discounts to manage policy-driven charging. For more information, see "Policy-Driven Charging" in *BRM Setting Up Pricing and Rating*.

Discounts are separate, purchasable items similar to products. You bundle discounts with products in deals that customers purchase.

For example, you can offer a deal called GSM Plus that includes basic GSM telephony with 300 free peak and 500 free off-peak minutes for a $60 setup fee, a $40 monthly fee, and usage charges. The deal already includes one discount, the free minutes, but you can add an additional promotional discount to the deal that reduces the charges even more:

- 50% off the monthly fee for the first 6 months
- Waiver of the setup fee (a 100% discount, in effect)
- 25% off for usage over 750 minutes

This deal now includes discounts on events rated in real time (the monthly and setup fees) and events rated in batch by Pipeline Manager (the free minutes and the 25% usage discount). The discounts cover usage charges, recurring charges (the monthly fee), and purchase charges (the setup fee).

When you provide free resources such as free minutes, discounts and products can work together:

- You use products to grant the free resources. When you define the product, you use cycle events to grant the free resources. For example, you can define the product to grant 100 free minutes each week or 500 minutes on a one-time basis.
- You use discounts to determine how free resources are consumed. Charges are applied by rating before events are discounted, so in the case of free minutes, the discount credits the amount charged for the free minutes used and reduces the balance of free minutes.

Not all discounts necessarily grant, consume, or discount resources. For example:

- When an account shares free minutes with other accounts, you can use one discount to record the balance of available minutes in the sharing account. A second discount uses the recorded balance to consume free minutes for the account that made the call.
- When you set up billing-time discounts, you can use a discount to update a counter balance; for example, a balance that tracks total usage. You use a second discount to apply a percentage off based on the total usage balance.

BRM supports subscription discounts and system discounts.

You use Pricing Center to set up discounts for both real-time and batch pipeline rating.

Discounting Process Overview

Discounts are processed after rating and are based on events. Discounting changes the amount that was charged during rating. Discounting takes place in a pipeline, even if the rated event originates in real-time rating.

The way events are passed to discounting depends on whether they were rated in real time or in batch. (See "Understanding the Discounting Architecture".) But the discounting process is similar for both real-time and batch events.

Discounting works on event data records (EDRs), each of which contains data about a single rated event. Every EDR includes one or more charge packets that result from the event being rated. (Real-time rating does not normally produce EDRs or charge packets, but real-time charges are converted to EDR format when passed to a discounting pipeline.)

There are four main phases to the discounting process:

1. **Discount analysis**. During discount analysis, every EDR is examined to determine whether there are any discounts that apply to this event type. For example, if the EDR contains a Global System for Mobile Communications (GSM) usage event, the discount analysis module checks for discounts that apply to GSM usage.
2. **Filtering by event attribute**. In this phase, discounting evaluates the charge packets in each EDR to determine whether they qualify for discounting. When you define a discount model, you specify event attributes that determine whether charge packets qualify for a particular discount. The discount filter can filter by date and time, zone model, impact category, service code, and other attributes. See "Filtering EDRs for Discounting" for information about setting up discount filters.
3. **Checking conditions**. If a charge packet passes through the discount filter successfully, discounting now checks whether the charge packet meets the conditions that trigger discounting. Conditions include factors such as charge, usage quantity, and number of events. You can also create conditions that include arithmetic operations and other expressions.

   Note:

   - Discount triggers are required. However, if you do not need to set conditions, you can configure the trigger to pass all charge packets.
   - If *either* the usage quantity or the charge amount is 0, discounting does not evaluate the charge packet and no discounts are applied.

   See "Determining if Usage Qualifies for Discounting" for more information about defining discount conditions.
4. **Calculating and applying the discount**. If all the required discount conditions are met, discounting calculates and applies the discount by using a discount rule. A discount rule defines the amount of usage to consider for discounting, how much of the usage to discount, the discount amount, and the balances to impact. See "Defining How Discounts Are Applied".

Discounts and Balances

When discounting processes an EDR, it requires access to balances related to the event and the discount. Balances keep track of resources, which can be monetary or

non-monetary. Non-monetary resources include usage quantities such as minutes or bytes, as well as special-purpose resources such as loyalty points.

For example:

- If you grant free minutes, discounting needs to know how many minutes have already been consumed so that it can determine how much of a call is covered. Discounting also requires the ability to update the free minutes balance to reflect the minutes consumed.
- If you give a 25% discount on usage charges over $100, discounting needs to know the current balance of usage charges in order to determine whether the discount applies and to calculate the amount of the discount.
- If you set up a billing-time discount that grants a discount based on the total amount of usage during a certain period, discounting needs to know the amount of usage during the validity period.

Some balances are primarily aimed at tracking balances that are granted and then used, such as free minutes. Other balances keep track of usage or consumption, such as the total quantity of data received or minutes used. This latter kind of balance is called an *aggregation counter* and can be used as the basis for granting a discount. For example, you can create a discount that grants 10 frequent flier miles for every 60 minutes of usage.

You specify which balances are used by a discount when you set up the discount balance impact. See "Defining How Discounts Are Applied". If you use an aggregation counter to calculate the discount, you include that balance in the expression you enter for the calculation. See "Using Expressions in Discount Models".

Some kinds of balances, such as those used for free minutes, can be affected by rollovers. For example, the current balance of free minutes could include minutes granted this month as well as minutes rolled over from previous months. You set up rules that govern how balances are rolled over when you set up products. For more information, see "About Rollovers" in *BRM Setting Up Pricing and Rating*.

The balances referenced by discounts are configured when you set up plans. For more information, see "About Tracking Resources in Account Sub-Balances" in *BRM Setting Up Pricing and Rating*.

BRM handles balances differently for real-time discounting and batch discounting. See "Balances and Real-Time Discounts" and "Balances and Pipeline Discounts".

In addition to balances that are maintained permanently, you can configure temporary event balances for use in discounts. See "Using Event Balances in Discounts".

Balances and Real-Time Discounts

For real-time rating and discounting, BRM stores balances in the BRM database. These balances are updated in real time as transactions occur. Because balances are stored in only one location, there is no need for synchronization. The discounting pipeline gets and updates these balances through the Connection Manager (CM). See "Real-Time Discounting Architecture".

Balances and Pipeline Discounts

For pipeline rating and discounting, BRM stores balances in data modules in the pipeline as well as in the BRM database. Because data is stored in two locations, balances must be synchronized:

- Rating and discounting in the pipeline result in balance impacts that must be reflected in the BRM database. For example, if a discount results in the reduction of a free minutes balance, that change must be reflected in the BRM database so that the balance can be displayed accurately in Customer Center. These updates are made to the BRM database by Rated Event (RE) Loader when it loads rated events.
- Activity in the BRM database affects discount balances used in pipeline rating. For example, a customer might purchase a discount that provides 500 free minutes or a customer service representative (CSR) could post a debit to a balance. These balance changes must be reflected in the discount balance handled by pipeline rating. When these balance changes occur, BRM uses the Account Synchronization Data Manager (DM) to send the discount balance impact to Pipeline Manager.

See "Pipeline Discounting Architecture" for more information about how discounting updates balances.

About Billing-Time Discounts

Important: The billing-time discounts feature is included in Advanced Discounting Manager, an optional feature that requires a separate license.

A *billing-time discount* is determined at the end of the billing cycle. This allows you to grant discounts based on resources used during a billing cycle. For example, you can create a billing-time discount to discount $10 if the total usage fee is more than $100 or to grant 10 free minutes if the total minutes used are more than 500. A billing-time discount can also be based on values other than usage; for example, the number of months a customer has subscribed to a service.

Billing-time discounts are calculated and balance impacts are applied when BRM billing is run. You can configure the discount to be applied in the current or next billing cycle. For example, an account can purchase a billing-time discount that grants 10% of the total telephony minutes used in the current cycle to the next billing cycle as free minutes. If, at the end of the current billing cycle, the account has a total of 100 minutes, when the **pin_bill_day** script is run to run billing, 10 free minutes are granted to the account for the next billing cycle.

Billing-time discounts require an aggregation counter that tracks the aggregated amount for which the discount is granted. For example, to grant 20% off all usage charges for the month, usage amounts are added to the aggregation counter. When billing is run, the discount is based on the amount in the aggregation counter.

To offer a billing-time discount, you set up the following elements:

- A non-currency resource for the aggregation counter to track the total charges or amount that is discounted. You include this resource in the price plan. BRM provides some aggregation counter resources for specific types of discounts, such as volume-based discounts.
- Two discounts:
  - A usage discount to update the aggregation counter. See "About Using Discounts to Aggregate Usage".
  - A billing-time discount to apply the discount when billing is run.

For more information, see "Creating a Real-Time Aggregation Discount".

Billing-time discounts can also be shared with accounts in a discount sharing group. The billing-time discount is based on the total resources used by the entire group. The discount can be applied to the group owner or it can be distributed among group members. See "About Snowball Discounts".

About Using Discounts to Aggregate Usage

Aggregation discounts are usage discounts that increment an account's aggregation counter by an amount that is equal to the usage. For example, when a subscriber makes a 10-minute call, the discount increments the aggregation counter by 10.

Aggregation discounts do not reduce fees or grant resources. Instead, they are used in conjunction with billing-time discounts, which are granted based on the aggregated balance; for example, 10% off when the number of minutes used for all calls exceeds 300.

The amount aggregated can be charges or units used, such as minutes, SMS messages, and bytes. The aggregation counter that is incremented is always a non-currency balance regardless of whether the usage aggregated is a charge or a unit.

Aggregation discounts can also be shared. You share aggregation discounts for two reasons:

- To aggregate the service usage fees for several accounts when a discount is granted to one account based on the total usage of all accounts. For example, a corporate account can receive 15% off its monthly service charge when the total usage for all its employees exceeds $900.
- To track the consumption of free resources (such as minutes or text messages) when those resources are shared among several accounts. You do this when you want to limit the amount of resource that an account can consume. For example, an account shares 600 free minutes with two user accounts whose usage should be limited to 100 minutes each. The user's service usage is aggregated in each user's account. When a user's aggregation counter reaches 100, the discount does not allow the user to consume additional free minutes.

For more information about sharing discounts, see "About Shared Discounts".

For information on setting up aggregation discounts, see "Setting Up Billing-Time Discounts".

About Cycle-Event Discounts

A cycle event is a recurring event typically used for charging a fee, such as a monthly subscription fee. You can apply a discount to one or more cycle events. For example, you can grant 50% off the subscription fee for the first six months of usage.

You can also grant discounts on cycle events that are not dependent on a billing or accounting cycle (known as flexible cycles). You select the type of cycle event to discount when creating the discount. For information about flexible cycles, see "About Flexible Cycles" in *BRM Configuring and Running Billing*.

If a cycle-event discount grants non-currency resources, such as free minutes or bonus points, those resources are applied to either the current or future accounting cycle, regardless of the type of cycle event you select. You specify the accounting cycle for which the granted resource is valid when you configure the discount balance impact.

For information about setting up cycle-event discounts, see "Setting Up Cycle-Event Discounts".

About Shared Discounts

Discounts can be shared among several accounts. For example, you might offer 10 free minutes for every 500 minutes of combined usage for all accounts or share a pool of free minutes with a group of accounts or services.

To share discounts, you create discount sharing groups. A discount sharing group consists of an owner account or service and one or more services from member accounts. The owner shares its discount with the members. For more information, see "About Discount Sharing Groups" in *BRM Managing Accounts Receivable*.

With discount sharing, you sometimes need several discounts to specify how resources are granted or consumed for each account. For example, to offer 20% off to member accounts when the total usage for all accounts exceeds 1,000 minutes, you set up a discount to record the usage for each account and another discount to apply the percentage off based on the total usage.

When a discount is shared, BRM can access and impact the balance that belongs to the discount owner or the event owner. For example, a discount can read the balance of minutes from the owner account and update the currency balance in the member account.

Discounting can access balances in different accounts for the same discount only when that discount is shared. If the discount is not shared, only the discount owner's balance can be accessed. However, you can still use discounts that are not shared in a discount sharing scenario by creating event balances. See "Using Event Balances in Discounts".

About Snowball Discounts

A snowball discount is a shared billing-time discount that distributes a percentage off to all accounts in a discount sharing group. The percent that is granted to each account can be distributed evenly or based on how much usage each account accrued.

For example, a discount sharing group is owned by account A1 and has telephony services from three member accounts: M1, M2, and M3. The owner account purchases a plan that includes a snowball discount that grants $0.01 for every minute of telephone usage. At the end of a billing cycle, the group has made 5,000 minutes of telephone calls. The owner account and member accounts M1 and M2 made 1,000 minutes of calls each and account M3 made calls totaling 2,000 minutes. When **pin_bill_day** is run, $50 is granted to the group. The owner account and accounts M1 and M2 are granted $10 each, and account M3 is granted $20.

You specify whether a discount is a snowball discount when you create the discount in Pricing Center. To implement a snowball discount, you use two discounts: one that updates a common aggregation balance when accounts incur usage, and another that calculates and distributes the discount based on the aggregated amount. For more information, see "Setting Up Snowball Discounts".

About Discounts Based on Query Values

You can create discounts based on data that is searched for and retrieved from the BRM database. For example, you can create the following types of discounts:

- Billing-time discounts for subscribers' most-called numbers. The list of most-called numbers can be based on such factors as:
  - The number of calls to each number
  - The total connection time

  - The total cost
- Billing-time discounts with discount levels based on the type of call, such as:
  - National, international, and in-network calls
  - The call destination (such as Access Point Names (APNs), SMS servers, and URLs)
- Retroactive discounts based on usage during billing periods that have already passed

You implement these discounts by writing iScripts that retrieve the data required for the discount. Data can be retrieved in two ways:

- Directly from the database with opcode calls in the iScript
- From the EDR, after data returned by customized opcodes is added

See "Setting Up Discounts Based on Query Values" for information about configuring query-based discounts.

You trigger the iScript functions by including **EVAL** tokens in expressions when you create discount models in Pricing Center. An **EVAL** token points to an iScript function that returns the data required for the discount. See "Understanding the EVAL Token" for more information.

You can use provisioning tags to help configure query-based discounts. The provisioning tags define **/profile** objects that can be used to store data for use in discounts. See "Using Provisioning Tags with Query-Based Discounts".

BRM includes iScripts and other sample components that enable you to configure discounts based on the subscriber's most-called numbers. See "Discounts Based on Most-Called Numbers" for more information.

Understanding the EVAL Token

When you create discount models in Pricing Center, you can use **EVAL** tokens in discount expressions to invoke iScript functions. See the Pricing Center Help and "Using Expressions in Discount Models" for more information about discount expressions.

An **EVAL** token in a discount expression calls a specific function in an iScript file, which must be defined in the registry of the DAT_Discount module. This iScript function can include code that calls a BRM opcode. When BRM evaluates the discount expression, the numeric value returned by the function is substituted for the **EVAL** token in the discount expression.

For example, suppose you want to implement different discounts on data usage depending on the access point used. You want to apply a 10% discount for data usage from AccessPointX and a 15% discount for usage from AccessPointZ.

To implement this discount, you could include **EVAL** tokens in the **Base Expression** fields for two discount balance impacts. Each token includes the name of a different iScript function:

- **EVAL("AcPtX")** refers to an iScript function that returns the total amount of data usage from AccessPointX.
- **EVAL("AcPtZ")** refers to an iScript function that returns the total amount of data usage from AccessPointZ.

When BRM processes this discount, the usage amounts for each access point are substituted into the expressions. A 10% discount is applied to AccessPointX usage and a 15% discount to AccessPointZ usage.

Like other discount expressions, expressions including an **EVAL** token can be used in the following discount model components:

- Discount Rule
- Discount Condition
- Discount Balance Impact
- Discount Step

Performance Impacts of Query-Based Discounts that Include Opcode Calls

iScripts that call opcodes can slow the processing of events to which a query-based discount applies. The exact performance impact depends on system configuration and resources.

Some of the impact results from the time taken to route the opcode call to and from the CM. Using a global or wrapper opcode to call individual opcodes is more efficient than calling those same individual opcodes individually from the iScript. Because a wrapper opcode reduces the total number of round trip calls to the CM, performance is improved compared to a series of individual calls.

The bulk of the performance impact results from opcode processing, however. The complexity of the opcodes called therefore plays a large role in the performance overhead of a query.

There is a separate performance impact for each expression in a discount model that uses an **EVAL** token to call an opcode through an iScript function. Because the discounting pipeline evaluates each expression separately, each opcode call is handled separately.

Because of these performance impacts, query-based discounts that use opcode calls should be configured as billing-time discounts. The overhead associated with opcode calls is more acceptable at billing time than during normal event processing.

Important: Pricing Center does not prevent you from including the **EVAL** token in regular discounts, but you should use this functionality for those discounts only with extreme caution.

Using Provisioning Tags with Query-Based Discounts

You can use provisioning tags to define information used in query-based discounts. For example, you can create provisioning tags that define search and aggregation criteria to be used in collecting data for the query. See "About Provisioning Tags" in *BRM Setting Up Pricing and Rating* for general information. See "Using Provisioning Tags for Most-Called-Number Discounts" for specific information.

You define provisioning tags in the **pin_config_provisioning_tags.xml** file. The XML definition specifies the content of a **/profile** object. The data in the **/profile** object can be accessed by opcodes for use in discounting or rating.

You assign provisioning tags to discounts in Pricing Center by selecting the tag on the **Detailed Discount Info** tab of the Discount Attributes dialog box when you create a discount object. (See the Pricing Center Help for more information.) When a discount

with a particular provisioning tag is purchased, BRM creates an instance of the **/profile** object defined by the provisioning tag.

You can use provisioning tags to configure multiple discounts for the same service. For example, you might want to provide one discount for the most-called numbers outside your network and another for the most-called numbers within your network. In this case, you could define two provisioning tags specifying different search and aggregation criteria.

About Setting Up Discounts

You set up discounts in Pricing Center. A complete discount comprises a number of different components that work together:

- A *discount master*, which filters EDRs based on their attributes to determine whether the charge packets should be discounted. See "Filtering EDRs for Discounting". You select the master to use when defining a discount rule.
- A *discount trigger*, which defines the conditions that must be met before a discount can take place. These conditions are based on balances and usage and can include charge, usage quantity, or number of events. For example, to apply a discount for usage over $100, the condition is that the account balance is greater than $100. See "Determining if Usage Qualifies for Discounting".

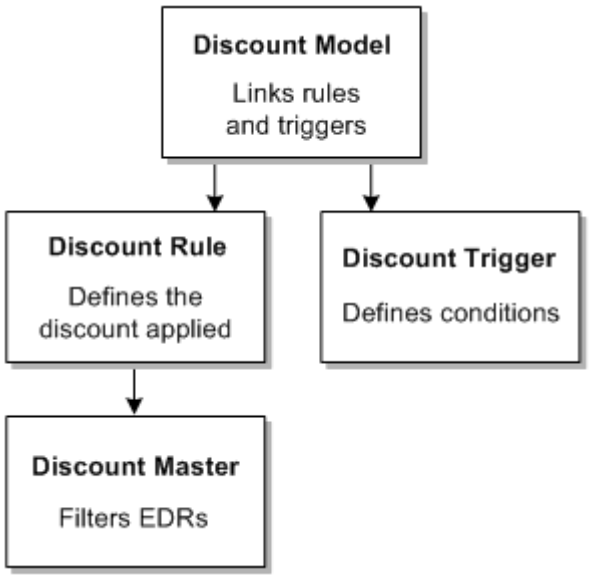
  Note: Not every discount configuration requires a trigger. Although you must include a trigger in the discount, you can configure the trigger to pass every EDR.

- A *discount rule*, which defines the type and amount of the discount. A discount is applied based on the amount or quantity of usage, which is defined by thresholds. The discount can impact multiple resource balances. See "Defining How Discounts Are Applied"
- A *discount model*, which links discount triggers and rules, grouping all the discount components together. See "Grouping Discount Components into Discount Models".

After you configure discount components, you add discounts to price lists. When adding a discount, you map discount models or discount model selectors to the events that are covered by the discount. See "Creating Discounts".

Figure 10–1 shows the relationships among the components of a discount:

Figure 10–1 Discount Component Relationships



Filtering EDRs for Discounting

You filter EDRs to determine whether they contain charge packets that are eligible to be discounted. You filter EDRs by creating *discount masters* that evaluate the event attributes. For example, you can use a discount master to filter for long distance calls made during off-peak time or for local SMS usage. You specify which master to use when you define a rule. See "Defining How Discounts Are Applied".

When discounting processes EDRs, it compares the data in the EDR to the data defined in the discount master. For example, if an EDR includes charge packets for an event that occurred at 6 a.m. and a discount master specifies a time range of 8 a.m to 5 p.m., those charge packets do not pass through the filter.

A discount master includes one or more discount details. The discount details specify the filter criteria. If a master includes more than one detail, an EDR must only pass through only a single detail to be discounted. An EDR must pass through all the criteria in at least one detail before it can be discounted.

You create discount masters and discount details in Pricing Center.

When you create a discount detail, you must enter a starting date. An end date is optional, as are starting and ending times. To pass the filter, the charge packets in the EDR must fall within the range of dates and times specified.

In addition to the date and time criteria, you can filter by the contents of many of the fields that appear in EDRs. For example, if you specify **TEL** for the **Service Code** field, all EDRs that pass the filter represent usage of the Telephony service.

You can enter either specific values or regular expressions for these criteria. The default value for most criteria is a dot followed by an asterisk (**.***), which means that any value is valid. For more information, see "About Using Regular Expressions when Specifying the Data to Extract".

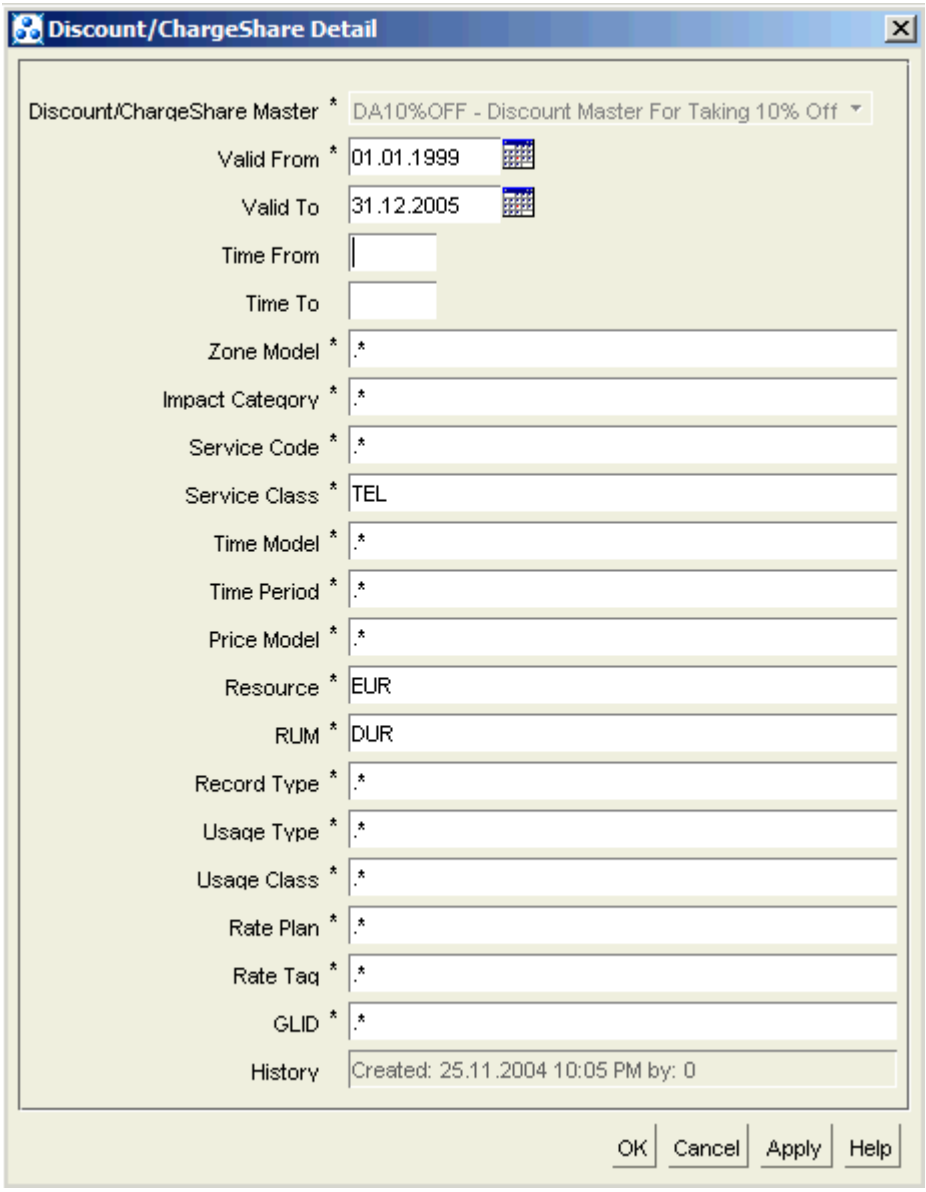
Because real-time events are not processed by the full rating pipeline, the following fields are not available for filtering real-time discounts. You should enter **.*** for these fields for real-time discounts:

- Time Model

- Time Period
- Service Code
- Service Class
- RUM

Figure 10–2 shows the Discount/ChargeShare Detail dialog box, which you use to specify filter criteria:

Figure 10–2 Discount/ChargeShare Detail Configuration



Determining if Usage Qualifies for Discounting

A discount might require usage or balances to reach certain levels before the discount is applied. In this case, you create a *discount trigger* to define the conditions that must be met before the discount can take effect. A trigger is linked to a specific discount rule in a discount model.

Not every discount configuration requires specific conditions to trigger the discount. For example, you may want discounting to process every EDR that passes through the discount master's filters. You must configure a discount trigger, but you can configure it to pass all EDRs.

> **Note:** Discounting does not evaluate charge packets that have both usage quantity as well as charge amount as 0, even if you configure the trigger to pass all EDRs.

A discount can be triggered in many different ways. For example:

- When a specified charge amount is reached; for example, after 50 US dollars worth of usage.
- When usage is less than a particular value; for example, when the number of international call minutes is less than 120.
- When a specified quantity has been reached; for example, a discount for the first 100 minutes and another discount for the second 100 minutes.
- When a specified number of events has occurred; for example, a discount for the first 100 downloads.
- When a combination of conditions are met; for example, for GPRS, when the bytes used is greater than 1 MB and the session duration is longer than 60 minutes.
- When a balance has available resources; for example, available free minutes that can be consumed.

When you define a discount trigger, you define one or more *discount conditions*. If a trigger includes more than one condition, all conditions must be met for the discount to take effect. For example, the trigger for a GPRS discount can include conditions that specify that the total charges in the EDR must be greater than $5 and that the total quantity of received data is greater than 10,000 bytes.

A condition is essentially a comparison of one value or amount to another. The comparison is based on an operator such as **Greater than** or **Less than**. If the comparison statement is true, then the condition is met.

A discount condition includes the following elements:

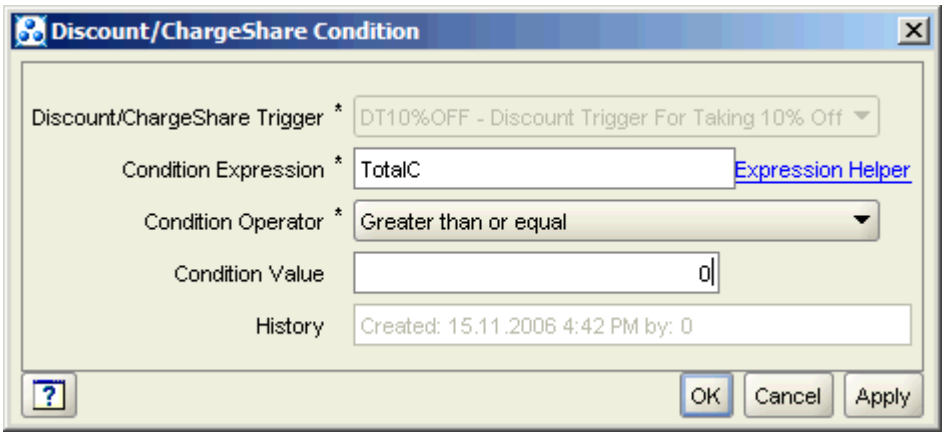
- A **condition expression**. The condition expression specifies the basis of the comparison. The expression can refer to a specific resource balance, total charges, or several other values. It can also include arithmetic operators and decimal constants. (See "Using Expressions in Discount Models".) The condition expression is evaluated, which results in a value that is used for the comparison.
- A **condition operator**. The valid operators are:
  - Greater than
  - Greater than or equal to
  - Less than
  - Less than or equal to
  - Equal to
  - Not equal to
- A **condition value**. This value is compared to the result of the condition expression by using the condition operator. For example, a condition value of **0**

combined with a **Greater than** condition operator stores a positive number, which usually represents a charge to the customer. Conversely, a condition value of **0** combined with a **Less than** condition operator stores a negative number, which represents a credit balance.

You define discount triggers and conditions in Pricing Center.

Figure 10–3 shows the Discount/Charge Share Condition dialog box, where you define a condition for a discount trigger:

Figure 10–3 Discount/ChargeShare Condition Configuration



Defining How Discounts Are Applied

You define the type and amount of the discount as well as the balances that are impacted in *discount rules*. For example, a rule can specify that GSM usage up to 20 minutes be discounted 20% and that usage over that amount be discounted 35%.

When you set up a discount rule in Pricing Center, you define the following attributes:

- **The rule itself**. The rule defines the basic attributes, such as discount master to use for the rule, the DRUM (discount ratable usage metric) expression, the DRUM type, and the rule type. The DRUM defines the amount or quantity of usage to consider for discounting. For example, to discount a call made to a specific area code, the DRUM might be the total charge for that call. See "Defining the Usage Amount to Consider for Discounting".
- **The balance impacts for each discount step**. *Discount steps* define threshold values that affect the amount of discount applied. For example, you can define one step for the first 20 minutes of a call and a second for all usage over 20 minutes. Each step has one or more balance impacts that define which resource balances are affected by the discount and how the balances are impacted. For example, you can specify that the customer's balance of free minutes be debited by the length of the call and that charges for usage not covered by free minutes should be discounted by 10%. See "How Thresholds Define the Amount of Discount Applied".

Defining the Usage Amount to Consider for Discounting

The DRUM is the amount or quantity of usage to consider for discounting. The DRUM can specify several values:

- The total charge or quantity used from the EDR; for example, to discount a single call made to a friend or family member.

- An account's counter balance; for example, to apply a billing-time discount based on the total charges or total minutes used.
- A decimal constant. You typically use a decimal constant when the discount is applied regardless of the usage level; for example, to consume free minutes or apply a fixed discount amount such as 500 bonus points for purchasing a new service.

You enter the DRUM as an expression. The expression can include arithmetic operators and discount expressions. For example, to specify the charge in an EDR, you use the discount expression **TotalC**. To specify an account balance, you use the expression **Bal**(*resource_ID*). For more information, see "Using Expressions in Discount Models".

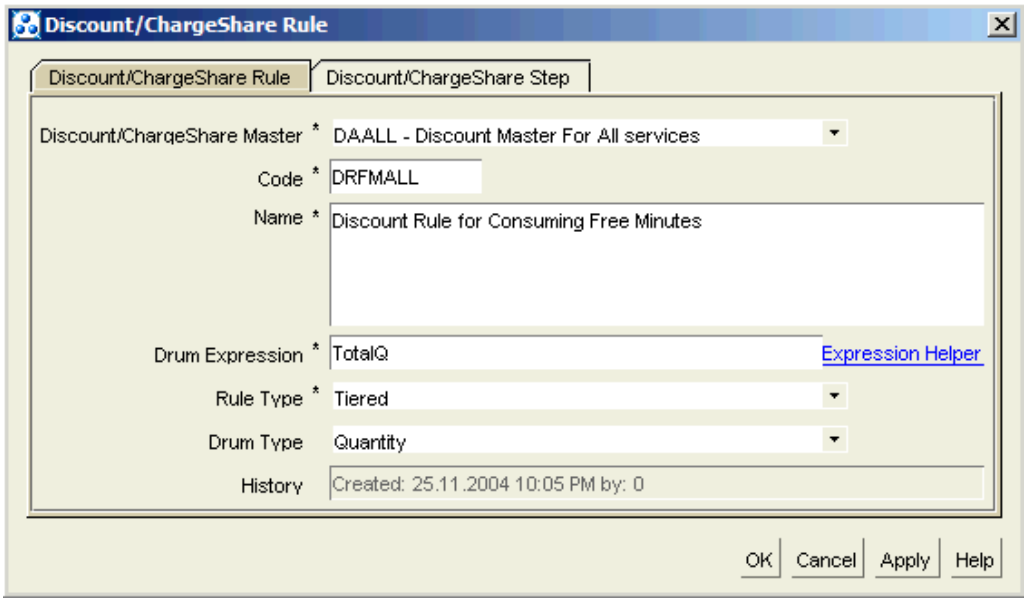
The DRUM expression is evaluated, resulting in a specific amount or quantity. This value is compared with the step thresholds to determine the balance impacts that are applied. Whether the DRUM can overlap a threshold or must fall entirely within it is determined by the rule type. The discount rule includes a **Rule Type** value, which can be set to either tiered or threshold. For more information, see "How Thresholds Define the Amount of Discount Applied".

You also choose the DRUM type, which can be either **Charge** or **Quantity**. This setting determines whether the DRUM and the threshold values are based on monetary charges or on quantities such as bytes.

For more information about possible DRUM values, see "How DRUMs, Steps, and Balance Impacts Work Together".

Figure 10–4 shows the Discount/ChargeShare Rule dialog box, where you specify discount rules and DRUM values:

Figure 10–4 Discount/ChargeShare Rule Configuration



How Thresholds Define the Amount of Discount Applied

The amount of discount applied can be based on any amount of usage, the total amount of usage, or portions of the usage. You define the usage levels by setting their threshold values in discount steps.

To discount one or more portions of the usage, you set up one or more steps in the same discount rule.

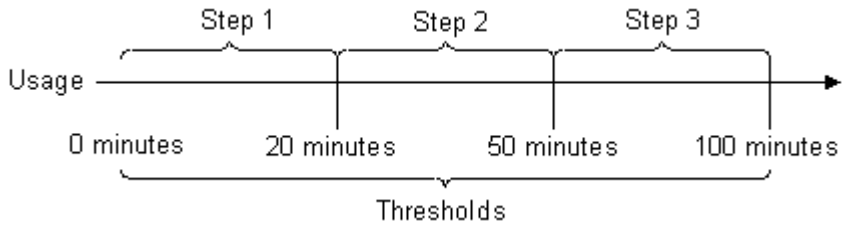
> **Note:** To apply different discounts to the same usage or to select from several possible discounts based on the usage amount, you create a discount model that includes multiple rules and triggers.

Each discount rule can have one or more steps. A step's threshold values determine when the step becomes effective. Each step in turn can have one or more balance impacts, which determine the amount of the discount and the balance that is impacted.

> **Note:** Threshold values for steps should not overlap.

For example, you could define these three steps for a GSM usage shown in Figure 10–5:

Figure 10–5 Example GSM Minute Usage Steps



For each step, you define **Threshold From** and **Threshold To** values, which define the lower and upper limits of the step. The DRUM, which specifies the amount of usage to consider for discounting, is compared to the threshold values in the step. If the DRUM falls within a step's threshold, the discount balance impacts are applied for that step. In Figure 10–5, if the DRUM is the total length of a call specified in the EDR and the call lasted 10 minutes, the usage falls within step 1, so the balance impacts configured for step 1 are applied.

Threshold From must be a decimal value, but the **Threshold To** value can include an expression, typically to reference a balance; for example, the total charge in the charge packet. See "Using Expressions in Discount Models" for more information about expressions.

If the discount should be applied regardless of the level of usage, you specify the threshold as unlimited (0 to infinity).

Defining How Thresholds Are Interpreted

Determining if usage falls within a threshold depends on whether the discount rule type is **Tiered** or **Threshold**:

- In a tiered rule, the amount of usage specified by the DRUM can overlap the threshold values and still qualify for the discount in that step. For example, if the step threshold is 0 to 20 and the call lasts 30 minutes, the discount for that step is applied to 20 minutes of the call.
- In a threshold rule, the total usage must fall within the threshold to qualify for the discount in that step. For example, if the step threshold is 0 to 20 and the call lasts 30 minutes, the discount is not applied. If a second step has a threshold from 20 to

60, the discount for that step is applied to the entire 30 minutes of usage. Threshold rules are useful for applying discounts based on a count, such as the number of months a user has been a subscriber.

Prorating the Threshold Balance Impact

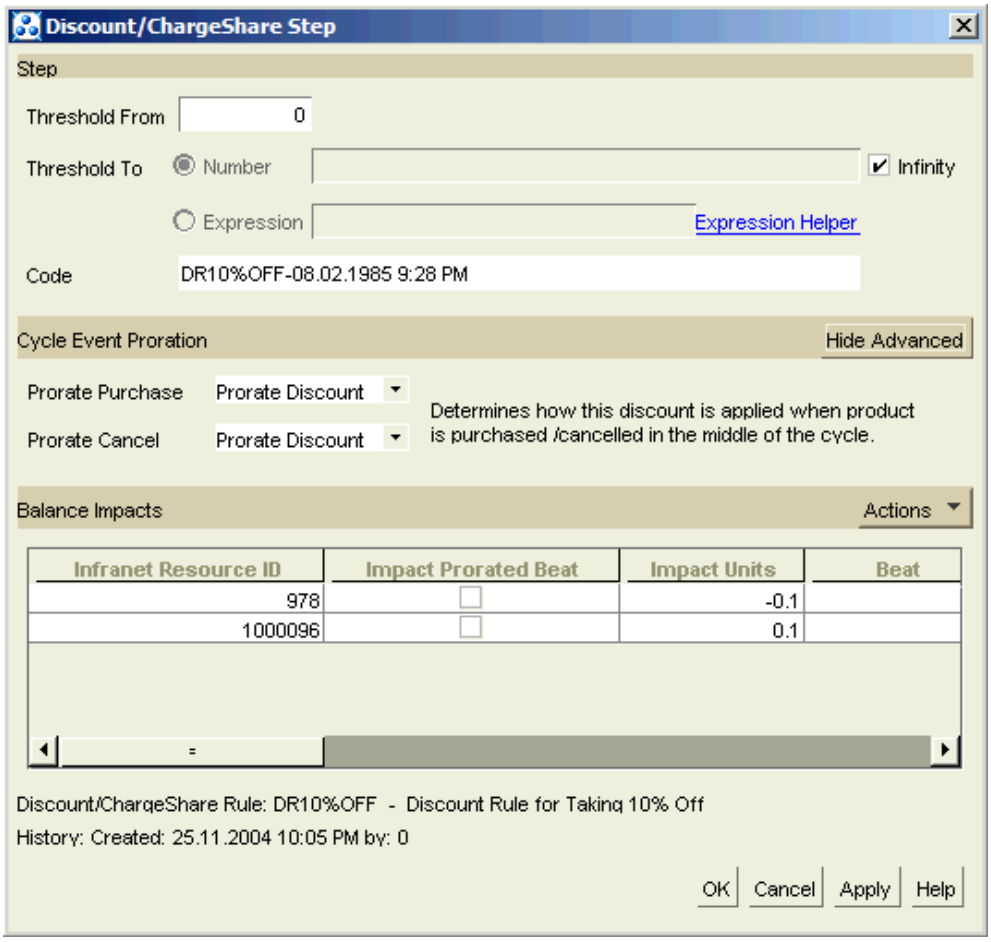
When you define a discount step, you can select options to prorate the balance impact for the amount that falls within the step threshold. These proration options apply only to discounts on cycle fees.

- The **Prorate Purchase** option enables you to choose how a discount should be prorated if a purchase event occurs during the accounting cycle.
- The **Prorate Cancel** option enables you to choose how the discount should be prorated if a cancellation event occurs during the accounting cycle.

For both proration options, you can choose to not apply the discount, to prorate the discount, or to discount for the entire cycle.

Figure 10–6 shows the Discount/ChargeShare Step dialog box, where you specify discount steps:

Figure 10–6 Discount/ChargeShare Step Configuration



Defining the Threshold Balance Impacts

Discount balance impacts determine the actual discount amounts for a step as well as which balances are impacted. A step can have one or more balance impacts. Any resource balance can be impacted.

A discount balance impact can be a percentage or a fixed amount:

- A percentage discount changes the amount to discount by the percentage you enter.
- A fixed amount discount changes the amount to discount by an amount that you enter.

Balance impacts do not necessarily have to be a reduction of an amount like they are for a traditional discount. For example, you can configure a balance impact to award one loyalty point for every minute of usage. In this case, the balance of loyalty points *increases* as a result of the discount.

Some kinds of discounting require two or more balance impacts for each step. For example, if a discount includes free minutes, each step requires two balance impacts: one to discount the charge by 100% and another to reduce the free minutes balance to reflect the usage.

When you set up a discount balance impact in Pricing Center, you enter several different types of information:

- The ID of the resource that is impacted. Resources include currency, free minutes, and so on. You choose from a list of available resources.
- Whether the discount should be applied to the balances of the account that owns the event or the account that owns the discount. This is relevant when the discount is shared in a discount sharing group because the account that generates the usage can be different from the account that owns the discount. Choose **Discount/ChargeShare Owner** to apply the discount to the discount owner's balances. If the discount is not shared, the account that generates the usage is also the discount owner. For more information, see "About Shared Discounts".
- The base expression. This expression determines the basis of the discount calculation. The discount is calculated on the value in the base expression, and the resulting balance impact is applied to the account balance.

  The base expression can include a decimal value, an arithmetic expression, or a reference to a balance. To reference a balance, you use a discount expression. For example, the expression **Bal(1234)** references the account balance for the resource with ID 1234, and the expression **StepC** or **StepQ** respectively references the charge or quantity used in the charge packet. See "Using Expressions in Discount Models".

  Important: If multiple discounts can apply to an event and the discount type is **Cascading**, the base expression must be **StepC** or **StepQ** for all the discount balance impacts. If a different expression is used, discounting ignores that expression and chooses **StepC** or **StepQ** based on the resource type (currency or non-currency).

  When the **EVAL** token is used in discount base expressions, discounting does not consider the proration scale for evaluating the expression. The iScript function triggered by the **EVAL** token should use the prorate scale to calculate the return value. For example, if the iScript function returns the total cost value, the return value should be prorated based on the proration scale value.

When the base expression references a specific resource balance, the resource in the base expression can be different from the resource that is impacted. For example, to add bonus points based on the minutes used, the base expression refers to an aggregation counter for all minutes used, and the resource to impact is the balance of bonus points.

When the base expression references a specific resource balance, it always refers to the balance of the account that owns the discount.

For more information about the values you can specify in the base expression, see "How DRUMs, Steps, and Balance Impacts Work Together"

- The discount amount or percentage. This value is applied to the value in the base expression to compute the discount. For example, a discount percentage is 10 and the base expression refers to a balance of 120. The 10% discount is calculated on 120, which results in a balance impact of 12.

  Enter a positive number to reduce the value of the resource impacted and a negative number to increase it. For example, entering **25%** *reduces* the value; entering **-25%** *increases* it.

  A discount amount is an absolute value. If the discount is not based on the quantity of usage, the absolute discount amount is applied directly to the account balance. For more information, see "How DRUMs, Steps, and Balance Impacts Work Together".

- The beat, or discount increment. For example, to give a monetary credit for each 10 seconds of call usage, the beat is 10.

  Note: The beat is relevant only for amount discounts.

  The beat value determines whether the discount amount is based on the entire value in the base expression or to incremental portions of the base expression value:

  – If the beat is set to **0** or a negative number, the discount amount is applied to the balance. For example, to grant one bonus point for every SMS message sent, set the amount to **1** and the beat to **0**. For this discount, you would specify the bonus point resource as the balance to impact. When the subscriber sends an SMS message, the SMS event triggers this discount and one bonus point is added to the account balance.

    Note: BRM cannot prorate usage discounts when the beat is set to **0**. To prorate a fixed usage discount, use a recurring discount in a product.

  – If the beat is set to a positive number, the value in the base expression is divided by the beat value. The resulting number is then multiplied by the discount amount to arrive at the final discount. The discount calculation looks like this: **discount balance impact = (Base Expression/Beat) *Amount**.

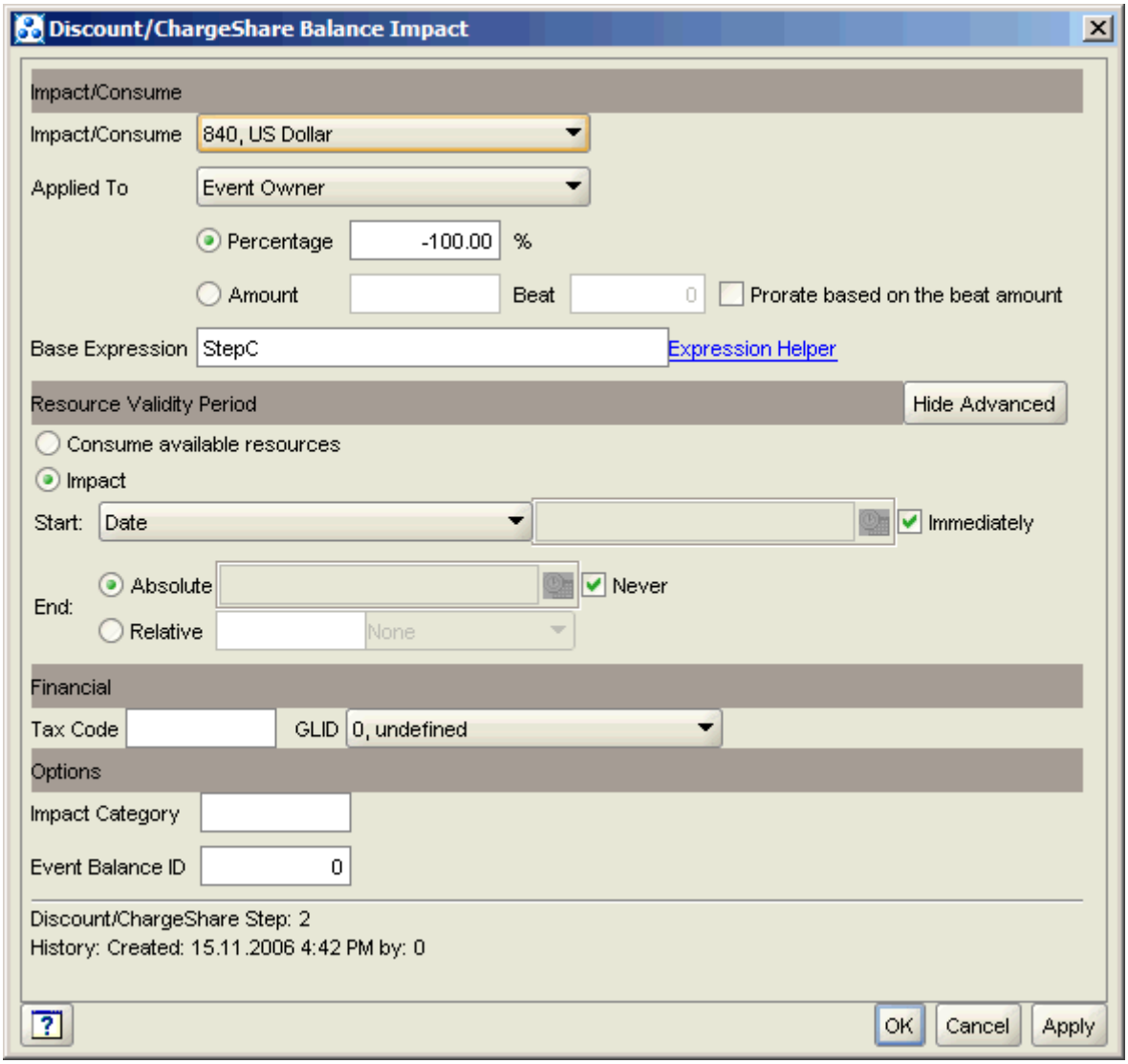
    For example, the base expression references an account balance of 100 minutes of usage. The amount is **1** and the beat is **20** to apply a discount of 1 bonus point for every 20 minutes used. The base expression (100) divided by the beat (20) results in a value of 5. This value is multiplied by the discount amount (1) resulting in a balance impact of 5 bonus points.

You can also apply the discount amount to portions of the usage by using a mathematical expression in the base expression. For example, if **Bal(1000002)** references the account balance of 100 minutes, to apply 1 bonus point for every 20 minutes used, the base expression is **Bal(1000002)/20**. For more information, see "Dividing Base Expressions into Increments".

- Whether to prorate the discount amount for partial beats. For example, if you give a credit of 10 free downloads for every 100 calls, you can credit 5 free downloads for every 50 calls. If you do not specify to prorate the discount, a partial beat is counted as a whole beat, and the full amount of the discount is applied.
- Whether to grant or consume the resource impacted:
  - Select **Consume** to consume a non-currency resource such as free minutes or to discount a currency resource.
  - Select **Impact** only to grant a non-currency resource. You specify the period during which the resource granted, such as free minutes, can be consumed. You can set the resource validity period to start immediately, on a date relative to the grant date, or when the subscriber consumes the resource for the first time (on first usage). For information about resources that start on first usage, see "About Balance Impacts That Become Valid on First Usage" in *BRM Setting Up Pricing and Rating*.
- Financial information, such as the tax code, general ledger (G/L) ID code, and impact category.
- An optional event balance ID. Entering an event balance ID creates a temporary event balance that stores the discount balance impact. The balance impact is not applied to the account balance. You use event balances when you need the results of one discount to calculate another discount. The event balance can be referenced in expressions that you use in discount rules, conditions, and subsequent balance impacts. See "Using Event Balances in Discounts" for more information.

Figure 10–7 shows the Discount/Charge Share Balance Impact dialog box, where you specify the balance impacts for the step:

Figure 10–7 Discount/ChargeShare Balance Impact Configuration



How DRUMs, Steps, and Balance Impacts Work Together

The values you enter in the components of a discount rule are interdependent and are based on the type of discount you are creating. For information about possible values, see the following topics:

- How the DRUM Value, Step Thresholds, and Rule Type Determine the Discount that Is Applied
- Determining the Number of Steps Needed
- Referencing Account Balances in Base Expressions
- Using Absolute Values in Base Expressions
- Dividing Base Expressions into Increments
- Determining the Discount Amount

How the DRUM Value, Step Thresholds, and Rule Type Determine the Discount that Is Applied

The DRUM expression should be set to the amount of usage that is being considered for discounting, which can be either a charge (currency) or a quantity (non-currency) value.

To reference the amount of usage in consideration, enter one of the following values as the DRUM expression:

- **TotalC**, which evaluates to the total charge in the EDR.
- **TotalQ**, which evaluates to the total quantity used in the EDR.
- **Bal(***resource_ID***)**, which references the account's aggregation balance, such as the total minutes used or total accumulated charges.

> **Note:** Currency balances are not loaded into Pipeline Manager memory; therefore, you cannot directly retrieve an account's currency balance for discounting. If the DRUM refers to an account balance, it must be a non-currency balance.

The value of the DRUM expression is then compared with the step thresholds to determine which balance impacts to apply. This is a two-step process:

1. BRM determines which discount steps qualify for discount evaluation. The steps that qualify will have their discounts applied to the account. The type of rule determines whether more than one step can qualify:
   - For a tiered type rule, all steps with a threshold range that overlaps the DRUM range qualify. The DRUM range is 0 to the DRUM value.

     For example, if the DRUM expression evaluates to a quantity of 100 (the DRUM range is 0 to 100):

     - Step 1 with a threshold of 0 to 60 qualifies for evaluation.

     - Step 2 with a threshold of 60 to 120 also qualifies for evaluation.

     - Step 3 with a threshold of 120 to infinity does not qualify for evaluation.
   - For a threshold type rule, only the step with a threshold range that encompasses the DRUM value qualifies for discount evaluation.

     For example, if the DRUM expression evaluates to a quantity of 100:

     - Step 1 with a threshold of 0 to 60 does not qualify for evaluation.

     - Step 2 with a threshold of 60 to 120 qualifies for evaluation.

     - Step 3 with a threshold of 120 to infinity does not qualify for evaluation.
2. BRM determines the amount of usage that receives the discount in each qualifying step by checking the value of the base expression. The value of the base expression also depends on the type of rule:
   - For a tiered type rule, the base expression is typically either **StepC** or **StepQ**. In this case, the value of **StepC** or **StepQ** is the portion of the DRUM range that intersects with the step's threshold range. The step's discount is applied to that amount. For more information, see "Referencing Steps in Base Expressions".

     For example, if the DRUM expression evaluates to a value of 100 (the DRUM range is 0 to 100):

- Step 1 with a threshold of 0 to 60 qualifies: **StepC** or **StepQ** evaluates to 60 and Step 1's discount is calculated on 60 units (such as seconds, minutes, or bytes).

- Step 2 with a threshold range of 60 to 120 also qualifies: **StepC** or **StepQ** evaluates to 40 (100 - 60 = 40) and Step 2's discount is calculated on 40 units.

- Step 3 with a threshold range of 120 to infinity does not qualify, so its discount balance impacts are not applied.

- For a threshold type rule, the base expression is typically **TotalC**, **TotalQ**, or **Bal**(*resource_ID*). In this case, the discount balance impacts of the qualifying step are calculated on the entire amount in the base expression.

  For example, if the DRUM expression evaluates to a value of 100:

  - Step 1 with a threshold of 0 to 60 does not qualify, so its discount balance impacts are not applied.

  - Step 2 with a threshold range of 60 to 120 qualifies: Step 2's discount is calculated on the entire amount of **TotalC**, **TotalQ**, or **Bal**(*resource_ID*).

  - Step 3 with a threshold range of 120 to infinity does not qualify, so its discount balance impacts are not applied.

  Discounts with threshold rules typically apply to the entire amount of a balance: for usage discounts, the total charge (**TotalC**) or quantity (**TotalQ**) in the EDR; for billing-time discounts, an account's aggregation balance (**Bal**(*resource_ID*)). You do not use **StepC** or **StepQ** in the base expression for threshold discounts because this would calculate the discount on only part of the usage rather than the entire usage.

Determining the Number of Steps Needed

At the discount step level, you never need to completely disqualify the entire usage from discounting. Therefore, you set up steps so that the usage falls within at least one step. You use different steps for three types of usage levels:

- To apply a discount for any amount of usage, use only one step with a threshold that is unlimited (from 0 to infinity). In this case, any positive amount specified by the DRUM falls within the threshold and the balance impact is applied without regard to the amount of usage.

  This step is useful for one-time discounts that do not depend on usage; for example, a $15 credit for the purchase of an extra handset. This step can also be used for billing-time or usage discounts when the discount is applied to the entire balance. For example, you can apply a promotional discount of 30% off all usage in the first six months. In this case, the base expression refers to the total usage balance. (See "Referencing Account Balances in Base Expressions".)

- To apply a discount for the total amount of usage, use one step and reference the balance containing the total usage (such as the total charge or number of seconds in the charge packet) as the **Threshold To** value. In this case, the DRUM specifies the minimum amount of usage required. For example, to consume free seconds, only one second of usage is required. To provide 10% off for usage over $100, the minimum amount is 100.

  This step is useful for consuming free seconds, discounting calls within specific networks, and applying discounts such as bonus points based on the total usage.

- To apply different balance impacts to different portions of the usage, use multiple steps. This is the only case in which you need multiple steps. In this case, the

DRUM typically references the balance containing the amount or quantity of usage. For example, a discount that grants bonus points for usage up to 180 minutes, and 10% off for usage over 180 minutes, has two steps, and the DRUM references the account balance containing the total count of minutes used.

Referencing Steps in Base Expressions

When you use a step with a specific threshold because the amount of usage is significant, you always reference the amount from the step in the base expression. By referencing the step, you calculate the discount on the amount of usage that falls within the step threshold.

To reference the step, you enter the discount expression **StepC** if the resource to impact is currency or **StepQ** if the resource to impact is non-currency. **StepC** evaluates to the currency charge for the amount that falls within the step threshold. **StepQ** evaluates to the non-currency quantity that falls within the step threshold.

For example, in a tiered rule, the DRUM is 120 minutes and the step threshold is between 0 and 60. A base expression of **StepQ** evaluates to 60 minutes because that is the amount of the usage (the DRUM) that falls within the step threshold. The discount is then calculated for a value of 60 and the resulting balance impact applied to the account balance.

Referencing Account Balances in Base Expressions

You refer to an account balance in the base expression to calculate the discount for the entire balance. For example, to grant a billing-time discount of 10 bonus points for every 60 minutes of usage, you reference the total usage balance. In this case, any amount of usage is considered, so the DRUM is **1** and the step threshold is unlimited.

Using Absolute Values in Base Expressions

When the discount is an absolute amount that is applied regardless of the usage level, the base expression can be any positive number because you do not need to calculate the discount. For example, to apply a promotional discount of 50 free minutes for the first six months, The DRUM is **1**, the threshold unlimited, and the base expression is **1**. The discount amount (50 free minutes) is then applied directly to the resource balance.

Dividing Base Expressions into Increments

There are two ways you can apply a discount amount to portions of the usage. For example, to grant 10 frequent flyer miles for every hour of telephony usage, use one of the following methods:

- Specify a beat value. If you use a beat, you enter the following values:
  - The base expression is **Bal**(*total_usage*); the balance that tracks the total minutes used.
  - The beat is 60 (assuming the resource is minutes).
  - The discount amount is 10.
  - The balance to impact is the resource of frequent flyer miles.
- Use mathematical operators in the base expression. In this case, you enter the following values:
  - The base expression is **Bal**(*total_usage*)**/60**: the total minutes used divided by 60.
  - The beat is 1.

- The discount amount is 10.
- The balance to impact is the resource of frequent flyer miles.

In both of the above examples the discount calculation is the same. For example, if the balance of total minutes used is 300, the discount calculation is **(300/60) * 10 = 5**.

Determining the Discount Amount

When the discount is a percentage, the percentage is multiplied by the value in the base expression to determine the balance impact.

When the discount is an absolute amount, that amount can be one of the following:

- The actual amount that should be applied to the account balance. For example, to grant 100 free minutes as a birthday bonus, the amount is 100.
- An amount that should be applied to portions of the usage. For example, to grant 10 frequent flyer miles for every hour of usage, the amount is 10 and the beat is 60 (assuming the resource is minutes).
- An amount that should equal the usage. For example, to consume one free minute for every minute used, the amount is 1 and the beat is 1. This is also useful for copying an account balance into a temporary event balance. (See "Using Event Balances in Discounts".)

Grouping Discount Components into Discount Models

A discount model links discount triggers with discount rules. Each discount model can have one or more *discount versions*, which are defined by validity periods. Only one version can be valid at any one time. During rating, BRM uses the discount model version that is valid at the time the discounted event occurred.

Each version includes one or more *discount configurations*. A discount configuration associates a trigger with a discount rule for a particular discount model version. When all of the trigger's conditions are met, the discount rule is evaluated for eligible EDRs.

> **Note:** A trigger is not required in a configuration. If you want discounting to process all EDRs that pass through the discount master associated with a rule, do not include a trigger in the configuration.

You create different discount configurations to provide various types of discounts; for example, discounts that provide a percentage off and discounts for consuming free minutes.

You can mix and match rules and triggers to meet your business needs. You can use the same rules and triggers in more than one model. You select the model you need when you create the purchasable discount. See "Creating Discounts".

When you set up a discount model configuration, you specify whether it is cascading, parallel, or sequential. This setting and the cascading/parallel/sequential setting in the **/discount** object determine whether discounts are applied to values from the entire charge packet or only to the amount not yet discounted.

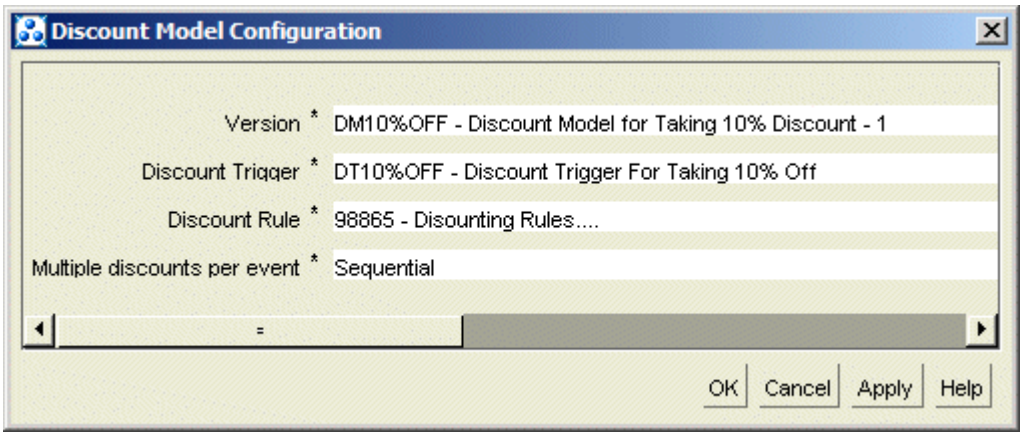
For example, if an EDR meets the conditions for two discount configurations that are set for cascading, discounting first applies the discount configuration that has a rule with the highest priority. If there is still an amount left in the discount balance account, the discount configuration that has a rule with the second highest priority is applied.

Note: Cascading discounts are designed for discounts that consume non-currency resources or that discount currency charges, not for discounts that grant resources.

For more information, see "About Cascading, Parallel, and Sequential Discounts".

Figure 10–8 shows the Discount Model Configuration dialog box:

Figure 10–8 Discount Model Configuration



Prioritizing Discount Model Components

You can prioritize the order in which discounting evaluates certain discount model components:

- Discount details in a discount master
- Discount steps in a discount rule
- Discount model configurations in a discount model version

For each charge packet, discounting evaluates the ranked components in order and Uses the first one that is valid.

Creating Discounts

When you create a discount in Pricing Center, BRM creates a **/discount** object. The **/discount** object is the top-level discounting component and links together the other components.

Discount objects are purchasable items similar to products. You include them in deals along with products.

You can create the following types of discount:

- Subscription
- System

When you create a discount object, you specify which events are covered by the discount. For example, you can include the following events:

- Delayed GSM Session Event
- Monthly Cycle Forward Event

For each event, you choose either a discount model or discount model selector that will be used to discount the event. For example, for the Delayed GSM Session Event, you could choose a discount model that supplies free minutes; for the Monthly Cycle Forward Event, you could choose a model that applies a percentage discount. See "Grouping Discount Components into Discount Models" for more information.

You map events to discount models and model selectors in the **Map an Event to a Discount Model/Model Selector** section of the Discount Attributes dialog box in Pricing Center.

You can also flag an event for inclusion in a discount that is used to distribute the results of a discount among group members. See "About Snowball Discounts".

Other attributes you specify for discount objects include the following:

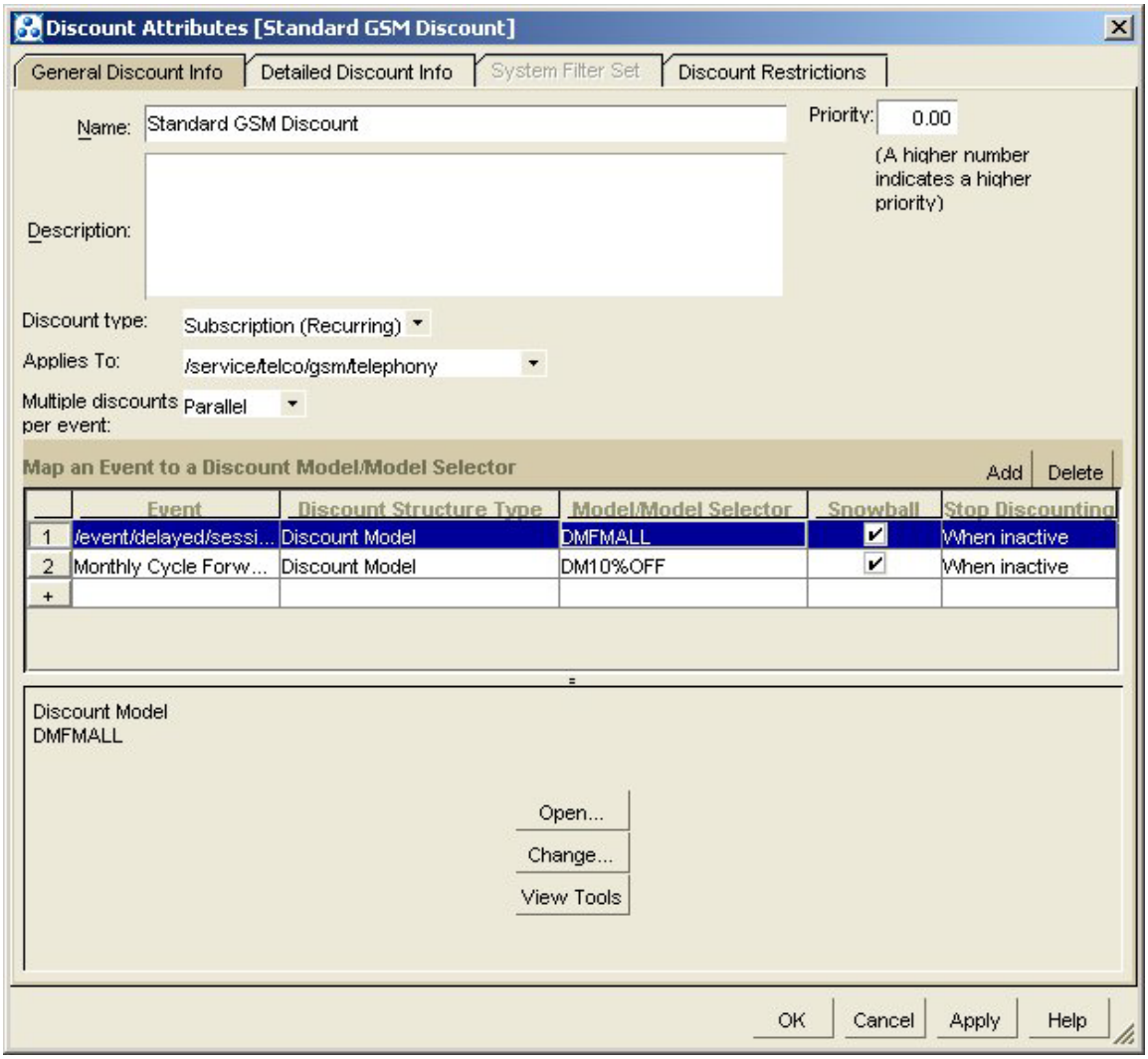
- The purchase level. This determines whether the discount applies to events related to all services owned by the account or only to a specific service.
- How to handle multiple discounts. See "About Cascading, Parallel, and Sequential Discounts".
- The discount priority. When an account owns more than one discount, the priority determines the order in which they are processed.
- Whether to continue discounting if the discount is canceled or inactivated.
- (Optional) A provisioning tag. You can use a provisioning tag to configure a discount's service with additional information. For example, you can use a provisioning tag to create an extended rating attribute (ERA). See "Working with Provisioning Tags" in *BRM Setting Up Pricing and Rating*.
- Start and end dates.
- Discount quantity allowed.
- Ownership quantity allowed.
- Whether the discount is valid for the entire month or only part of the month when it is purchased or canceled mid-cycle. See "Prorating Discount Balances".
- Whether the discount can be purchased with other discounts or whether it is mutually exclusive of other discounts. See "About Discount Exclusion Rules". You define discount exclusion rules in the **Discount Restrictions** tab of the Discount Attributes dialog box.

If a rate plan has discounts, enable the **Override Credit Limit** option for the rate for all products in Pricing Center.

> **Caution:** If an event can be discounted but the required resources to rate the event exceed the available resources and the **Override Credit Limit** option is not enabled, the rating engine does not rate the event correctly. If the available currency resource is 0, the rating engine fails with the "Credit limit exceeded" message without applying the available discounts.

You define discount objects in the Discount Attributes dialog box in Pricing Center, as shown in Figure 10–9:

Figure 10–9 Discount Attributes Configuration



For more information, see "Defining a Discount Based on the Number of Subscriptions".

Prorating Discount Balances

When you use BRM to manage customers, you can create or change a customer's discount balance. When you do so, you often need to prorate the discount balance to handle changes that do not coincide with the customer's accounting cycle; for example:

- If the account creation date is different from the billing day of month, the customer will have a partial accounting cycle. For example, if the account is created on July 15, but the billing day of month is the 1st, the customer has a partial accounting cycle of 15 days. If you give 100 free minutes per month, you can prorate those free minutes and give only 50 free minutes for the partial accounting cycle.
- If customers change a service in the middle of an accounting cycle, they might purchase different discounts. In that case, you can cancel the existing discount balance and delete the prorated amount.

For information about the proration settings, see "About Applying Discounts Activated or Canceled in Mid-Cycle".

For information about how BRM calculates prorated cycle fees, see "Calculating Prorated Cycle Fees" in *BRM Configuring and Running Billing*.

You define discount validity rules in the **Detailed Discount Info** tab of the Discount Attributes dialog box.

Using Expressions in Discount Models

You can use *expressions* when defining several different discount model components. Unlike a normal value that is written directly into the database, an expression is evaluated by BRM to produce a value.

For example, if you enter **Bal(444)** into a field, this expression is evaluated to mean the current balance of the resource with ID 444.

An expression can include more than one *token* or individual element. For example, if you use a discount to provide one free SMS message for every 100 minutes of telephony usage, the discount balance impact could include the expression **Bal(1001)/100**, where **Bal(1001)** refers to the current total of telephony minutes.

Note: Currency balances are not stored in Pipeline Manager memory; therefore, you cannot reference an account's currency balance in a discount expression. If you need to use an account's currency resource to calculate a discount, set up an aggregation counter and use a separate discount to update the counter balance when usage occurs. You can then use this counter balance to calculate the discount amount.

You can use expressions when defining discount conditions and rules:

- **Discount condition**

  You use a discount expression in the **Condition Expression** field. For example if the discount consumes free minutes, you use the **Bal(***resource_ID***)** expression to reference the account balance that contains those free minutes. You can then use the condition to check whether there are free minutes available for consumption.

  If you grant a percentage off for all usage over 300 minutes, the condition expression can reference the account balance that contains the total number of minutes used. You can then use the condition to check that the subscriber used over 300 minutes to qualify for the discount.

  For more information about conditions, see "Determining if Usage Qualifies for Discounting".

- **Discount rule**

  In a discount rule, you can use an expression when defining the DRUM, steps, and balance impacts:

  – **DRUM**

    You can use a discount expression in the **DRUM Expression** field. If the amount of usage is relevant to the discount (defined by limiting the step threshold), the DRUM refers to the usage balance. This can be the total usage in the EDR or an account's counter balance.

    For example, to discount calls made within a specific network, use the expression **TotalC**, which evaluates to the total charge in the EDR. To consume

free minutes, use the expression **TotalQ**, which evaluates to the total quantity used in the EDR.

To apply a billing-time discount based on the total usage, use the expression **Bal(***resource_ID***)** to refer to the account's usage balance; for example, the aggregation balance that tracks total charges.

For more information about DRUMs, see "Defining the Usage Amount to Consider for Discounting".

- **Step**

  You can use a discount expression in the **Threshold To** field. This is typical when the usage amount is relevant but there is only one step. For example, when a discount consumes free minutes, you use the **Bal(***resource_ID***)** expression to refer to the account's balance of free minutes. This limits the consumption to the amount in the balance.

  For more information about steps, see "How Thresholds Define the Amount of Discount Applied".

- **Balance Impacts**

  You can use a discount expression in the **Base Expression** field. The base expression is the amount for which to calculate the discount. You can reference a non-currency account balance or the amount of usage that falls within the step threshold. You always use the amount from the step when the **Threshold To** value is specific (not unlimited).

  To refer to an account balance, use the expression **Bal(***resource_ID***)**. To refer to the amount from the step, use the expression **StepC** or **StepQ**. **StepC** evaluates to the currency charge for the amount that falls within the step threshold. **StepQ** evaluates to the non-currency quantity that falls within the step threshold.

  For example, to apply a billing-time discount based on the total usage for the month, the base expression refers to the account balance (**Bal(***resource_ID***)**) that tracks the total usage. To reduce a balance of free minutes by the number of minutes used, the base expression is **StepQ** (the number of minutes used that fall within the step threshold). To discount the currency balance for free minutes used, the base expression is **StepC** (the charge for the minutes used that fall within the step threshold).

  Important: If multiple discounts can apply to an event and the discount type is **Cascading**, the base expression must be **StepC** or **StepQ** for all the discount balance impacts. If a different expression is used, discounting ignores that expression and chooses **StepC** or **StepQ** based on the resource type (currency or non-currency).

  For more information about balance impacts, see "Defining the Threshold Balance Impacts".

Using Event Balances in Discounts

An event balance stores the results of a discount balance impact for a single event. You use event balances when you need the results of one discount to calculate another discount for the same event. This is useful when:

- Sharing discounts or applying a discount to one account based on a balance from another account. For example, one discount records the number of minutes used by account A and stores it in an event balance. A second discount reduces the number of free minutes in account B based on the amount in the event balance. In this example, the event balance is passed between separate discounts.
- Applying discounts based on two or more attributes of the EDR. For example, one discount records the number of minutes used and stores it in event balance 1. A second discount records the number of bytes sent and stores it in event balance 2. A third discount calculates the discount based on the values in event balance 1 and 2. In this example, the event balance is passed between several discount configurations in the same discount. For a complete description of this example, see "Example of Using Event Balances to Discount Based on Multiple EDR Attributes".

Unlike other balances, event balances are temporary. They are maintained only while a single event is in the process of being discounted. Balance impacts stored in event balances are not included in the discount packet that records the results of a discount. If multiple discounts are applied to a single event, the event balance is maintained until all the discounts are processed. However, for billing-time discounts, event balances are only maintained while a single discount is processed. See "Using Event Balances in Billing Time Discounts".

Because the balance impact in an event balance is not applied to account's resource, it does not matter whether you select to impact the event owner or the discount owner, or whether to impact or consume the resource when you define the discount balance impact. For more information, see "Defining the Threshold Balance Impacts".

Just as with other balances, you can use event balances in expressions that determine whether an event qualifies for discounting and how the discount is calculated. For example, a discount condition can include an event balance expression that requires GPRS usage in a event to be greater than 1 MB. In this example, the event balance stores the number of bytes used. (See "Determining if Usage Qualifies for Discounting" for information about discount conditions.)

The way you reference event balances is different from the way you reference other balances, however. Before referencing an event balance, you must define it. You define event balances in discount balance impacts by entering an ID in the **Event Balance ID** field. Specifying an ID in this field causes an event balance with that ID to be created when the balance impact is processed.

You reference an event balance by using the **EBal** expression with the appropriate ID number. For example, if you create an event balance with the ID 99, you reference it with the expression **EBal(99)**. (See "Using Expressions in Discount Models".) You can reference the event balance in the same fields that can take discount expressions: in conditions, steps, and balance impacts.

Using Event Balances in Billing Time Discounts

You cannot use event balances for billing-time discounts to pass a balance impact from one discount object to another. Event balances are deleted at the end of the discount transaction. When discounting usage events, all discounts for that event are processed in the same transaction. However, for billing events, each discount associated with the event is processed in a separate transaction; therefore, the event balance is deleted after the discount that creates the event balance is processed.

Example of Using Event Balances to Discount Based on Multiple EDR Attributes

An EDR can potentially include many different charge packets containing charges and amounts for several different ratable usage metrics (RUMs), such as duration or bytes received. To narrow the scope of the event balance, you use discount masters to filter in the charge packets you want. This capability is illustrated in the following example.

You can create a 10% discount on total charges for GPRS usage when the session duration is 60 minutes or longer and the number of bytes sent or received is 1 MB or greater. This example uses two event balances: one records the minutes used and another records the number of bytes sent or received.

This discount requires a discount model with three discount configurations:

- In the first configuration, you filter in charge packets that record the event duration and then save that duration in an event balance. To filter in charge packets recording duration, you create a master that specifies duration as the RUM.

  In the balance impact for the step and rule associated with the master, you create an event balance, **EBal(1)**. You configure the rule and balance impact to record the total quantity in the charge packet that passed the filter by setting the following values:

  - In the rule, set the DRUM expression to **TotalQ**, or the total quantity in the charge packets that passed the filter. (In this case, **TotalQ** reflects the duration of the event.)
  - Make the step threshold unlimited.
  - Set the discount amount to 1 and specify a beat of 1.

  A trigger is not necessary in this configuration.

- In the second configuration, you filter in the bytes sent and received, then record them in a second event balance, **EBal(2)**. You set up the second configuration in the same way you setup the first, except that you specify bytes as the RUM and supply a different event balance ID.
- In the third configuration, you set up a master to filter in all EDRs for the GPRS service. You also set up a trigger with two conditions. One condition specifies that **EBal(1)** must be greater than or equal to 60 and the second specifies that **EBal(2)** must be greater than or equal to 1024. The balance impact for the rule and step triggered by these conditions discounts the total charges (**TotalC**) by 10%.

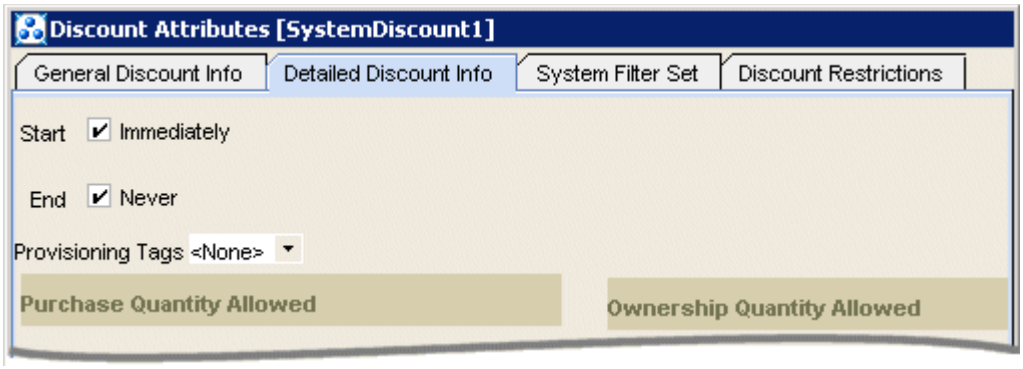
Using Provisioning Tags in Discounts

You can use custom provisioning tags to include ERAs in discounts. This enables you to vary the discount based on an attribute of an event.

Important: You cannot use provisioning tags with discounts for telco services.

When creating discounts in Pricing Center, you choose the provisioning tag on the **Detailed Discount Info** tab of the Discount Attributes dialog box as shown in Figure 10–10:

Figure 10–10 Provisioning Tag Configuration in Detailed Discount Info Tab



You use provisioning tags to specify opcodes to run and parameters for the opcodes to use when a discount is purchased and canceled. For example, a provisioning tag can create a custom profile used in a discounting policy, such as an ERA.

To create provisioning tags, see "Working with Provisioning Tags" in *BRM Setting Up Pricing and Rating*.

About Discount Model Selectors

A discount model selector chooses a discount model during rating based on specific values in an EDR. This enables you to provide discounts of different amounts for different scenarios in a single discount object.

You can use discount model selectors to apply preferential, promotional, or other selective discounts based on EDR characteristics.

The elements of a discount model selector, and the relationship between them, are as follows:

- A *discount model selector* contains one or more configurations. You rank the configurations in the order in which you want Pipeline Manager to evaluate them.
- A *configuration* maps a price/discount model selector rule to a discount model. Each configuration has a validity period.

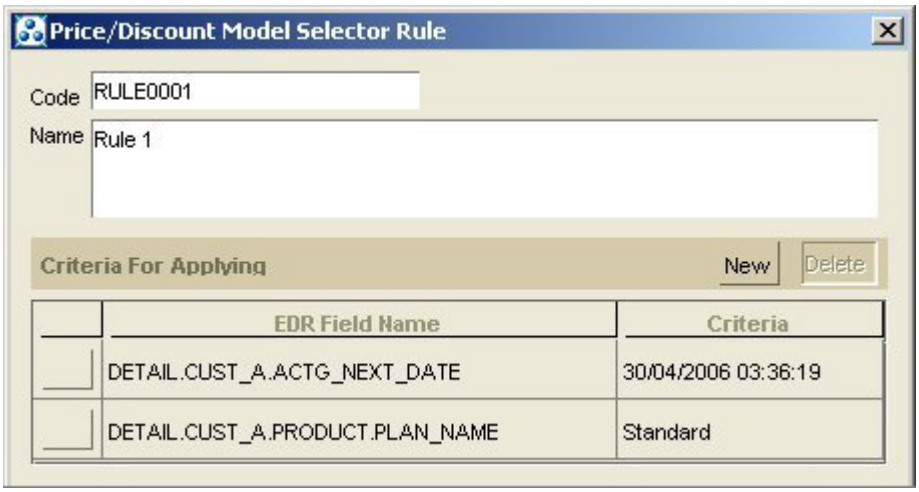
Figure 10–11shows the Price Model Selector Configuration dialog box, where you map a price/discount model selector rule to a discount model.

Figure 10–11 Price Model Selector Configuration

- A *price/discount model selector rule* associates an EDR field with a specific value. A rule can contain one or more such associations, all of which are logical ANDs and must be in the EDR for the rule to be satisfied.

Figure 10–12 shows the Price/Discount Model Selector Rule dialog box, where you associate an EDR field with a specific value.

Figure 10–12 Price/Discount Model Selector Rule Configuration



Note:

- You can use any EDR field that is defined in the data dictionary except EDR fields that have the data format of **Block**.
- You use the same price/discount model selector rules for both discount model selectors and price model selectors.

Pipeline Manager evaluates the configurations in a discount model selector in the order in which you rank them. When a rule in a configuration is satisfied (that is, when the specified EDR fields contain the specified values), Pipeline Manager uses the discount model that is mapped to that rule and ignores any configurations that are ranked lower. If an EDR does not contain the specified values, it does not qualify for the associated discount.

You set up discount model selectors in Pricing Center.

You also need to configure the Pipeline Manager DAT_ModelSelector module. See "DAT_ModelSelector".

Example of Using a Discount Model Selector

In this example, you rate a discount model selector to apply a specific discount model only to EDRs for particular roaming partners.

The following are the overall steps for setting up this example:

1. Create and configure a *discount model* that gives the discount you want to apply.
2. Create a *price/discount model selector rule* that maps the EDR fields DETAIL.SOURCE_NETWORK and DETAIL.TARIFF_CLASS to the names of the network and the tariff class that qualify for this discount.

Create as many rules as you need to represent all the roaming partners to whom you want to apply this discount.

3. Create a *discount model selector* that maps the discount model you created to the rules you defined.
4. In the price list, add a discount with a discount usage map that maps your discount model selector to the event to which the discount applies.
5. Create a customer to purchase the discount you created.

When an EDR for this customer is for both the destination network and tariff class in the rule, the rule is satisfied and the associated discount model is used. EDRs that do not contain both these pieces of information do not qualify for this discount.

About Cascading, Parallel, and Sequential Discounts

The same EDR can be eligible for discounting by multiple discounts and by multiple discount configurations within each discount. For example, two different discounts can be applied to the same event type. Similarly, within a discount model, filters and conditions can pass the same EDR to multiple discount model configurations for processing.

You use the **Multiple discounts per event** setting (**Cascade**, **Parallel**, or **Sequential**) at the *discount object level* and the **Cascading**, **Parallel**, or **Sequential** setting at the *discount model configuration level* to determine how EDRs should be processed when multiple discounts and configurations apply. For more information, see "Creating Discounts" and "Grouping Discount Components into Discount Models".

These multiple discount types work as follows:

- **Cascading discounts** are evaluated so that no part of the charge packet is used as the basis for more than one discount. A cascading discount can be used only if part of the charge packet has not yet been evaluated for a discount. In general, cascading results in smaller discounts.

  Note: Cascading discounts are designed for discounts that consume non-currency resources or that discount currency charges, not for discounts that grant resources. If you specify **Cascading** for a discount, you must also configure the discount balance impact to consume available resources, otherwise the discount calculated will be incorrect.

- **Parallel discounts** are evaluated independently of each other. The entire charge packet is discounted, regardless of whether it has been discounted previously. In essence, discounting starts over from the beginning for each discount. This usually results in larger discounts.
- **Sequential discounts** are applied as long as a customer charge remains. The size of these discounts usually falls between those of cascading and parallel, but not always. This always reduces the customer charge, irrespective of whether the discount value is negative or positive.

Examples of Using Multiple Discount Types

To understand how the cascading/parallel/sequential setting works at the discount object level, consider the following two examples:

- A customer has purchased a deal that includes two discounts that apply to GSM usage. Usage is $0.10 per minute. One discount has a higher priority, provides a 10% discount, and is present in the cascaded mode. The other discount applies a 20% discount to all GSM usage charges. The customer makes a 100-minute ($10) call.

  The 10% discount is processed first because it has a higher priority. The call is discounted 10%, resulting in a $1 discount.

  - If the second discount is set to **Cascading**, no further discounts apply because the entire charge packet has already been evaluated by the previous discount. The total charge for the call is $9.
  - If the second discount is set to **Parallel**, it applies the 20% discount to the entire charge packet, resulting in a second discount to the entire 100-minute event. This results in another discount of $2, which is combined with the first discount of $1, for a total charge of $7.
  - If the second discount is set to **Sequential**, it applies to the remaining charge, which is $9. The second discount is 20% of $9, or $1.80. The discounts combine for a total charge of $7.20.

- Another customer makes a 100-minute call ($10), but that customer's deal has different discounts. The first discount is 50 free minutes, and the second discount is 20% off all minutes.

  The first discount, which has the highest priority, is processed first. The discount determines that there are 50 minutes that can be provided for free and provides them.

  - If the second discount is set to **Cascading**, the customer gets 20% off the remaining portion (50 minutes) and so pays a total charge of $4. The cascading discount applies in this example because the first discount did not evaluate the entire charge packet.
  - If the second discount is set to **Parallel**, it applies the 20% discount to the entire 100 minute call, even though there was no charge for the first half. The discount is 20% of $10. This results in another discount of $2. The total charge for the call is $3.
  - If the second discount is set to **Sequential**, it applies to the remaining charge, which is $5, resulting in a charge of $4. In this case, the final charge to the customer is the same as that provided by a cascading discount.

Examples of Using Discount Configurations in Discount Objects

Within a discount model, the **Cascading**, **Parallel**, and **Sequential** settings in the discount model configuration work the same way as the **Cascade**/**Parallel**/**Sequential** setting in discount objects. If the discounts in the previous examples were included in two configurations within the same model, the results would be the same as they are for separate discounts.

The discount type settings on discount objects and the discount model configurations they include can differ. The discount configuration settings (particularly cascading) in one object affect the way that discounts are evaluated by configurations in subsequent objects. The complex discounting structures such as those described below are not typical, but may help you understand the implications of your settings.

Consider a scenario that includes two discount objects, one with two discount configurations.

- Discount Object 1: any type

  - Discount Configuration A: 10% off if the charge is between $0 and $60
- Discount Object 2:
  - Discount Configuration B: 20% off
  - Discount Configuration C: 10% off
  - There is a charge of $100.

The following examples show how the discounts are calculated differently depending on whether:

- Discount Configuration A is set to **Cascading**, **Parallel**, or **Sequential**.
- Discount Object 2 is set to **Cascading**, **Parallel**, or **Sequential**.
- The configurations included in Discount Object 2 are set to **Cascading**, **Parallel**, or **Sequential**.

Example one

Consider the following objects and configurations:

- Discount Object 1: any type
  - Discount Configuration A: any type
- Discount Object 2: **Parallel**
  - Discount Configuration B: **Sequential**
  - Discount Configuration C: **Sequential**

The following rules apply:

- Discount Configuration A applies to charges up to $60, so only $60 is considered for discount. The discount is **60 * 10% = $6**.
- Discount Object 2 is parallel, so it ignores previous discounting and applies to the original amount of 100 minutes and charge of $100.
- Discount Configuration B is sequential so it applies to the entire charge evaluated by Discount Object 2. The discount is **100 * 20% = $20**.
- Discount Configuration C is sequential so it applies to the charge that remains after Discount Configuration B is applied, **80.00 * 10% = $8**.

The final charge is $66.

Example two

Consider the following objects and configurations:

- Discount Object 1: any type
  - Discount Configuration A: **Cascading** versus **Parallel** or **Sequential**
- Discount Object 2: **Cascading**
  - Discount Configuration B: **Cascading**
  - Discount Configuration C: **Parallel**

The following rules apply:

- Discount Configuration A discounts on charges up to $60, (that is, only $60 is considered for discount). **The discount is 60 * 10% = $6**. **The remaining charge is $94 (100 - 6)**.

- Discount Object 2 is cascading, so only the part of the charge packet that has not been evaluated yet is considered for discount.
- Discount Configuration B is cascading. The application of a cascading discount changes if a cascading configuration discount has already been applied.

  If Discount Configuration A is parallel or sequential, the basis for Discount Configuration B is **94.00 * 20% = $18.80**.

  However, subsequent cascading discounts apply to the part of the charge packet that has not been evaluated for discount. If Discount Configuration A is cascading, the basis for Discount Configuration B is **40 * 20% = $8**.
- Because Discount Configuration C is parallel, it applies to the full amount of the charge packet that Discount Object 2 evaluates.

  If Discount Configuration A is cascading, Discount Configuration C provides **40 * 10% = $4**.

  If Discount Configuration A is parallel or sequential, Discount Configuration C provides **94 * 10% = $9.40**.

The final charge possibilities are $82 (if Discount Configuration A is cascading) or $65.80 (if Discount Configuration A is parallel or sequential).

Example three

Consider the following objects and configurations:

- Discount Object 1: any type
  - Discount Configuration A: cascading versus parallel or sequential
- Discount Object 2: sequential
  - Discount Configuration B: sequential
  - Discount Configuration C: cascading

  The following rules apply:
  - Discount Configuration A discounts on charges up to $60, so only $60 is considered for discount. **The discount is $60 * 10% = 6**. **The remaining charge is $94 (100 - 6)**.
  - Discount Object 2 is sequential, so it applies to the remaining amount of $94.
  - Discount Configuration B is sequential, so it applies to the entire remaining charge. **The discount is $94 * 20% = 18.80**. **The remaining charge is $76.20**.
  - Discount Configuration C is cascading, but it is in a sequential object. If a cascading configuration discount is in a parallel or sequential discount object, the cascading configuration applies to parts of the charge packet that have already been evaluated.

    If Discount Configuration A is parallel or sequential, Discount Configuration C provides **76.20 * 10% = $7.62**.

    If Discount Configuration A is cascading, the discount for Discount Configuration C is based on the amount remaining from Discount Configuration A; that is **100 - 60 = $40**. Subtract from this amount the discount by Discount Configuration B, **40 - 18.80 = $21.20**. Discount Configuration C provides **21.20 * 10% = $2.12**.

The final charge possibilities are $74.58 (if Discount Configuration A is cascading) or $68.58 (if Discount Configuration A is parallel or sequential).

Sequential Discounting of Cycle Fees

Sequential discounting of cycle fees is a process that evaluates cycle fee discounts that are purchased or canceled mid-cycle in conjunction with other discounts that are valid during the same period. BRM applies all discount exclusion rules that are in effect at the time of this evaluation.

The process refunds all cycle fee discounts already charged from the point when the discount purchase or cancellation occurs. BRM then reapplies all remaining and any new cycle fee discounts from that point to the end of the cycle.

To accomplish this reevaluation, BRM divides a cycle (internally) into subcycles based on where discounts start and end. Discounts and exclusion rules are then applied for each subcycle. Both refunds and charges are determined based on the subcycles.

Although this process is run for all cycle fee discounts that are purchased or canceled mid-cycle, it can change the outcome only for sequential cycle fee discounts that are prorated. For general information about cycle fee proration, see "Calculating Prorated Cycle Fees" in *BRM Configuring and Running Billing*.

In essence, this process reevaluates past discounting activity in light of new information. This means that BRM always applies sequential discounts correctly over time, regardless of when they are purchased or canceled.

When this process is not enabled, BRM treats sequential cycle fee discounts as parallel discounts when they are purchased or canceled mid-cycle. You can use the BRM rerating process to adjust the outcome in that case.

This process has the following consequences and limitations:

- This process locks the account involved in the discount purchase or cancellation during the entire process or refunding and reapplying discounts.

  Locks are released for all members at the end of the transaction. The **propagate_discount** entry specifies when sharing begins. All members of the discount sharing group are locked, if the discount is shared and the **propagate_discount** entry is set to **1** in the Connection Manager (CM) **pin.conf** file. See "Configuring the Start and End Times for Discount Sharing" in *BRM Managing Accounts Receivable*.

- This process does not retroactively change resource balances.

  When a discount that grants a non-currency resource is purchased or canceled mid-cycle, calls rated based on the non-currency balance can be incorrect.

  For example, a cycle fee discount grants 100 free minutes and the subscriber uses all 100 minutes in the first week. The discount is then canceled mid-month. The discount amount is reevaluated, resulting in an adjusted grant of 50 free minutes. However, the minutes already used by the subscriber beyond the new grant amount are not recovered because the calls have already been rated. In this case, you can use BRM rerating to rerate the calls for the account.

- This process uses the discount exclusion rules that are in effect at the time of the evaluation.

  For example, if an exclusion rule changes on the 15th of the month and discounts are reevaluated on the 20th, the exclusion rule that took effect on the 15th is used when reevaluating discounts for the entire cycle.

Example of Recalculating Sequential Discounts in Mid-Cycle

In this example, an existing discount is applied to the monthly fee for June when the fee is charged. A second, lower priority discount is purchased mid-cycle.

These are the cycle fee and discount values used in this example:

- Monthly cycle fee for June: **$30**
- Discount D1 - existing

  Sequential; prorated; priority 2: **10%**
- Discount D2 - purchased June 11

  Sequential; prorated; priority 1: **20%**

Because the discounts are sequential and prorated, D1 is refunded from the point at which D2 becomes effective to the end of the cycle. BRM then handles the two discounts sequentially for the rest of the month.

Specifically, to handle D1 and D2 sequentially from June 11 on, BRM refunds the portion of D1 for June 11 through June 30 so that the portion of D1 that applies to the rest of the month can be calculated to form the basis for calculating D2. See Table 10–1.

Note: The formula for proration scale is (**number of days to prorate) / (number of days in cycle**). A 30-day cycle is used in this example.

Table 10–1 Recalculating Sequential DIscounts (mid-cycle)

Date	Activity	Calculation	Running Balance
June 1	Charge cycle fee less D1	30 * (1-10%)	27.00
June 11	Refund the portion of the discounted cycle fee that applies to June 11 through 30	30 * 20/30 * (1-10%) = 18.00	9.00
June 11	Charge the cycle fee for June 11 through 30, applying D1 and D2 sequentially	30 * 20/30 * (1-10%) * (1-20%) = 14.40	23.40
July 1	Charge cycle fee less D1 and D2, applied sequentially	30 * (1-10%)* (1-20%)	21.60

This process of refunding and charging is the same for discount changes in other scenarios, including cancellation, purchase and cancellation in the same cycle, long cycles, subscription changes, and plan changes (where the plans have different discounts).

There can be more than the two discounts used in this example.

Enabling Sequential Discounting of Cycle Fees

By default, sequential discounting of cycle fees is disabled in BRM. You can enable this process by modifying a field in the **billing** instance of the **/config/business_params** object.

- With sequential discounting of cycle fees *enabled*, BRM processes mid-cycle purchase and cancellation of sequential cycle fee discounts sequentially.
- With sequential discounting of cycle fees *disabled*, BRM processes mid-cycle purchase and cancellation of sequential cycle fee discounts as parallel discounts.

You modify the **/config/business_params** object by using the **pin_bus_params** utility.

To enable sequential discounting of cycle fees:

1. Use the following command to create an editable XML file from the **billing** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```

   This command creates the XML file named **bus_params_billing.xml.out** in your working directory. If you do not want this file in your working directory, specify the path as part of the file name.

2. Search the XML file for following line:

   ```
   <SequentialCycleDiscounting>disabled</SequentialCycleDiscounting>
   ```

3. Change **disabled** to **enabled**.

   Caution: BRM uses the XML in this file to overwrite the existing **billing** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM billing configuration.

4. Use the following command to load this change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_billing.xml
   ```

   You should run this command from the *BRM_Home*/**sys/data/config** directory, which includes support files used by the utility. *BRM_Home* is the directory where you installed BRM components. To run it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.

5. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   See "Using testnap" in *BRM Developer's Guide* for general instruction on using **testnap**. See "Reading Objects by Using Object Browser" in *BRM Developer's Guide* for information on how to use Object Browser.

6. Stop and restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

7. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see "pin_multidb" in *BRM System Administrator's Guide*.

About Applying Discounts Activated or Canceled in Mid-Cycle

Discounts are typically activated when they are purchased. They can also be purchased as inactive and activated later, or purchased with a deferred activation time.

When a discount is activated or canceled in the middle of an accounting cycle, BRM can prorate the discount, apply a full discount, or apply no discount for the accounting cycle in which it is activated or canceled. You specify the period in which the discount is applied by using Pricing Center to define discount validity rules.

When customers activate or cancel discounted products, BRM sets the discount start or end dates according to the discount validity rules. The discount is then granted for the period in which it is valid.

For example, if you specify that a discount activated in the middle of a cycle is granted for the entire cycle (full discount), the discount's start date is set to the first of the month and the discount is effective for the entire month in which it is activated.

You specify discount validity rules for cycle discounts (such as discounts on subscription fees) and for usage discounts (such as discounts on phone calls and text messaging).

Important: Discount validity rules apply to cycle events that are aligned to a monthly billing cycle only. If the billing cycle is not monthly, discount validity rules do not apply.

About Discount Validity Rules

You set discount validity rules for three scenarios:

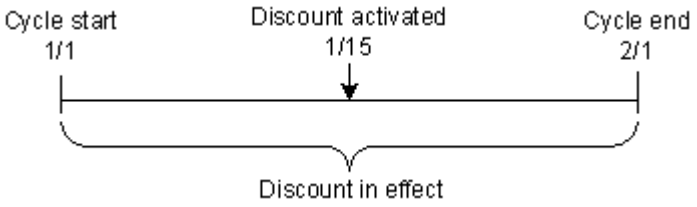
- Discounts that are activated in the middle of a cycle
- Discounts that are canceled in the middle of a cycle. The middle of a cycle is any time after the accounting cycle start time and before the accounting cycle end time.
- Discounts that are activated and canceled in the middle of the same cycle

For each scenario, you specify whether the discount is granted for the full cycle, prorated for part of the cycle, or not granted for the cycle.

You use Pricing Center to set the following discount validity rules:

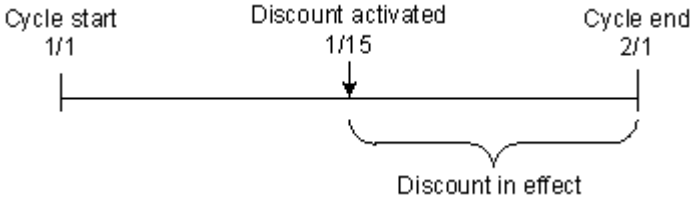
- **Valid from middle of cycle**: Applies to a discount that is activated at any time during an accounting cycle and is still owned at the end of the cycle:
  - **Full discount** sets the discount start time to the start of the accounting cycle. The discount is granted for the entire cycle in which it is activated as shown in Figure 10–13.

Figure 10–13 Valid from Middle of Cycle - Full Discount



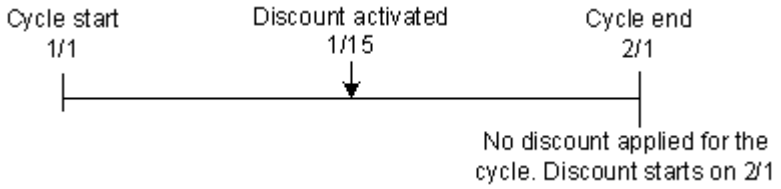
  - **Prorated discount** sets the discount start time to the purchase time. The discount is granted for the period it is valid during the cycle in which it is activated as shown in Figure 10–14.

Figure 10–14 Valid from Middle of Cycle - Prorated Discount



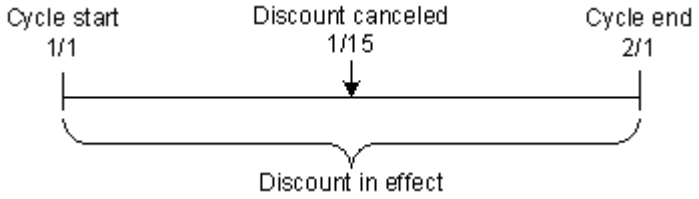
– **No discount** sets the discount start time to the beginning of the next accounting cycle. The discount is not granted during the cycle in which it is activated as shown in Figure 10–15.

Figure 10–15 Valid from Middle of Cycle - No Discount



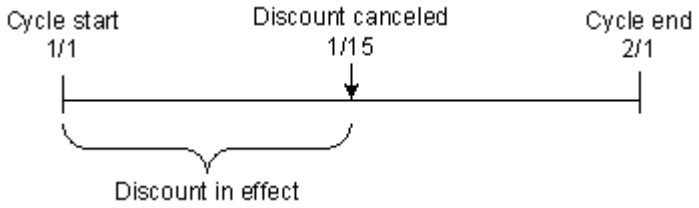
- **Valid to middle of cycle**: Applies to a discount owned at the beginning of an accounting cycle and canceled at any time during the cycle:
  – **Full discount** sets the discount end time to the end of the accounting cycle in which it is canceled. The discount is granted for the entire cycle in which it is canceled as shown in Figure 10–16.

Figure 10–16 Valid to Middle of Cycle - Full Discount



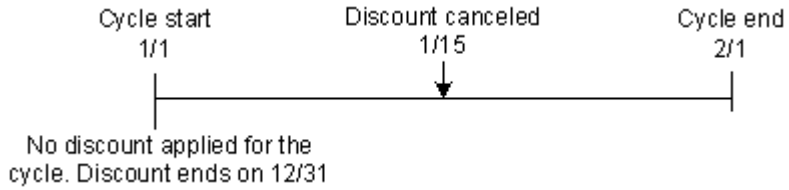
  – **Prorated discount** sets the discount end time to the cancellation time. The discount is granted for the period it is valid during the cycle in which it is canceled as shown in Figure 10–17.

Figure 10–17 Valid to Middle of Cycle - Prorated Discount



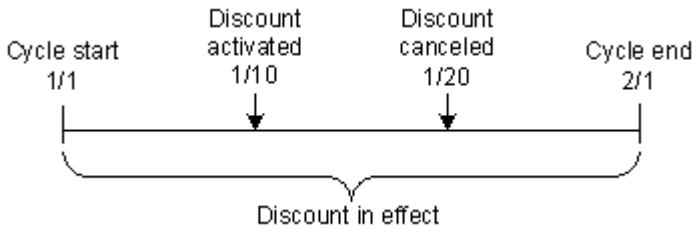
  – **No discount** sets the discount end time to the end of the previous accounting cycle. The discount is not granted during the cycle in which it is canceled as shown in Figure 10–18.

Figure 10–18 Valid to Middle of Cycle - No Discount



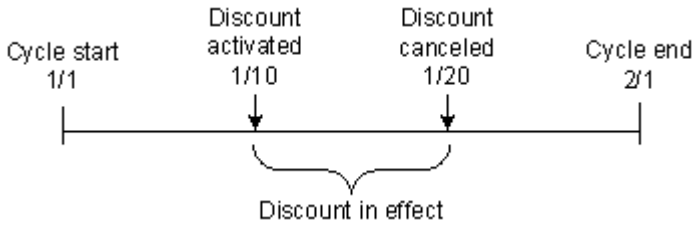
- **Valid only part of the cycle**: Applies to a discount that is both activated and canceled in the same accounting cycle:
  - **Full discount** sets the discount start time to the start of the accounting cycle and sets the discount end time to the end of the accounting cycle. The discount is granted for the entire cycle in which it is activated and canceled as shown in Figure 10–19.

Figure 10–19 Valid Only Part of the Cycle - Full Discount



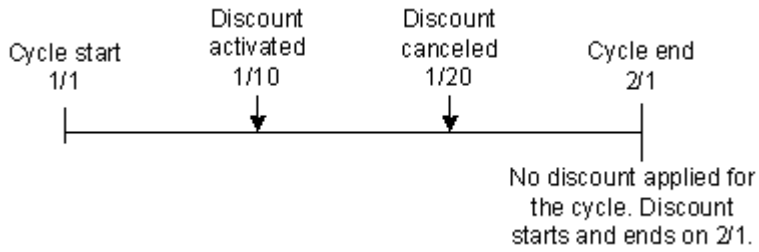
  - **Prorated discount** sets the discount start time to the activation time and sets the discount end time to the cancellation time. The discount is granted only for the period it is valid as shown in Figure 10–20.

Figure 10–20 Valid Only Part of the Cycle - Prorated Discount



  - **No discount** sets both the discount start time and end time to the beginning of the next accounting cycle. The discount is not granted during the cycle in which it is activated and canceled as shown in Figure 10–21.

Figure 10–21 Valid Only Part of the Cycle - No Discount



Discount validity rules are stored in the **/discount** object. When a discount is activated, its start and end times are set in the **/purchased_discount** object associated with the customer's account.

You set discount validity rules by using Pricing Center. To implement discount validity rules, you must also configure the batch rating pipeline if you use one, and for certain validity rules, run rerating to apply the discounts and run a utility to change the status of expired discounts.

For more information about implementing discount validity rules, see "Implementing Discount Validity Rules".

Immediate Cancellation of Discounts Canceled in Mid-Cycle

When a discount is canceled in the middle of an accounting cycle and the proration for the discount is set to **Full discount**, you can configure BRM to set the discount end date to the cancellation date, thereby cancelling the discount immediately.

Use the **CancelFullDiscountImmediate** business parameter to specify how BRM handles a discount canceled in the middle of a billing or accounting cycle when the proration for the discount is set to full discount. **CancelFullDiscountImmediate** is located in the **subscription** instance of **/config/business_params** object and takes one of the following values:

- **disabled**. The default value. When **CancelFullDiscountImmediate** is **disabled** and a discount is canceled in the middle of an accounting cycle, BRM sets the discount end date to the end date of the accounting cycle.
- **enabled**. When **CancelFullDiscountImmediate** is **disabled** and a discount is canceled in the middle of an accounting cycle, then if the proration for the discount is set to full discount, BRM cancels the discount immediately. It sets the discount end date to the cancellation date.

You modify the **/config/business_params** object by using the **pin_bus_params** utility. For more information on this utility, see "pin_bus_params" in *BRM Developer's Guide*.

To immediately cancel discounts canceled in mid-cycle:

1. Use the following command to create an editable XML file from the **subscription** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsSubscription bus_params_subscription.xml
   ```

   This command creates the XML file named **bus_params_subscription.xml.out** in your working directory. If you do not want this file in your working directory, specify the path as part of the file name.

2. Search the XML file for following line:

   ```
   <CancelFullDiscountImmediate>disabled</CancelFullDiscountImmediate>
   ```

3. Change **disabled** to **enabled**.

   Caution: BRM uses the XML in this file to overwrite the existing **subscription** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM subscription configurations.

4. Save this file and rename it as **bus_params_subscription.xml**.

5. Use the following command to load this change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_subscription.xml
   ```

   You should run this command from the *BRM_Home*/**sys/data/config** directory, which includes support files used by the utility. To run it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.

6. Read the object with the **testnap** utility or Object Browser to verify that all fields are correct.

   The resulting flist of the **testnap** utility must resemble the following example, with the PIN_FLD_PARAM_VALUE field value set to **1**:

```
0 PIN_FLD_POID POID [0] 0.0.0.1 /config/business_params 9806 0
0 PIN_FLD_ACCOUNT_OBJ POID [0] 0.0.0.1 /account 1 0
0 PIN_FLD_DESCR STR [0] "Business logic parameters for Subscription"
0 PIN_FLD_HOSTNAME STR [0] "-"
0 PIN_FLD_PARAMS ARRAY [7] allocated 4, used 4
1 PIN_FLD_DESCR STR [0] "Parameter to enable or disable cancelling a prorated
full discount in the middle of a billing cycle. 1 means enabled."
1 PIN_FLD_PARAM_NAME STR [0] "cancel_full_discount_immediate"
1 PIN_FLD_PARAM_TYPE INT [0] 1
1 PIN_FLD_PARAM_VALUE STR [0] "1"
```

   See "Using testnap" in *BRM Developer's Guide* for general instruction on using **testnap**. See "Reading Objects by Using Object Browser" in *BRM Developer's Guide* for information on how to use Object Browser.

7. Stop and restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

About Discount Exclusion Rules

Exclusion rules establish a mutually exclusive relationship between discounts or between a discount and a plan. When a mutually exclusive relationship exists between two discounts, only one of the discounts can be applied. When such a relationship exists between a discount and a plan, the discount is not applied.

After you configure two discounts as mutually exclusive, at run time BRM does not allow a deal to be committed when it has both of those discounts in the deal.

Exclusion rules between discounts can be applied at billing time or when cycle fees are applied. Exclusion rules between discounts and plans can be applied at purchase time or at billing time, but not when cycle fees are applied.

There are two types of discount exclusion rules:

- Discount exclusion rules that apply at billing time and govern purchases, such as shared discount groups
- Discount exclusion rules that are determined by ownership and govern discount application

Discounts can be combined with price plans and can belong to multiple price plans. If you define an exclusion rule between a plan and a discount, a customer who owns that pricing plan cannot purchase that excluded discount.

The following sections describe:

- About Exclusion Rules between Discounts
- About Exclusion Rules between Discounts and Plans
- About Cycle Fees and Discount Exclusions
- About Billing-Time Discount Exclusions

For information on configuring discount exclusion rules, see "Setting Up Discount Exclusion Rules".

How Exclusion Rules Are Evaluated

During discount evaluation, *all* discounts (system, billing, shared, usage, and cycle discounts) are retrieved and checked for exclusion rules. The discounts that are not excluded are then checked for the event type associated with the discount, such as the billing-time event or cycle event. Discounting then processes the discounts associated with the event that occurred.

For example, an account has three cycle discounts (**CD1**, **CD2**, and **CD3**) and a usage discount (**UD**). Exclusion rules are set between:

- **CD2** and **CD1** (**CD1** is applied)
- **CD1** and **UD** (**UD** is applied)

When cycle events are processed, all discounts are retrieved and checked for exclusion rules. At that time, **CD2** and **CD1** are eliminated. **UD** is eliminated as well because it is not a cycle discount. Only **CD3** is applied for cycle events.

When usage events are processed, all discounts are retrieved and checked for exclusion rules. **CD2**, which is mutually exclusive with **CD1**, is eliminated. **CD1**, which is mutually exclusive with **UD**, is eliminated. **CD3**, which is not a usage discount, is eliminated. Only **UD** is applied for usage events.

About Exclusion Rules between Discounts

You can set up exclusion relationships between system, shared, or subscription discounts.

- A *system discount* is granted automatically to all accounts. This type of discount is always applied, except when specifically excluded by an exclusion rule.
- A *shared discount* is shared by members of an account or service group. Any exclusion rule that restricts a shared discount for use when another discount is active denies that discount to that service or account. Conversely, all members of a discount group are granted the discount as long as it is not restricted by an exclusion rule.
- A *subscription discount* is purchased by a customer as a separate discount or as part of a plan.

About Exclusion Rules between Discounts and Plans

If an exclusion relationship exists between a discount and a plan, a customer can own the discount or the plan, but not both. Further, the customer cannot own any discounts associated with the plan if the customer owns the excluded discount.

About Cycle Fees and Discount Exclusions

When discounts are applied to cycle fees and the **ValidateDiscountDependency** flag is set in the **/config/business_params** object, all discounts are filtered based on the cycle event type and then checked for discount exclusions stored in the **/dependency** object. Discounts that are filtered and checked include:

- All system discounts for the service type.
- All discounts for the subscription service.
- All shared discounts (if the discount is a member of a sharing group).
- All non-billing-time subscription discounts for the service instance.

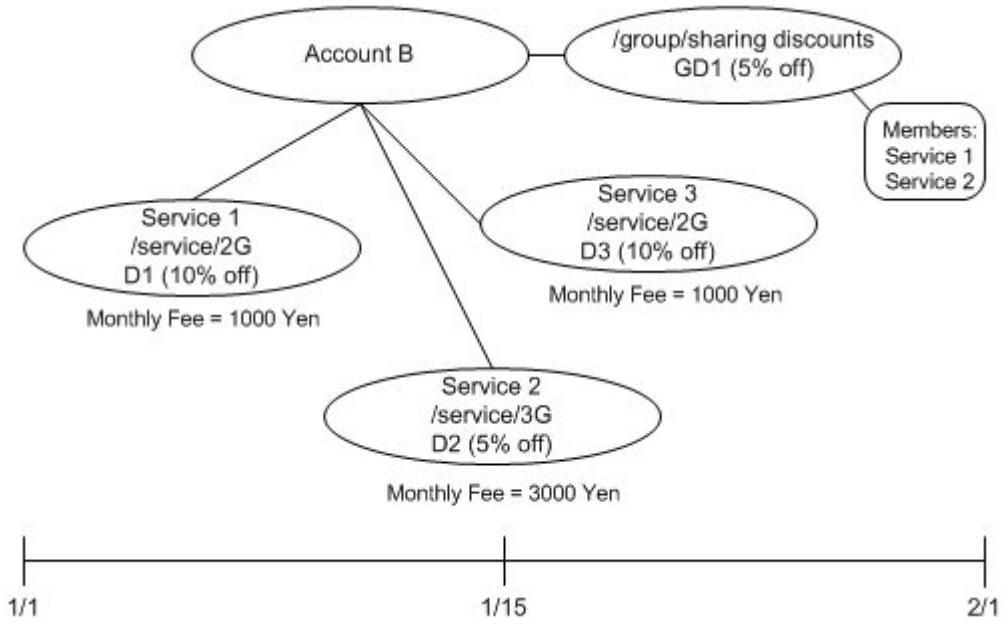
Applicable discounts are calculated, and a final discount list is created for processing.

Example of Discount Exclusion with Cycle Fees

In the example illustrated in Figure 10–22, Account B has three subscription services: Service 1 and Service 2 are part of the discount share group GD1, and Service 3 is not part of the share group. There are four discounts:

- A 10% subscription discount for cycle fees (D1)
- A 5% system discount (SD1) that is defined as mutually exclusive with D1
- A 5% subscription discount (D2)
- A 5% shared discount for cycle fees (GD1)

Figure 10–22 Discount Exclusion with Cycle Fees Example



At billing time, the following charges are owed by Account B:

- Service 1 is billed 850 Yen. Two discounts are applied: a 10% discount for D1 and a 5% group discount because Service 1 is a member of the GD1 shared discount group. The system discount SD1 is not applied because it is mutually exclusive with D1.
- Service 2 is billed 2550 Yen. Three discounts are applied: a 5% discount for D2, a 5% group discount because Service 2 is a member of the GD1 shared discount group, and a 5% discount for SD1. (SD1 is not mutually exclusive with D2 so can be applied.)
- Service 3 is billed 900 Yen. A 10% discount is applied for D3. No other discount is applied because Service 3 is not a member of the GD1 shared discount group.

Example of Plan-Discount Exclusion with Cycle Fees

The following example of discount exclusions between plans and discounts where the resulting valid discount is applied for cycle fees. In these examples, the automatic system discounts **System_Discount1** and **System_Discount2** apply to cycle forward, cycle arrears, and cycle forward arrears events.

When both discount-to-discount and plan-to-discount exclusions are enabled (see "Configuring Exclusion Rules"), the system validates the discount-to-discount exclusions first, followed by the discount-to-plan exclusions.

The following exclusion rules in Table 10–2 are used:

Table 10–2 Exclusion Rules

Exclusion Rule Set Between	Result
System_Discount1 and **Plan2**	**System_Discount1** is not applied.
System_Discount2 and **Plan1**	**System_Discount2** is not applied.
System_Discount2 and **Subscription_Discount2**	**System_Discount2** is not applied.

The monthly fees for each plan are:

- **Plan1** = 3000 Yen
- **Plan2** = 4000 Yen

Each plan includes two discounts:

- **System_Discount1** (automatic system discount) = 10%
- **System_Discount2** (automatic system discount) = 40%

These additional discounts are purchased separately:

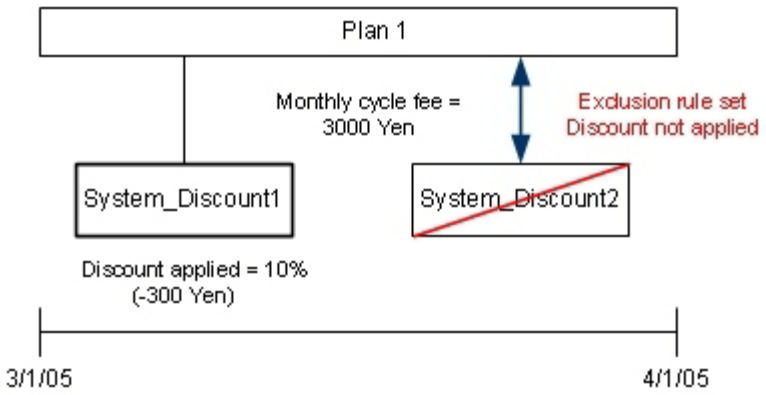
- **Subscription_Discount1** (subscription discount) = 20%
- **Subscription_Discount2** (subscription discount) = 30%

In this example, an account purchases **Plan1** at the beginning of the month. **Plan1** has two automatic system discounts (**System_Discount1** and **System_Discount2**) and a mutually exclusive relationship with **System_Discount2** as shown in Figure 10–23.

When the discounts are calculated, the mutual exclusion between **Plan1** and **System_Discount2** eliminates **System_Discount2**. **System_Discount1** has no exclusion restriction and is applied.

The monthly cycle fees for **Plan1** are 3000 Yen. **System_Discount1** provides a 10% discount of 300 Yen, resulting in an adjusted cycle fee of 2700 Yen.

Figure 10–23 Plan-Discount Exclusion with Cycle Fees Example



About Billing-Time Discount Exclusions

When billing-time discounts are applied and the **ValidateDiscountDependency** flag is set in the **/config/business_params** object, discount exclusions are performed based on the flag setting. Values for the **ValidateDiscountDependency** flag can be any combination of the following:

- **0x00**: No discount dependency validation (this is the default value).
- PIN_DISC_VALIDATE_DISC_DISC_DEP (**0x01**): Validate exclusions between discounts.
- PIN_DISC_VALIDATE_PLAN_DISC_DEP (**0x02**): Validate exclusions between discounts and plans.
- PIN_DISC_VALIDATE_DISABLE_PURCH_TIME (**0x04**): Disable validations between plans and discount system wide. When this flag is set, no dependency validations occur at purchase time.

When specifying these values, you enter the sum of any of the parameter values shown above. For example, to enable exclusion rules between discounts **(0x01)** and between a discount and a plan **(0x02)**, enter **(0x03)**. Or, to enable exclusions between a discount and a plan only, but to disable exclusions at purchase time, enter **(0x06)**.

Discounts that are filtered and checked include subscription, system, and shared billing-time discounts. System and subscription discounts are sorted and paired, and each pair is checked to see if an exclusion rule is set between them. If an exclusion rule is present, the *dependent* discount is dropped and the *dependee* discount is retained and added to a list that is sent to the discount pipeline for processing. For more information about dependent and dependee discount objects, see "Managing /dependency Objects" in *BRM Setting Up Pricing and Rating*.

Discount-to-discount exclusions are processed first, then discount-to-plan exclusions, if any.

Discount-to-plan exclusions are handled in much the same way as discount-to-discount exclusions. Discounts and plans owned by an account are compared for exclusion rules. Discounts for any plan/discount combinations with exclusion rules set are not applied.

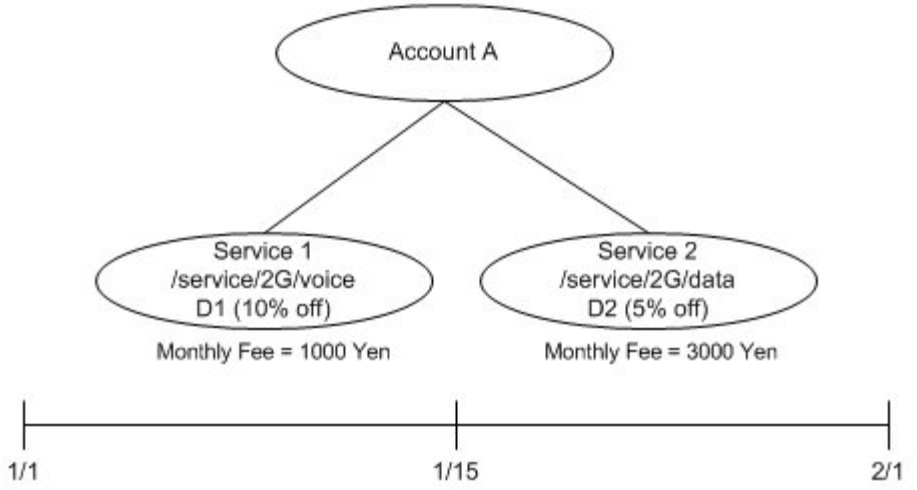
The exclusion rules are applied for each service level or account level.

Examples of Exclusion Rules Applied at Billing Time

In this example, Account A has 2 subscription services (Service 1 and Service 2) and three discounts as illustrated in Figure 10–24:

- A 10% subscription discount for cycle fees (D1)
- A 5% system discount (SD1) that is defined as mutually exclusive with D1
- A 5% subscription discount (D2)

Figure 10–24 Exclusion Rules Applied at Billing Time Example 1



At billing time, the following charges are owed by Account A:

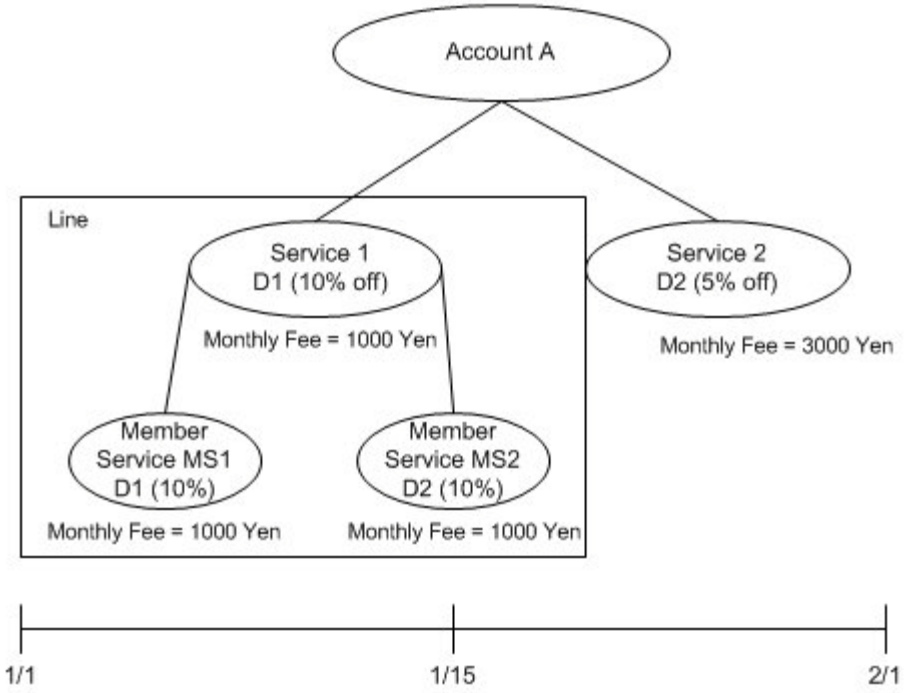
- Service 1 is billed 900 Yen after the 10% D1 discount is applied. The SD1 system discount is not applied because it is mutually exclusive with D1.
- Service 2 is billed 2700 Yen after two discounts are applied: the 5% D2 discount and the 5% SD1 system discount.

In this example, Account A has the following discounts as illustrated in Figure 10–25:

- A 10% subscription discount for cycle fees (D1)
- A 5% subscription discount (D2)
- A 5% system discount (SD1) that is defined as mutually exclusive with D2

Service 1 has member services with their own balance groups MS1 and MS2.

Figure 10–25 Exclusion Rules Applied at Billing Time Example 2



At billing time, the following charges are owed by Account A:

- Service 1 is billed 850 Yen. Two discounts are applied: a 10% discount for D1 and a 5% system discount for SD1.
- MS1 is billed 750 Yen. Three discounts are applied: a 10% discount for D1, a 5% system discount for SD1, and a 10% discount inherited from Service 1.
- MS2 is billed 800 Yen. Two discounts are applied: a 10% discount for D2 and another 10% discount inherited from Service 1. The system discount SD1 is excluded.
- Service 2 is billed 2850 Yen. One discount is granted: a 5% discount off the monthly fee. The system discount SD1 is excluded.

For more information, see "Setting Up Discount Exclusion Rules".

About Exclusion Rules for Usage Discounts

You can apply exclusion rules between a usage discount and any of the following discounts:

- Billing-time discounts
- Cycle discounts
- System discounts
- Other usage discounts

Usage discounts are applied when subscribers use their services; for example, by making phone calls.

Setting up exclusion rules between a usage discount and other discounts or plans is useful when you want to prevent additional discounting, including aggregation discounting, for an excluded billing-time discount. For example, if you purchase two

cycle discounts (a 5% discount and a 10% discount) and set up an exclusion rule between them, one is eliminated. For more information, see "About Exclusions Between Usage Discounts and Billing-Time Discounts".

Usage discount exclusion can occur in both the real-time and batch pipeline.

About Exclusions Between Usage Discounts and Billing-Time Discounts

Billing-time discounts are granted based on balances accumulated by an aggregation discount. An *aggregation discount* is a discount that increments a counter resource balance when usage occurs.

Setting up an exclusion rule between a billing-time discount and another discount or a plan does not prevent aggregation. Because you want to prevent aggregation when the billing-time discount is excluded, you must also set up an exclusion rule between the aggregation discount associated with the billing-time discount and the other discount:

- If the billing-time discount is excluded by a plan, set up an exclusion rule between that plan and the aggregation discount.
- If billing-time discount 1 is excluded by billing-time discount 2, set up an exclusion rule between the aggregation discount and the billing-time discount 2.
- If the billing-time discount is excluded by a cycle fee discount, set up an exclusion rule between the aggregation discount and the cycle fee discount.

If a billing-time discount is owned for part of the month only, during the time the discount is owned, discount information added to the EDR and the discount is applied.

When a billing-time discount is owned for part of the month, the associated usage aggregation discount aggregates usage only during the period that the billing-time discount is owned. Therefore, the billing-time discount is based on aggregated usage for part of the month only.

For more information on billing-time and aggregation discounts, see the following topics:

- Setting Up Billing-Time Discounts
- Creating a Real-Time Aggregation Discount

Example of Usage Discount Exclusion Rules

Discount exclusion rules can affect how usage is aggregated for an account that owns several subscription services (lines) and a billing-time discount, and the billing-time discount is excluded by discounts or plans associated with the subscription services.

In this example, an account owns:

- One billing-time discount (**BTD**). This discount is granted based on the aggregated usage of all subscription services in the account.
- One account-level aggregation discount: a shared discount owned by the object owner (**AGD**). This discount is shared with each subscription service and aggregates the usage for all subscription services.
- An account-level resource balance (**ARB**). This balance stores the aggregated usage amount for all subscription services.

The following plans and discounts are associated with the subscription services:

- Three plans (**Plan1**, **Plan2**, and **Plan3G**).
- Two call-by-call (usage) discounts (**CBC1** and **CBC2**).

- A subscription service-level resource balance (**LRB**). This balance stores the individual subscription's monthly usage amount.

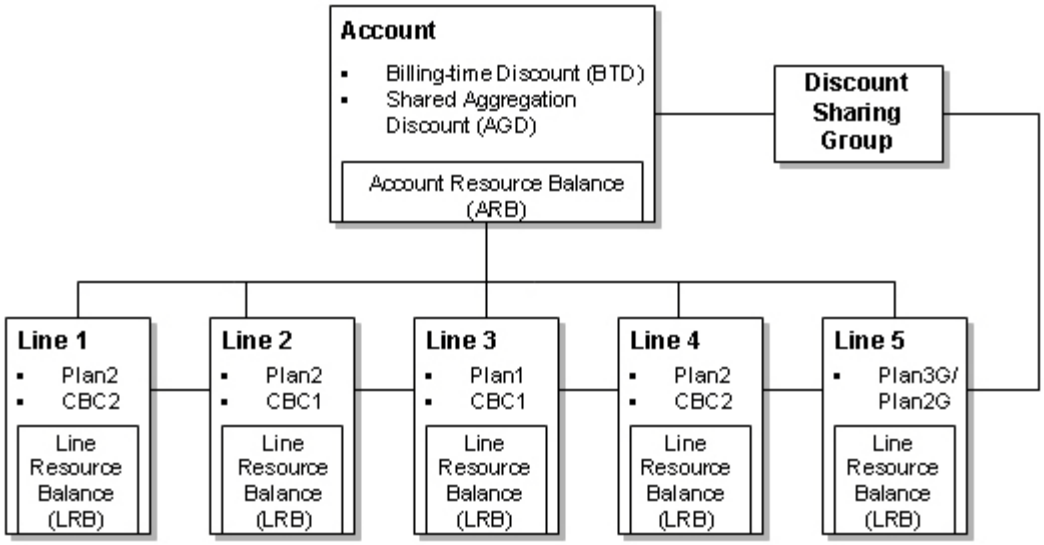
When billing is run for the account at the end of the month, BRM does the following:

- Determines for each subscription service whether the billing-time discount applies.
- Applies a percent discount based on the aggregated **ARB** for total minutes on all subscription services in the account.

> **Note:** In this example, there is only one account-level billing-time discount. If another billing-time discount was owned by each subscription service, an additional percent discount would be applied for individual subscription service activity during the month, based on **LRB**.

Figure 10–26 shows the account structure, including the subscription services (the lines), discounts, and resource balances:

Figure 10–26 Usage Discount Exclusion Rules Example



Exclusion rules are set between the following discounts and plans:

- **BTD** and **Plan1** (**BTD** is not applied)
- **AGD** and **Plan1** (**AGD** is not applied)
- **BTD** and **CBC1** (**BTD** is not applied)
- **AGD** and **CBC1** (**AGD** is not applied)
- **BTD** and **Plan3G** (**BTD** is not applied)
- **AGD** and **Plan3G** (**AGD** is not applied)

When these exclusion relationships are present and a subscription service owns any of the excluded discounts or plans, the aggregation discount is not applied and the account's aggregation balance is not incremented.

For more information, see "Setting Up Discount Exclusion Rules".

Table 10–3 shows the monthly subscription service activity for the account and each subscription service. It demonstrates, based on the subscription service activity, how exclusion rules and discounts are applied and how resource balances are incremented.

Table 10–3 Monthly Subscription Service Activity

Line	Monthly Subscription Service Activity	Usage Discounts Applied	Resource Balance Activity
Line1	▪ Owns **Plan2** and **CBC2**. ▪ Total minutes called: **5500**.	▪ No exclusion rules apply. ▪ The aggregation discount is applied. ▪ The **CBC2** discount is applied.	▪ The **ARB** increases from zero to 5500 minutes. ▪ The line resource balance (**LRB**) increases from zero to 5500 minutes.
Line 2	▪ Owns **Plan2** and **CBC1**. ▪ Total minutes called: 2300.	▪ The aggregation discount is not applied because it is mutually exclusive with **CBC1**. ▪ The **CBC1** discount is applied.	▪ The **ARB** is not incremented. ▪ The **LRB** is increased to 2300 minutes.
Line 3	▪ Owns **Plan1** and **CBC1**. ▪ Total minutes called: 4000.	▪ The aggregation discount is not applied because it is mutually exclusive with both **Plan1** and **CBC1**. ▪ The **CBC1** discount is applied.	▪ The **ARB** is not incremented. ▪ The **LRB** is increased to 4000 minutes
Line 4	▪ Owns **Plan2** and **CBC2**. ▪ Total minutes called: 4500.	▪ No exclusion rules apply. ▪ The aggregation discount is applied. ▪ The **CBC2** discount is applied.	▪ The **ARB** is incremented from 5500 to 10,000 minutes. ▪ The **LRB** (line resource balance) increases from zero to 4500 minutes.
Line 5	▪ Owns **Plan3G**. ▪ Minutes for the first half of the month: 1900. ▪ Changes from **Plan3G** to **Plan2G** mid-month. ▪ Minutes for the second half of the month: 2200.	For the first half of the month: ▪ Because it is mutually exclusive with **Plan3G**, the aggregation discount is not applied. For the second half of the month: ▪ No exclusion rule applies between the aggregation discount and **Plan2G**, so the aggregation discount is applied.	For the first half of the month: ▪ The ARB is not incremented. ▪ The LRB increases from zero to 1900 minutes. For the second half of the month: ▪ The **ARB** is incremented from 10,000 to 12,200 minutes. ▪ The **LRB** increases from 1900 to 4100 minutes.

In Table 10–3, the billing-time discount is applied at the following rates listed in Table 10–4:

Table 10–4 Discount Rates

Total Minutes Consumed by all Subscription Services	Discount Rate On Charges
1 – 5000	10%
5001 – 10000	20%
10001 – maximum	30%

At the end of the cycle, the total usage for all subscription services is 20,400 minutes. However, because the aggregation discount was excluded by other discounts and plans, the aggregation balance for all subscription service usage is 12,200 minutes. When billing is run, the billing-time discount is based on 12,000 minutes of usage, and a 30% discount is applied to the balance due for the account.

About Volume-Based Discounts

Volume-based discounts are granted based on threshold values that define levels of usage such as the number of calls made, the amount of money a subscriber spends on calls, the number of subscriptions purchased, and the number of days a user has subscribed to a service.

Important: Support for some volume-based discounts is included in Advanced Discounting Manager, an optional feature that requires a separate license.

You can set up several types of volume-based discounts. For information, see the following topics:

- About Discounts Based on Number of Subscriptions
- About Discounts Based on the Number of Contract Days
- About Discounts Based on Monthly Fees and Usage

About Discounts Based on Number of Subscriptions

You can grant discounts based on the number of active subscriptions in an account group. This type of discount is designed to be granted to corporate accounts. For example, you might offer a percentage off the monthly fee when the number of subscriptions exceeds certain limits.

The plans purchased by accounts in the account group must include a subscription service. The number-of-subscriptions discounts is granted at the end of each billing cycle and is based on the number of subscription services that are active at the end of the cycle.

BRM does not consider subscription services that were canceled or transferred out of the account group during the cycle. For example, an account group includes six subscription services at the beginning of a cycle. During the cycle, four subscription services are added and two subscription services are removed. At the end of that cycle, the discount is based on eight active subscription services (6+4-2=8).

An account's subscription service is not counted when the account owns a plan or discount that is mutually exclusive with the number-of-subscriptions discount. You set up exclusions between discounts and other discounts or plans by using discount exclusion rules. For information, see "About Excluding Subscriptions when Discount

Exclusion Rules Apply".

To offer discounts based on a number of subscriptions, you set up a hierarchical account group and a discount sharing group. For more information, see "About Setting Up Discounts Based on Number of Subscriptions".

About Resources that Track the Number of Subscription Services

The number of active subscription services in an account group is stored in the Line Counter resource balance. Line Counter is a type of aggregation counter resource. When the account group's parent account purchases a plan that includes a Line Counter, that resource is added to the parent account's balance group. Discounting can then track the number of subscription services associated with the account group in the parent account's balance.

When you set up the number-of-subscriptions discount, you base the discount on the value in the parent account's Line Counter balance. The Line Counter is updated when billing is run.

About Excluding Subscriptions when Discount Exclusion Rules Apply

BRM does not count a subscription service as an active subscription when you set up a discount exclusion rule between the number-of-subscriptions discount and a plan (or discount in the plan) that includes the subscription service.

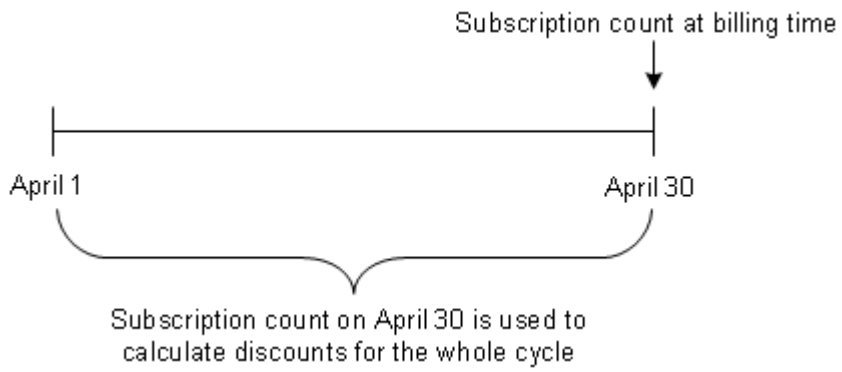
> **Important:** To exclude subscription services from being counted, you must enable and configure discount exclusion rules. See "About Discount Exclusion Rules".

An account can own other plans and discounts that are affected by discount exclusion rules and still have its subscription service counted, providing these exclusion rules do not exclude the number-of-subscriptions discount.

How Subscriptions Are Counted

BRM counts the number of subscription services at billing time, before discounts are calculated, as shown in Figure 10–27. All active subscription services in the account group are counted except for those excluded by discount exclusion rules.

Figure 10–27 Subscription Count at Billing Time



Subscription services that are added to the account group during the cycle are counted at the end of the cycle, providing they are active. Subscription services that are

removed from the account group during the cycle are not counted at the end of the cycle.

The status of exclusion rules and the discounts or plans associated with exclusion rule is considered only at the time subscription services are counted. If the status is changed after the end of the billing cycle, but before billing is run, the new status is not considered.

Updating the Count of Subscriptions

The count of subscription services is updated by the PCM_OP_SUBSCRIPTION_COUNT_LINES opcode. This opcode is called by the billing event when billing is run. It counts the active subscription services in the account group, updates the Line Counter in the parent account, and then generates an **/event/billing/lcupdate** event to record the update.

PCM_OP_SUBSCRIPTION_COUNT_LINES performs the following tasks:

1. Determines the identity of the parent account in the account hierarchy.
2. Checks that the parent account's balance group includes the Line Counter resource.
3. Checks each child account to determine the following:
   - How many subscription services are associated with the child account and whether the subscription services are active.
   - For each active subscription service, what exclusion rules exist between the subscription service's active plans and discounts, including shared discounts. If exclusion rules exist for the subscription service at the end of the cycle, the Line Counter is not incremented for that subscription service.
4. Updates the value of the Line Counter in the parent account's balance group.

You can customize the criteria for counting subscription services by implementing the PCM_OP_SUBSCRIPTION_POL_COUNT_LINES policy opcode.

About Counting Subscriptions During Rerating

The Line Counter is updated with the count of active subscription services at the end of each billing cycle. This count determines the discount amount applied by the number-of-subscriptions discount. If rerating is requested for a date occurring before the start of the current billing cycle:

- The number of subscription services is not recounted; instead, the count from the *beginning* of the cycle is used to apply discounts.
- Exclusion rules are re-evaluated so that the number-of-subscriptions discount is not applied if it is excluded.

About Setting Up Discounts Based on Number of Subscriptions

To offer discounts based on a number of subscriptions, you set up the following components:

- **The number-of-subscriptions discount**. How you set up this discount depends on what kind of discount you will offer. For example, this can be a billing-time discount applied to service usage fees or a cycle fee discount applied only to cycle fees. For information about billing-time discounts, see "About Billing-Time Discounts".

- **Plans that include subscription services** that will be purchased by the child accounts in an account group. A subscription service represents the customer's subscription. You set up subscription services by using Pricing Center.

  For information about subscription services, see "Managing Customers' Subscription-Level Services" in *BRM Managing Customers*.

- **Plans that include the Line Counter resource and the number-of-subscriptions discount** that will be purchased by the parent account in an account group. The Line Counter is used to track the number of active subscription services. You create plans by using Pricing Center.

- **An account hierarchy**. This hierarchy is needed because BRM counts the number of subscription services in the account group to determine the number of active subscriptions. You create account hierarchies by using Customer Center.

  For information about account hierarchies, see "About Hierarchical Account Groups" in *BRM Managing Accounts Receivable*.

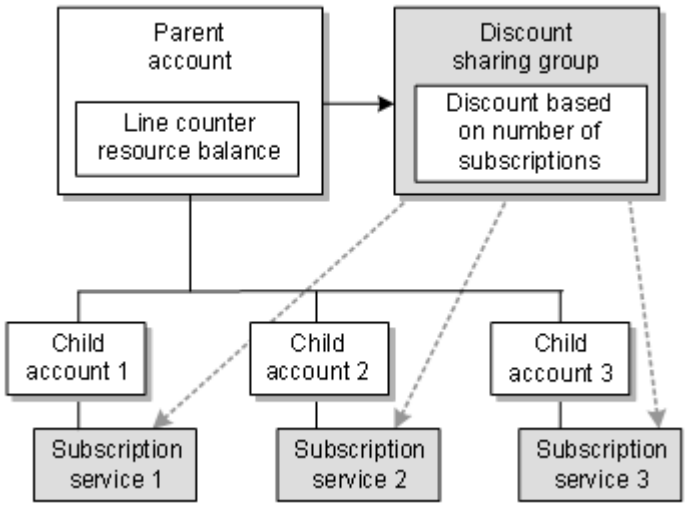
- **A discount sharing group** that is owned by the parent account and has the subscription services in the child accounts as members. You create a discount sharing group to share the number-of-subscriptions discount; for example, to grant a discount on the service fees for each child account based on the total number of subscription services.

For information about sharing discounts, see "About Shared Discounts".

For more information about setting up discounts based on number of subscriptions, see "Setting Up Discounts Based on Number of Subscriptions".

Figure 10–28 shows the hierarchical relationship and components required to apply discounts based on number of subscriptions:

Figure 10–28 Hierarchy of Discounts Based on Number of Subscriptions



Example of Applying a Discount Based on the Number of Subscriptions

The following example shows how a number-of-subscriptions discount is applied.

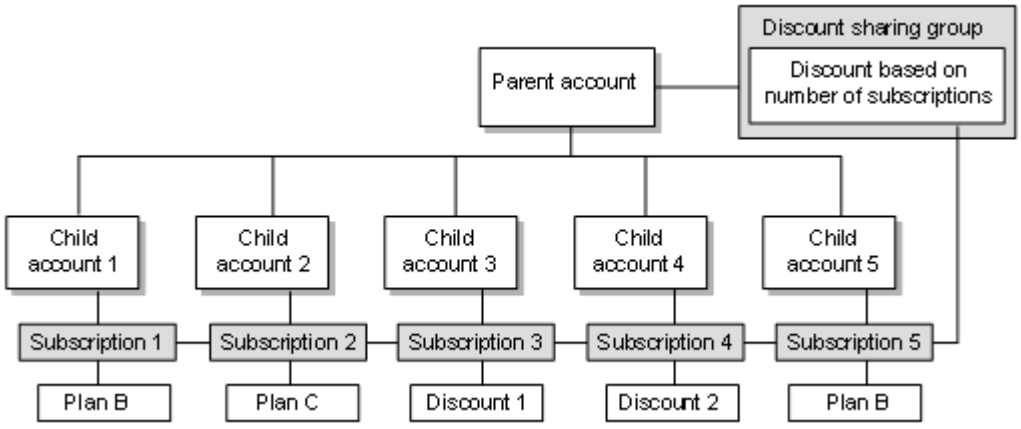
In this example:

- Each child account includes one subscription service. The subscription services are members of the discount sharing group that shares the number-of-subscriptions discount.
- The discount granted by the number-of-subscriptions discount is a percentage off the cycle fees:
  - 0 – 1 Subscription services: 0%
  - 2 – 3 Subscription services: 10%
  - 4 – 5 Subscription services: 15%
- Each child account's subscription service is associated with one of the following discounts or plans:
  - Plan B
  - Plan C
  - Discount 1
  - Discount 2
- Exclusion rules exist between the number-of-subscriptions discount and the following plan and discount:
  - Plan C
  - Discount 1

  Any subscription service associated with Plan C or Discount 1 at the end of the cycle, before billing is run, is not counted.

Of the subscription services shown in Figure 10–29, **Subscription 1**, **Subscription 4**, and **Subscription 5** are included in the subscription count at the end of the cycle. **Subscription 2** and **Subscription 3** are excluded because they are associated with a discount or plan that is mutually exclusive with the number-of-subscriptions discount.

Figure 10–29 Number of Subscriptions and Mutually Exclusive Discounts at Cycle End



In Table 10–29, based on the exclusion rules, BRM applies discounts as follows:

- **Subscription 1** receives the number-of-subscriptions discount.
- **Subscription 2** receives no discount because the number-of-subscriptions discount is excluded by Plan C.

- **Subscription 3** receives only Discount 1 because the number-of-subscriptions discount is excluded by Discount 1.
- **Subscription 4** receives Discount 2 and the number-of-subscriptions discount.
- **Subscription 5** receives the number-of-subscriptions discount.

The number-of-subscriptions discount grants a percentage of 10% off to each eligible account because three subscription services are included in the count at the end of the cycle.

About Discounts Based on the Number of Contract Days

Important: Support for discounts based on the number of contract days is included in Advanced Discounting Manager, an optional feature that requires a separate license.

BRM provides support for offering discounts based on the number of *active* days a customer has been under contract. This support enables you to track the number of days a customer's subscription service has been active from the date of enrollment and to provide discounts based on this value.

BRM calculates the active contract days as follows:

Contract days = billing date - enrollment date - days of suspension

BRM provides two aggregation counters that you use when setting up discounts based on contract days:

- **Contract Days Counter (CDC)** stores the total number of contract days.
- **Contract Days Counter for Discount (CDCD)** stores an account's aggregated usage fees for the month.

BRM updates the count of contract days when:

- Subscription services are created for a new account or added for an existing account.

  When a subscription service is created, the CDC is equal to the number of days between the subscription service creation date and the next billing cycle start date. If a subscription service is created in an inactive state, the CDC has a value of zero (0) until the subscription service is activated.

- Billing starts for the subscription service.

  When billing begins, the contract days counter is incremented by the number of days in the billing cycle.

- There is a change in the status of the subscription service.

  If the status of the subscription service changes from closed to active or from inactive to active, the number of days between the change-of-status date and the next billing cycle start date is added to the contract days counter.

  If the status of the subscription service changes from active to inactive or from active to closed, the number of days between the inactivation date and the start of the next billing cycle is subtracted from the contract days counter. This adjusts the total number of days that were added in the beginning of the cycle.

  If the status of the subscription service changes from inactive to closed or from closed to inactive, the contract days counter is not updated.

You can customize how the contract days counter is updated by implementing the PCM_OP_SUBSCRIPTION_POL_UPDATE_CDC policy opcode.

You can configure the contract days counter to include or exclude the days on which a subscription service changes status (for example, the day the subscription service is created or suspended).

For information on setting up discounts based on contract days, see "Setting Up Discounts Based on Contract Days" in *BRM Managing Accounts Receivable*.

For information on transferring a subscription service, see "About Transferring a Subscription Service to Another Subscriber" in *BRM Managing Customers*.

For more information on subscription management, see "How Service Status Changes Affect Subscription Services" in *BRM Managing Customers*.

About Discounts Based on Monthly Fees and Usage

Important: Support for discounts based on monthly fees and usage is included in Advanced Discounting Manager, an optional feature that requires a separate license.

This type of discount tracks the monthly fees and service usage in a *discount sharing group* (a group owner account that shares discounts with user or child accounts) so that you can grant discounts based on aggregated fees of all users in the discount sharing group.

To implement this feature, BRM provides three aggregation counter resources:

- Parent-level monthly fee and usage counter (MFUC)
- Child-level monthly and usage fee counter (CMFUC)
- Child-level aggregation counter (CHAGC)

When a billing-time discount is applied, the value of the parent counter, MFUC, is copied to the child counter, CMFUC. The child aggregation counter, CHAGC, aggregates all the fees of the child services for real-time discounts.

BRM augments the child aggregation counter when a billable event occurs, and it resets the counter at the end of the billing cycle or whenever an adjustment event updates the monthly fee and usage counter. You can customize how the counter is updated by implementing the PCM_OP_SUBSCRIPTION_POL_NOTIFY_AGGREGATION policy opcode.

Understanding the Discounting Architecture

Discounting for both real-time rating and batch rating takes place in a pipeline.

- For real-time events, discounting occurs in a separate discounting-only pipeline. See "Real-Time Discounting Architecture".
- For batch events, discounting occurs in modules that are part of Pipeline Manager. See "Pipeline Discounting Architecture".

Real-time discounting pipelines and batch rating/discounting pipelines use many of the same function modules and data modules. They also share the same interface for requesting balance data.

Real-Time Discounting Architecture

Discounting for events rated in real time takes place in a pipeline that is used only for this purpose.

Important: Real-Time Discounting is a feature of the Advanced Discounting Manager, an optional feature that requires a separate license.

Discounts are handled by a pipeline in the following way:

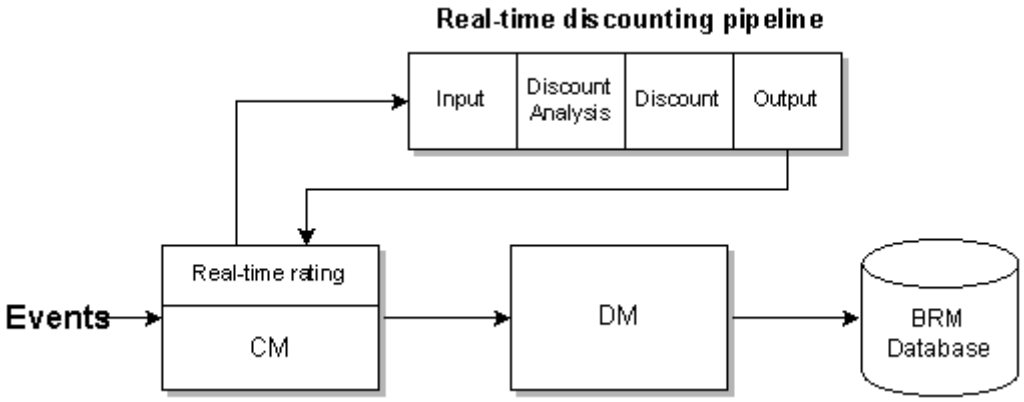
1. During real-time rating, BRM checks if the event qualifies for discounting.
2. If the event needs discounting, BRM sends the event to the CM. The CM adds data used for discounting and sends the event to Pipeline Manager. The data includes the A number.
3. The pipeline NET_EM module receives the discount request and sends it to the appropriate discount pipeline.
4. The INP_Realtime input module converts the data from flist to EDR format and creates an EDR container for the pipeline.

   Note: You can create and run an iScript to manipulate data in the EDR container before it is sent to the pipeline modules. See "About Customizing Mapping of Flist Fields to Rating EDR Container Fields".

5. Pipeline Manager processes the event and discounts it. It adds charge packets and discount packets to the EDR.
6. The OUT_Realtime module does the following:
   - Adds the discount balance impact.
   - Runs an iScript that you can customize to manipulate data. See "Creating iScripts and iRules" in *BRM Developer's Guide*.
   - Converts the data back into flist format.
   - Sends the flist to the NET_EM module.
7. The NET_EM module sends the flist back to the CM with the discount balance impact.
8. The customer's discount balances are updated in the BRM database, and the rated event object is stored.

Figure 10–30 illustrates the flow of data for real-time discounting:

Figure 10–30 Real-Time Discounting Data Flow

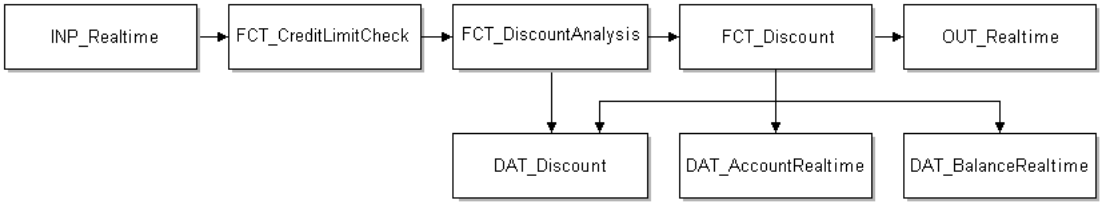


In addition to modules required for input and output, the discounting pipeline includes the following function and data modules:

- The discount analysis module (FCT_DiscountAnalysis) associates discounts with events. See "FCT_DiscountAnalysis".
- The discounting module (FCT_Discount) filters events, checks conditions, and calculates discounts. (See "FCT_Discount" and "Discounting Process Overview".) The FCT_Discount module requests data from the following data modules:
  - DAT_Discount, which stores the pricing data, such as discount models, that are used to process discounts. See "DAT_Discount".
  - DAT_BalanceRealtime, which provides balance data. The DAT_BalanceRealtime module gets data from the BRM database by connecting with the NET_EM module. It does not store data in memory, so it does not load data when you start Pipeline Manager. See "DAT_BalanceRealtime".
  - DAT_AccountRealtime, which provides account data. The DAT_AccountRealtime module gets data from the BRM database by connecting with the NET_EM module. It does not store data in memory, so it does not load data when you start Pipeline Manager. See "DAT_AccountRealtime".
- The credit limit checking module (FCT_CreditLimitCheck) is used during prepaid authorization to determine whether event owners have enough resources in their prepaid account balance to cover the cost of usage. See "FCT_CreditLimitCheck".

Figure 10–31 shows a real-time discounting pipeline and data pool:

Figure 10–31 Real-Time Discounting and Data Pool



For information about setting up a real-time discounting pipeline, see "Configuring a Real-Time Discounting Pipeline".

About Transaction Management for Real-Time Discounting

Because no data is added to the Pipeline Manager database, a real-time discounting pipeline does not use the Transaction Manager (TAM). Instead, transaction handling is provided by the CM.

Data Sent to Pipeline Manager for Real-Time Discounting

The data sent to Pipeline Manager includes:

- The event to discount, including data from the network such as the number, APN, and start time.
- Data pertaining to the A number account, including account number, balance data, product data, service data, and ERA data.

> **Note:** You can filter the types of ERAs considered during real-time rating to improve performance. For information, see "Filtering the ERAs Considered during Rating and Discounting" in *BRM System Administrator's Guide*.

Pipeline Discounting Architecture

Batch discounting is performed by Pipeline Manager. The following function modules process EDRs for discounting:

- The discount analysis module (FCT_DiscountAnalysis) associates discounts with events. This module analyzes the discounts owned by the account associated with the event. It does the following:
  - Compares discount validity dates to the event time.
  - Checks if the event matches an event type in the discount's event usage map.
  - Evaluates whether any discounts should be excluded based on discount exclusion rules. To support exclusion rules for usage discounts, the FCT_DiscountAnalysis module retrieves from the DAT_Discount module the value of the **/config/business_params** object's **ValidateDiscountDependency** entry. See "DAT_Discount".
  - Checks if a discount is active or inactive.

> **Note:** When BRM is configured to use filter sets, Pipeline Manager uses FCT_DiscountAnalysis to apply any standard or promotional discounts and FCT_Filter_Set to apply any system discounts. See "About Using Filter Sets to Apply System Products and Discounts".

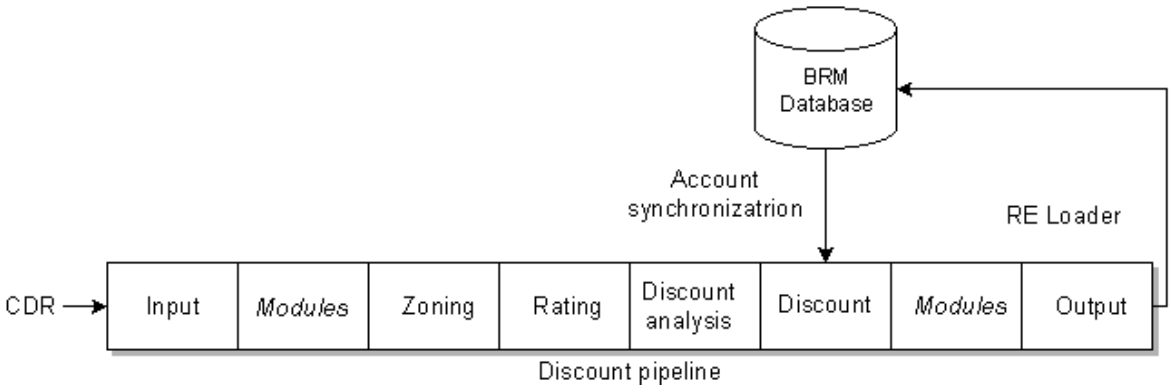
- The discounting module (FCT_Discount) filters events, checks conditions, and calculates discounts. (See "Discounting Process Overview".) The FCT_Discount module requests data from the following data modules:
  - DAT_Discount, which stores the pricing data, such as discount models, that are used to process discounts.
  - DAT_BalanceBatch, which stores balance data. This data is provided at startup by the BRM database and kept up to date by the Account Synchronization feature. See "About Account Synchronization" in *BRM Installation Guide*.

  - DAT_AccountBatch, which stores account data. This data is provided at startup by the BRM database and kept up to date by the Account Synchronization feature.

- The FCT_Rounding module rounds the discount balance impacts.
- The FCT_ApplyBalance module adds the discount balance impacts to the EDR and updates the Pipeline Manager memory.

Figure 10–32 illustrates the flow of data for discounting events rated by Pipeline Manager:

Figure 10–32 Pipeline Manager Discounting Events Data Flow



11

About Implementing Discounts

This chapter describes how to set up different types of Oracle Communications Billing and Revenue Management (BRM) discounts.

For general information about discounting, see "About Discounts". For information about configuring discount modules, see "Configuring Discounting Modules and Components".

Getting Started

Before you begin setting up discounts, you need to determine the kinds of discounts you need and the conditions that should trigger the discounts. For example,

- Do you need discounts based on time of day, billing-time discounts, or usage amounts?
- What types of events and services will the discounts apply to? For example, will the discounts apply to recurring events or to one-time purchase events? How many discounts can apply to any given event?
- What conditions are required to apply the discounts? For example, what level of charge or usage must be reached before the discount applies?
- Can any discounts be shared among several accounts?
- Will you need to perform credit checking for prepaid services?

Using Pricing Center to Configure Discounts

You configure discounts by using the Pricing Center application.

Common Setup Tasks

Every discount requires that you define the following Pricing Center discount components:

- Defining a discount/chargeshare master. A discount master defines the EDR attributes associated with the discount. For more information, see "Filtering EDRs for Discounting".
- Defining a discount/chargeshare rule. A discount rule defines the usage levels and the discount amount that applies to that usage. For more information, see "Defining How Discounts Are Applied".
- Defining discount/chargeshare triggers and conditions. A discount trigger defines the conditions that must be met before a discount can be applied. For more information, see "Determining if Usage Qualifies for Discounting".

- Defining a discount model. A discount model defines the events to discount and the rules and triggers that apply to those events. For more information, see "Grouping Discount Components into Discount Models".
- "Defining a Discount Based on the Number of Subscriptions". A discount object is created when you define a discount. The discount object defines the type of discount, validity settings, and multiple discount handling. For more information, see "Creating Discounts".

You can create the discount components in any order, but you cannot finish configuring some components until you define others. See "Discount Configuration Dependencies".

Discount Configuration Dependencies

Some discount components require other components in their configurations. Therefore, you must define certain components before you can configure others.

Discount component dependencies:

- Discount rules require discount masters.
- Discount models require discount rules and triggers.
- Discount objects require discount models.

You can define discount components that have prerequisites at any time by specifying a name and a description only, and configure them later after setting up the prerequisite components. This is convenient if you prefer to set up a discounting foundation before specifying the discount details.

Based on the component prerequisites, the following order is the most logical way to configure discounts:

1. Configure discount masters.
2. Configure discount triggers.

   Note: Discount triggers have no prerequisites.

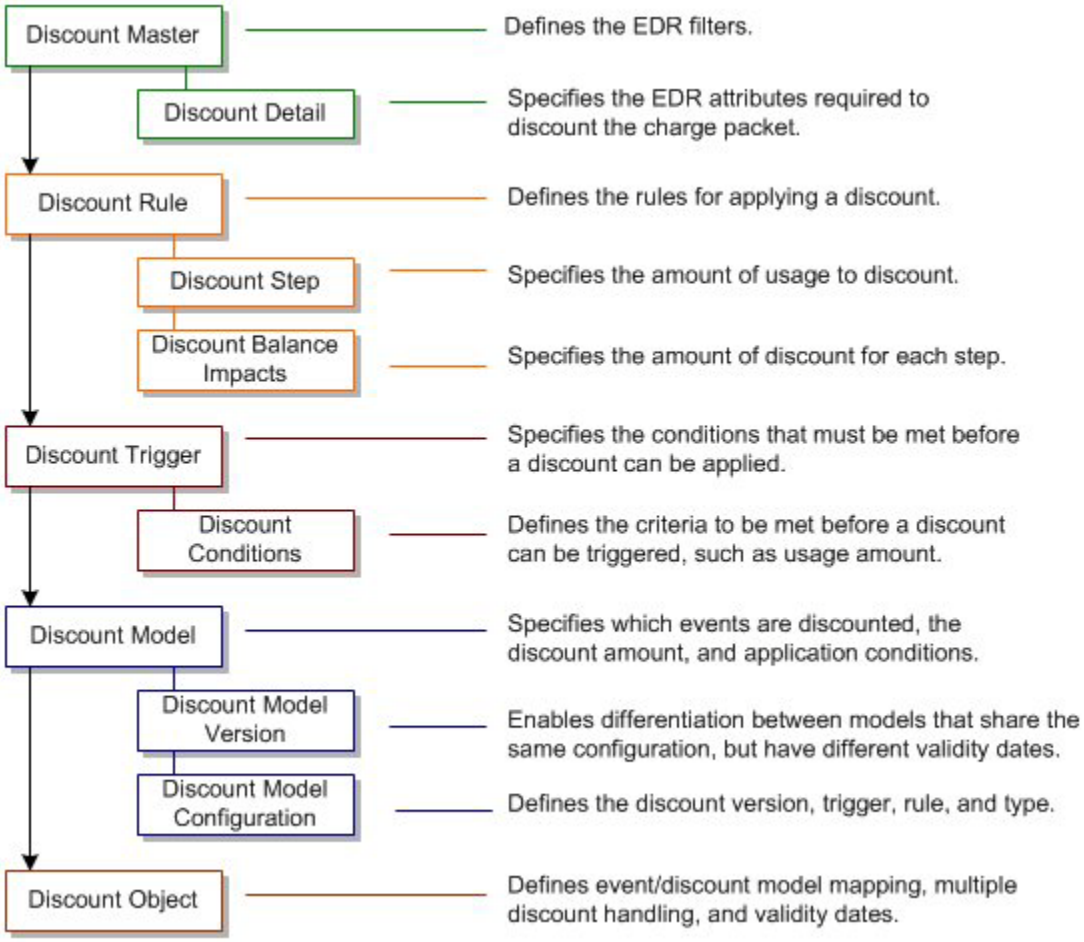
3. Configure discount rules.
4. Configure discount models.

   Tip: Optionally, you can define discount models in the next step when you define new discount objects.

5. Create discount objects.

You define discount objects when you add discounts to deals. You define discount components by using the Pipeline Toolbox. Figure 11–1 shows the Pricing Center component, subcomponents, and what they specify:

Figure 11–1 Pricing Center Components Specification



Example of Using Pricing Center to Configure a Free Minute Discount

The following example shows how you might configure a discount that consumes free minutes. To grant free minutes, you use a cycle event when you define the product. The discount you define determines how the free minutes are consumed.

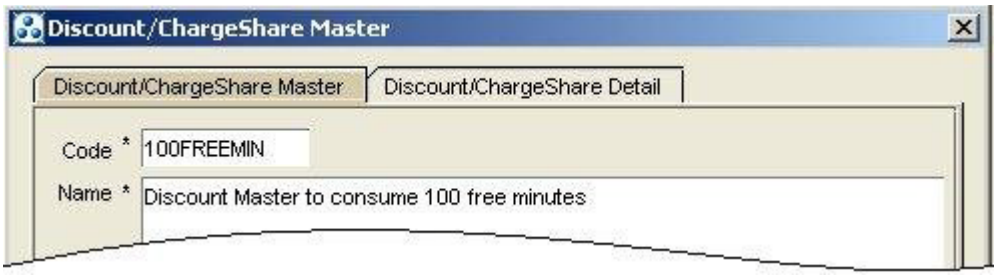
In this example,

- The plan grants 100 free minutes per month.
- The discount consumes free minutes granted by the plan.
- When calls are made, charges for free minutes are discounted and the free minutes balance is modified.

To set up a discount to consume the free minutes granted by the product:

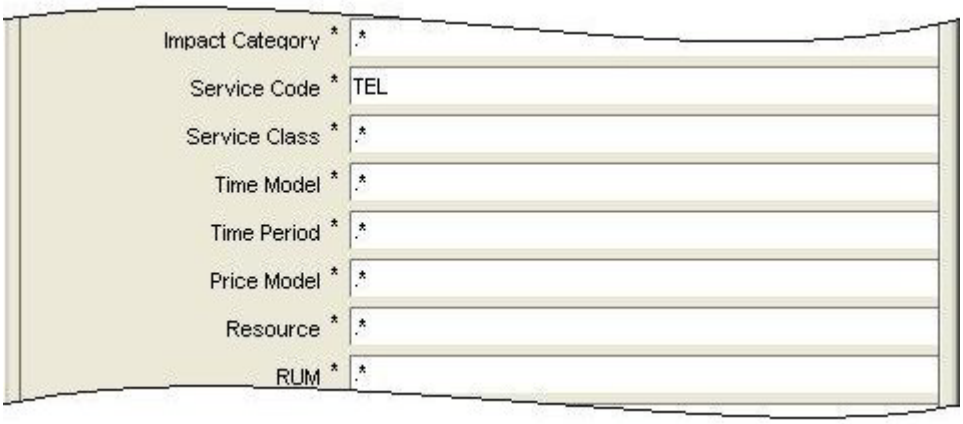
1. Create a discount master as shown in Figure 11–2.

Figure 11–2 Discount/ChargeShare Master Creation



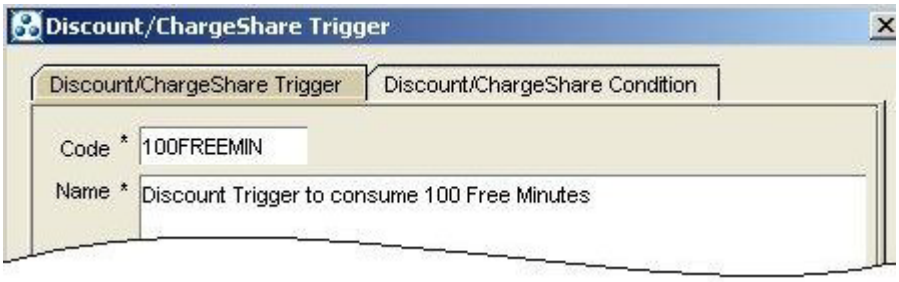
In the **Discount/ChargeShare Detail** tab, specify how to filter the EDR as shown in Figure 11–3. For this discount, the service code is TEL so only charge packets for telephone usage are discounted.

Figure 11–3 Discount/ChargeShare Filter Configuration



2. Define a discount trigger by entering the code and name as shown in Figure 11–4:

Figure 11–4 Discount/ChargeShare Trigger Configuration



3. Define the discount condition to check the account's balance of free minutes. For example, if the resource ID for free minute is **100010**:
   - Define the condition expression as **BAL(100010)**, where **BAL** is an expression that refers to the account balance. For more information, see "Using Expressions in Discount Models".
   - Define the condition operator as **Less Than**. You use this value because free minutes are stored as a negative value.
   - Define the condition value as **0**.

This configuration, shown in Figure 11–5, means that the discount can be applied as long as there are free minutes (resource ID 1000010) available in the account balance.

Figure 11–5 Discount/ChargeShare Condition Configuration

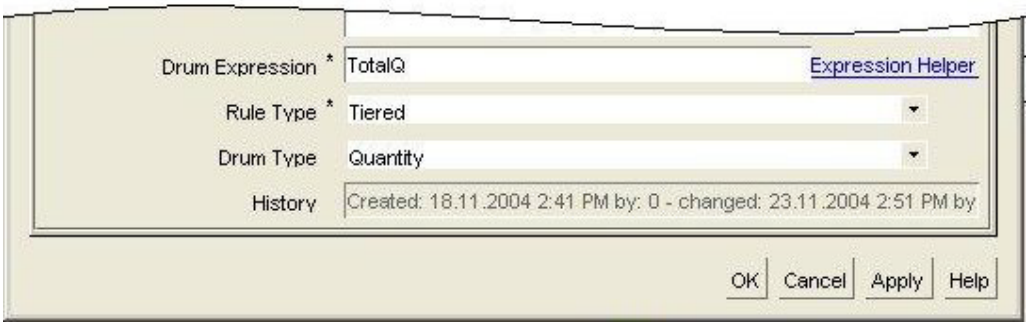
4. Create the discount rule:

   Specify the Discount Master that you created in step 1 as shown in Figure 11–6. This associates the master with this rule.

Figure 11–6 Discount Rule Configuration

Define the DRUM as the total quantity of minutes used for the call by entering **TotalQ** as the DRUM Expression. Specify **Tiered** as the Rule Type, and **Quantity** as the DRUM Type. Figure 11–7 shows these values populated.

Figure 11–7 DRUM, Rule Type and DRUM Type Configuration



5. Create a discount step:

   To make sure that free minutes are always available in the balance to cover the amount in the DRUM, define the lower threshold as **0** and the upper threshold as **Bal(1000010)** (the current balance of free minutes) as shown in Figure 11–8.

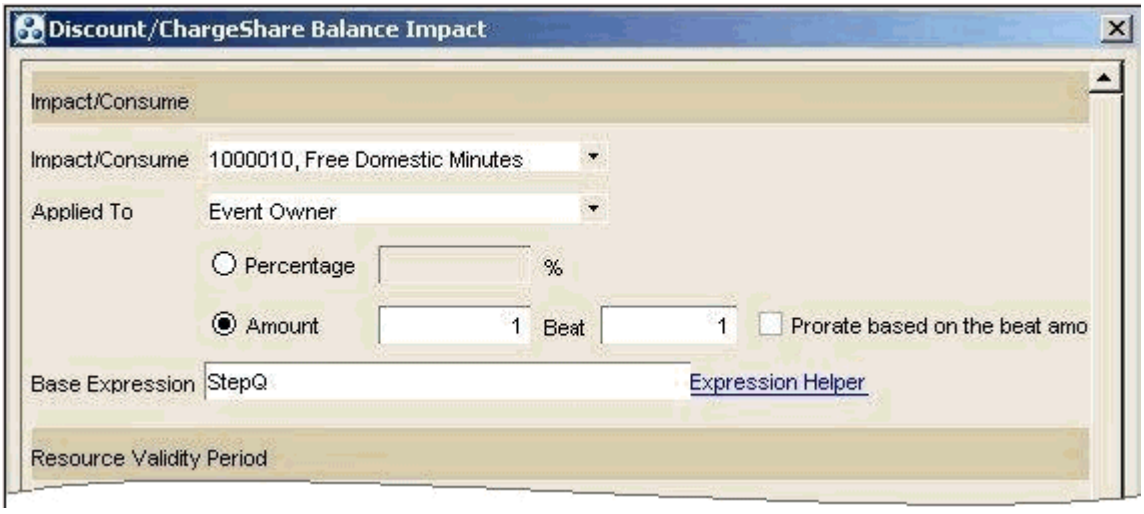
Figure 11–8 Discount/ChargeShare Step Creation

6. Set up two balance impacts: one to decrease the balance of free minutes used and another to discount the usage charges for those free minutes by 100%.

   To decrease the balance of free minutes, define a balance impact that specifies the following values as shown in Figure 11–9:

   - Define the resource consumed as **1000010, Free Domestic Minutes**.
   - Apply the discount to **Event Owner**.
   - Define the amount as **1** and the beat as **1**. This means that for every minute used, a free minute is consumed.
   - Define the base expression as **StepQ**. For more information on expressions, see "Using Expressions in Discount Models".

Figure 11–9 Discount/ChargeShare Free Minutes Balance Impact Configuration

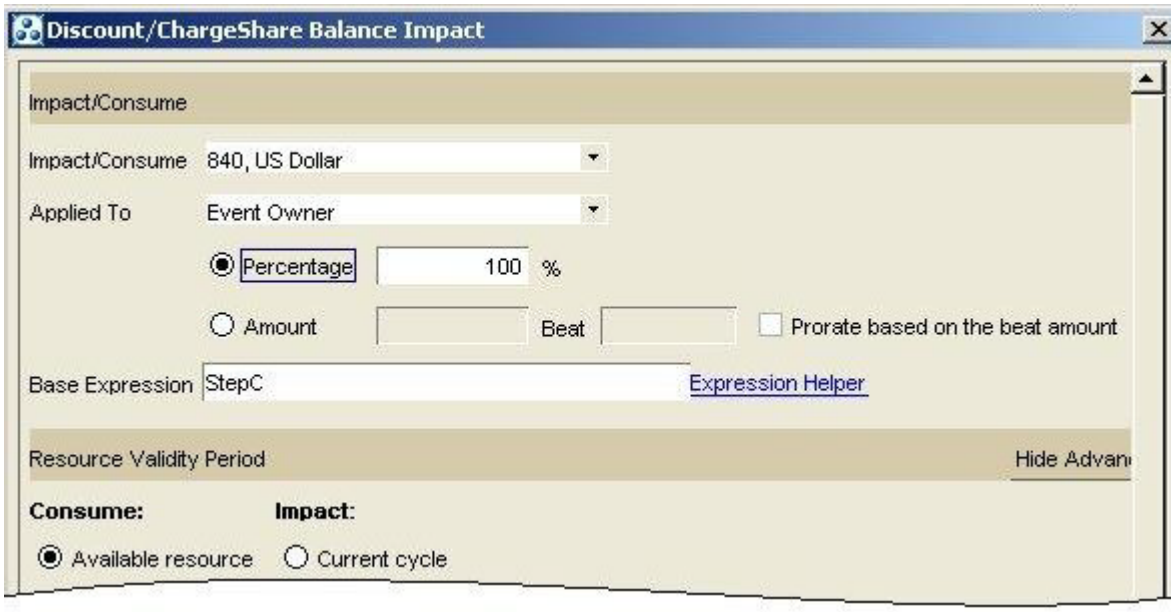


   To discount the usage charge for the free minutes used, define the second balance impact by specifying the following values as shown in Figure 11–10:

   - Define the currency resource consumed, for example, **840, US Dollar**.
   - Apply the discount to **Event Owner**.

- Define the percentage as **100%**.
- Define the base expression as **StepC**. For more information on expressions, see "Using Expressions in Discount Models".

Figure 11–10 Discount/ChargeShare US Dollar Balance Impact Configuration



Using RUMs with Discounts

For real-time discounts, an EDR can have multiple charge packets associated with the same ratable usage metric (RUM).

For pipeline discounts, discounts take place within a single charge packet, which is associated with a single RUM. An EDR can have several charge packets if more than one RUM applies.

Within a single charge packet, only discounts that apply to the associated RUM are eligible. For example, there are two discounts: one that discounts calls (duration) and one that discounts messages (bytes sent). If the RUM in the charge packet is Duration, only the discount for duration is applied.

When two discounts apply to the same RUM for a single charge packet, both discounts are considered. The order in which they are processed is based on the discount's priority, which you set when creating the discount. Whether subsequent discounts after the first are applied to the entire charge or only the remaining undiscounted charge depends on the type of discount. See "About Cascading, Parallel, and Sequential Discounts".

To apply a discount based on the RUM, you filter EDRs by specifying the RUM in the discount master.

Important: When you filter EDRs based on a RUM, use the same RUM in your rating configuration to make sure the quantity that is passed in the EDR is compatible with quantity that is discounted.

Following are three examples of discounts based on RUMs. The first example has two discounts for free minutes, the second example has discounts for free minutes and free bytes, and the third has discounts for free currency and free bytes. For all free minutes discounts, the RUM is set to Duration.

Example of Free Off-Peak Minutes and Free Anytime Minutes discounts

In this example, two cascading discounts apply to the same charge packet in this order:

- Free Off-Peak Minutes: the RUM is Duration. This discount has the highest priority.
- Free Anytime Minutes: the RUM is Duration.

The discounting module receives a charge packet for a 100-minute call that meets the conditions for both of these discounts. The account has 50 free off-peak minutes and 200 free anytime minutes.

1. The Free Off-Peak Minutes discount is applied first because it has the highest priority. All 50 free off-peak minutes are consumed, assuming that the call occurs during off-peak hours.
2. The Free Anytime Minutes discount is applied second and 50 free anytime minutes are consumed, leaving a balance of 150 free anytime minutes that are available for the next EDR.

Example of Free Minutes and Free Bytes discounts

In this example, two discounts apply to two separate charge packets:

- Free Minutes: the RUM is Duration.
- Free Bytes (sent or received): the RUM is Received or Sent.

The discounting module receives an EDR that meets the conditions for both of these discounts. The EDR contains a charge packet for 100 minutes used and another charge packet for 1000 bytes of data sent. The account has 50 minutes available from the Free Minutes discount, and 1024 bytes available from the Free Bytes discount.

1. When the charge packet with a RUM of Duration is rated, discounting finds the discount that filters EDRs based on Duration and applies the Free Minute discount. The 50 free minutes are consumed.
2. When the charge packet with a RUM of Sent or Received is rated, discounting finds the Free Bytes discount and 1000 of the free bytes are consumed.

Example of Free Currency and Free Bytes discounts

In this example, two cascading discounts apply to the same charge packet in this order:

- Free Euros (as non-currency units): applies to all RUMS, so this discount can apply to any charge. This discount has the highest priority.
- Free Minutes: the RUM is Duration.

A charge packet is received for a 100-minute call with a charge of 10 euros. The account has 5 euro units available from the Free Euros discount, and 100 minutes available from the Free Minute discount.

1. The Free Euros discount is applied first because it has the highest priority. The 5 free euros are consumed by the first 50 charged minutes of the call (because the 100-minute call is 10 euros, 5 euros of discount cover 50 minutes). That means 50 minutes remain to which the next discount can apply.

2. The Free Minutes discount is applied next. The 50 free minutes are consumed by the remaining 50 charged minutes of the call, leaving a balance of 50 free minutes that are available for the next EDR.

In this case, even though the balance impacts are applied to different resources, both discounts apply to the same quantity, minutes, so there is no conflict. To implement this type of discount, set the Free Euros discount to apply to any RUM by using an asterisk wildcard when specifying the RUM group, or set the Free Euros discount to apply to the same RUM (Duration) used by the other discount.

Setting Up Cycle-Event Discounts

To set up a cycle-event discount, use Pricing Center to define a discount model, including all necessary components such as versions, configurations, masters, rules, and so on. See "Common Setup Tasks".

When you define the discount in the Discount Attributes dialog box:

- Select **Subscription** in the **Discount type** drop-down list.
- Select the appropriate type of cycle event to discount in the **Event** list under Map an Event to a Discount Model. For example, to apply a discount on monthly fees, select **Monthly Cycle Forward Event**. To apply a discount on a flexible cycle fee, select the cycle forward event associated with that cycle.

> **Note:** If a cycle forward event type does not exist for the flexible cycle you wish to discount, you can configure one. See "About Flexible Cycles" in *BRM Configuring and Running Billing*.

Setting Up Billing-Time Discounts

A billing-time discount is based on an aggregated balance, for example, the total usage for the month or the number of months a customer has been a subscriber. The discount balance impacts are applied when billing is run. For more information, see "About Billing-Time Discounts".

To apply a billing-time discount, you set up the following components:

- Aggregation counter resources. Non-currency resources that track the aggregated amount, for example, the total minutes used.

  If you discount a currency balance based on the aggregated amount of a non-currency resource, you need two non-currency aggregation counters: one that tracks the currency charges and another that tracks the usage. For example, to apply 20% off all charges when the number of minutes used exceeds 500, you set up one counter that stores the aggregated minutes used and another counter that stores the aggregated charges. The discount evaluates the minutes used to determine whether to apply the discount, and then calculates the discount on the total charges counter.

Note: When defining an aggregation counter resource for a billing-time discount, ensure that the resource validity period is set to the billing cycle start and end dates, and the **Credit Limit** and **Credit Floor** are set to None. For more information, see the discussion of setting resource validity period, and credit limits and thresholds in Pricing Center Help.

- Two discounts:
  - A *usage discount* that updates the aggregation counter resource. For example, if the billing-time discount is based on total monthly charges, create a discount that updates the aggregation counter when charges are applied. See "Defining Discounts that Update Aggregation Counters".

    To update more than one aggregation counter, use two balance impacts. For example, you can increment a total charges counter and a minutes used counter.
  - A *billing-time discount* that calculates and applies a discount based on the amount in the aggregation counter, for example, 10% off all usage for the month, or 1000 bonus points for every year of subscription. The calculated discount is applied to the appropriate *account balance*, such as the currency balance or bonus points balance. See "Defining Billing-Time Discounts".

Defining Discounts that Update Aggregation Counters

The following are some specific values to consider when defining a discount that updates an aggregation counter:

- Discount trigger

  Any event that can impact the aggregation counter should trigger the discount; therefore, the condition in the trigger should have no restrictions. You can configure this in two ways:
  - By creating a trigger with no condition. This always applies the discount.
  - By creating a condition that is always passed. For example, use these values:

    Condition Expression = 1

    Condition Operator = Greater than

    Condition Value = 0
- Discount rule
  - DRUM

    Because you always want to apply the balance impact, enter a DRUM value that always falls within the step threshold. For example, enter **1** in the **DRUM Expression** and make the step threshold unlimited.
  - Rule type

    The DRUM always falls within the step threshold so the rule type can be either **tiered** or **threshold**.
  - Step

The entire amount in the DRUM should fall within the step so make the step threshold unlimited (**0** to **infinity**).

– Balance impact

If the billing-time discount is based on usage, the aggregation counter needs to be incremented an amount equal to the usage. To do this, use the following values:

- Impact/Consume = *Counter_balance_resource_ID*

- Amount = 1; Beat = 1

- Base Expression = TotalC if you are tracking total *charges* or **TotalQ** if you are tracking a *quantity* such as minutes used or bytes sent.

If the billing-time discount is based on a non-usage value, such as the number of subscription months, use the following values in the balance impact:

- Impact/Consume = *Counter_balance_resource_ID*

- Amount = The amount by which to increment the aggregation counter. This is typically **1**.

- Base Expression = 1

Defining Billing-Time Discounts

The following are some specific values to consider when defining the discount that is applied when billing is run:

- Discount trigger

  A billing-time discount should be triggered when the amount in the aggregation counter meets the minimum necessary to apply the discount. Therefore, reference the aggregation counter in the condition expression, and enter the required minimum usage as the condition value. Whether the value should be less than or greater than the condition expression depends on whether the aggregation counter is tracked as a negative or positive amount. For example, a balance that tracks charges can be a positive amount and a balance of free minutes can be a negative amount. If there is no minimum amount required, the condition value is **0**.

For example, if the billing-time discount applies 1 bonus point for every 10 minutes of usage, the condition can specify that there is at least 10 minutes in the aggregation counter, which can be defined as:

- Condition Expression = Bal(*aggregation_counter_resource_ID***)**

- Condition Operator = Greater than or equal

- Condition Value = 10

For more information, see "Determining if Usage Qualifies for Discounting".

- Discount rule
  – DRUM

    Reference the account's aggregation counter that contains the aggregated amount by using the discount expression **BAL(***aggregation_counter_resource_ID***)**.

    For more information, see "Defining the Usage Amount to Consider for Discounting".

– Rule type

  Billing-time discounts typically have a **Threshold** rule type because the discount is generally calculated for the entire amount in the aggregation counter. The threshold amount determines which discount balance impact is applied. You define the threshold levels in the discount step.

– Step

  A billing-time discount can have one or more steps. To select from several possible discounts based on the amount in the aggregation counter, use multiple steps. For example, you can offer a 10% discount if the counter is under 300 or a 20% discount if the counter is over 300. For more information, see "How Thresholds Define the Amount of Discount Applied".

– Balance impact

  A balance impact can be applied to a currency or non-currency resource balance. If you want to apply multiple discounts, you can define multiple balance impacts. For example, you can define a balance impact to grant 10% off, and another balance impact to grant bonus points.

  To give a percentage off on a currency balance, use following values in the balance impact:

  - Impact/Consume = *Currency_resource_ID*

  - Percentage = The percentage amount. Enter this as a negative amount, for example **-20%**.

  - Base Expression = Bal(*total_usage_counter_resource_ID***)**

  To give a fixed currency discount, use the following values:

  - Impact/Consume = *Currency_resource_ID*

  - Amount = The fixed amount discount

  - Base Expression = 1

  To grant a non-currency resource discount such as free minutes or bonus points, use the following values:

  - Impact/Consume = *Non-currency_resource_ID*

  - Amount = The quantity to grant. If you are granting a resource in increments, enter the incremental value. For example, to grant 1 bonus point for every 10 minutes of usage, enter **1**.

  - Base Expression = Bal(*total_usage_counter_resource_ID***)** if the grant is not incremental, or **Bal(***total_usage_counter_resource_ID***)/***increment_amount* if the resource is granted in increments. For example, if the aggregation counter tracks minutes and has a resource ID of 100002, to grant 1 bonus point for every 10 minutes of usage, enter **Bal(10002)/10**.

  To grant a fixed quantity of non-currency resource, for example, 500 bonus points when usage exceeds $300.00, use these values:

  - Impact/Consume = *Non-currency_resource_ID*

  - Amount = The quantity to grant

  - Base Expression = 1

  For more information, see "Defining the Threshold Balance Impacts".

Specifying whether Billing-Time Discounts are Inherited by Member Services in Subscription Groups

By default, billing-time discounts owned by a subscription service in a subscription group are automatically inherited by those member services that share the subscription service's balance group. When the discount is applied at billing time, it is applied individually to each member service. (For information about subscription service groups, see "Managing Customers' Subscription-Level Services" in *BRM Managing Customers*.)

If a subscription service owns a billing-time discount that grants a percentage off of subscription fees, the total discount can be greater than intended when the discount is inherited. For example, a subscription group includes one subscription service and ten member services that inherit a billing-time discount. The discount provides .05% off on the group's aggregated charges. If the aggregated charges total $1000.00 when billing is run, the discount grants $5.00 off to each member service that inherits the discount, and $5.00 off to the subscription service. The total discount is $55.00, which is an actual discount of .055% off the aggregated charges.

You can configure BRM to prevent member services from inheriting billing-time discounts owned by the subscription services by setting an inheritance parameter in the BRM Connection Manager (CM) configuration file:

1. Open the CM **pin.conf** file in *BRM_Home***/sys/cm**. *BRM_Home* is the directory where you installed BRM components.
2. Search for the following line:

   ```
   - fm_subscription btd_inheritance = 1
   ```

3. Set the value of the **btd_inheritance** entry to **0**.

   The default is **1**, which means billing-time discounts are inherited.
4. Stop and restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Defining when Billing-Time Discounts Are Applied

BRM performs accounting operations, such as applying cycle fees, rollover, and billing-time discounts, at the end of every accounting cycle. You can configure BRM to apply billing-time discounts at the end of the billing cycle instead of the accounting cycle. You might do this, for example, when the billing cycle spans multiple accounting cycles and a billing-time discount is based on the aggregated usage for the billing cycle.

To configure BRM to apply billing-time discounts at the end of the billing cycle, you modify a field in the **subscription** instance of the **/config/business_params** object.

You modify the **/config/business_params** object by using the **pin_bus_params** utility. For information on this utility, see "pin_bus_params" in *BRM Developer's Guide*.

> **Note:** This configuration applies only to regular billing, not to Bill Now and on-demand billing.

To apply billing-time discounts at the end of the billing cycle:

1. Use the following command to create an editable XML file from the **subscription** instance of the **/config/business_params** object:

```
pin_bus_params -r BusParamsSubscription bus_params_subscription.xml
```

This command creates the XML file named **bus_params_subscription.xml.out** in your working directory. If you do not want this file in your working directory, specify the path as part of the file name.

2. Search the XML file for following line:

```
<BillTimeDiscountWhen>disabled</BillTimeDiscountWhen>
```

3. Change **disabled** to **enabled**.

> **Caution:** BRM uses the XML in this file to overwrite the existing **subscription instance of the /config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM subscription configurations.

4. Save the file and change the file name from **bus_params_subscription.xml.out** to **bus_params_subscription.xml**.
5. Use the following command to load this change into the **/config/business_params** object:

```
pin_bus_params bus_params_subscription.xml
```

You should execute this command from the *BRM_Home***/sys/data/config** directory, which includes support files used by the utility. To execute it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.

6. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

See "Using testnap" in *BRM Developer's Guide* for general instructions on using **testnap**. See "Reading Objects by Using Object Browser" in *BRM Developer's Guide* for information on how to use Object Browser.

7. Stop and restart the Connection Manager (CM). See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Calculating Billing Time Discounts Based on Validity Dates

By default, billing-time discounts are calculated based on the available balances with validity dates that both start and end in the current billing cycle. For example, if the billing cycle is March 1 to April 1, the available balances include only those sub-balances that both start and end between March 1 and April 1. The balance does not include any sub-balances with a validity start date before March 1 or with a validity end date after April 1.

However, you can configure BRM to calculate billing-time discounts based on the available balances that are valid for a portion of the billing cycle. For example, if the billing cycle is March 1 to April 1, the balance on which to apply the discount could include the following:

- A sub-balance that is valid from February 15 to March 15
- A sub-balance that is valid from March 10 to March 20.
- A sub-balance that is valid from March 20 to April 20.

You configure BRM to calculate billing-time discounts based on balances that are valid for a portion of the billing cycle by modifying a field in the **activity** instance of the **/config/business_params** object.

> **Note:** This configuration applies only to regular billing, not to Bill Now and on-demand billing.

To calculate billing-time discounts based on balances that are valid during a portion of the billing cycle:

1. Use the following command to create an editable XML file from the **activity** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsActivity bus_params_activity.xml
   ```

   This command creates the XML file named **bus_params_activity.xml.out** in your working directory. If you do not want this file in your working directory, specify the path as part of the file name.

2. Search the XML file for following line:

   ```
   <BillingTimeDiscountBasedOnPeriod>0</BillingTimeDiscountBasedOnPeriod>
   ```

3. Change **0** to **1**.

   > **Caution:** BRM uses the XML in this file to overwrite the existing **activity instance of the /config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM subscription configurations.

4. Save the file and change the file name from **bus_params_activity.xml.out** to **bus_params_activity.xml**.

5. Use the following command to load this change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_activity.xml
   ```

   You should execute this command from the *BRM_Home***/sys/data/config** directory, which includes support files used by the utility. To execute it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.

6. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   See "Using testnap" in *BRM Developer's Guide* for general instructions on using **testnap**. See "Reading Objects by Using Object Browser" in *BRM Developer's Guide* for information on how to use Object Browser.

7. Stop and restart the Connection Manager (CM). See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Example of Granting 50 Frequent Flyer Miles for Every Hour of Phone Calls

This discount example grants 50 frequent flyer miles for every hour of phone calls made by the subscriber that owns the discount. There are two discounts to configure:

- **Count Minutes discount** increments a counter resource that tracks minutes used when the subscriber makes phone calls. The counter resource is named **Minutes Used** and its resource ID is **1000015**.
- **Frequent Flyer Miles discount** grants frequent flyer miles at billing time based on the total minutes used in the counter resource. The **Frequent Flyer Miles** resource ID is **1000003**.

Frequent flyer miles are granted for each whole hour. Partial hours are not counted. The maximum number of frequent flyer miles that can be accrued is 50,000.

Along with the typical discount configurations required when setting up a discount (see "Common Setup Tasks"), you enter the following values specific to this discount example:

Count Minutes Discount

Discount trigger:

- **Condition Expression = 1**
- **Condition Operator = Greater than**
- **Condition Value = 0**

This condition always passes so the discount is always processed.

- **Discount rule:**
  - **DRUM Expressions = 1**
  - **DRUM Type = Quantity**
  - **Rule Type = Tiered** or **Threshold**
- **Discount Step:**
  - **Threshold From = 0**
  - **Threshold To = infinity**
  - Ignore the proration settings. They apply only to cycle fee discounts.
    - **Impact/Consume** = 1000015, Minutes Used
    - **Amount** = 1
    - **Beat** = 1
    - **Base Expression** = **TotalQ**

      where, *TotalQ* refers to the total quantity used in the charge packet. **Amount** and **Beat** are both 1 to add a minute of usage to the counter for every minute in the charge packet.
- **Impact: Current cycle**

  You impact the current cycle to count the minutes used for this cycle.
- **Discount object**
  - **Discount Type = Subscription**
  - **Applied To = /service/telco/gsm**
  - **Multiple discounts per event** = **Parallel**

    Map the following events to the discount model for this discount:
    - **Real Time Telco GSM Session**

    * **Delayed Telco GSM Session**

Frequent Flyer Miles Discount

- **Discount trigger**
  - **Condition 1**: The subscriber has not reached the maximum number of frequent flyer miles allowed (50,000):

    - Condition Expression = Bal(1000003) (The Frequent Flyer Miles balance)

    - Condition Operator = Less than

    - Condition Value = 50,000

    If the maximum allowed has been reached, the subscriber must reduce the number of miles by using some or all of them before this discount can be applied.
  - **Condition 2:** The number of minutes used is at least 60. If the user did not make at least 60 minutes of calls, no frequent flyer miles are granted:

    - Condition Expression = Bal(1000003) (The Frequent Flyer Miles balance)

    - Condition Operator = Greater than or equal to

    - Condition Value = 60
- **Discount rule:**
  - **DRUM Expressions = 1**
  - **DRUM Type = Quantity**
  - **Rule Type = Threshold**
- **Discount Step:**
  - **Threshold From = 0**
  - **Threshold To = infinity**
  - Ignore the proration settings. They apply only to cycle fee discounts.
    * **Impact/Consume** = 1000015, Frequent Flyer Miles
    * **Applied To = Event Owner**
    * **Amount = 50**

      50 frequent flyer miles are granted for every hour of usage.
    * **Beat = 0**
    * **Base Expression** = **Bal(1000015)/60**

      To calculate the number of hours, the balance of minutes used (1000015) is divided by 60, the number of minutes in an hour.
    * **Impact: Date range; Start = 1/1/05, End = Never**

      These granted frequent flyer miles never expire, but you can specify an end date if you want. A start date is specified to allow all miles to be stored in the same sub-balance. (Resources with different validity periods are stored in separate sub-balances.) This is not required, but is more efficient.
- **Discount object**
  - **Discount Type = Subscription**

- **Applies To = /service/telco/gsm**

  This discount applies to GSM telephone service only.

- **Multiple discounts per event = Parallel**
- Map the **Billing Time Discount Event** to the discount model for this discount.

Sample Rating for Frequent Flyer Miles Discount

This section describes how the billing-time example above is rated.

1. The subscriber purchases a wireless plan. The Minutes Used and Frequent Flyer Miles balances are both 0.
2. The subscriber makes a 30 minute call. The Count Minutes discount is triggered by the call event. Because the condition in the trigger always passes when the discount is evaluated, discounting adds 30 units to the Minutes Used account balance.
3. During the month, the subscriber continues to make calls, and the counter is updated with each call. At the end of the month, the subscriber has made 450 minutes of calls.
4. When billing is run, the Frequent Flyer Miles discount is triggered by the billing event. The discount is evaluated because both conditions in the trigger pass: the subscriber made more than 60 minutes of calls, and the balance of frequent flyer miles is less than the maximum of 50,000.
5. The Frequent Flyer Miles discount divides the balance of minutes used by 60 to get 7.5 hours, and then adds 50 units to the frequent flyer miles balance for each whole hour, resulting in a balance of 350 miles.

Setting Up Shared Discounts

Accounts that sponsor other accounts can share discounted resources.

This section describes how to set up three kinds of shared discounts: discount sharing, charge sharing, and snowball discounts.

Setting Up Discount Sharing

In discount sharing, members of a discount sharing group benefit from the discounts of the group owner:

- **Currency discounts**: If the shared discount is currency based (for example, a 5% discount on a monthly cycle fee), a currency discount is granted to the balance of the member who generated the discounted event. The group owner sees no balance impact from this event.
- **Non-currency discounts**: If the shared discount is not currency based (for example, a discount of 30 free minutes on long distance calls), both the owner and member have a balance impact. When a member generates a discounted event and the discount is applied, the discount impacts the shared non-currency resources of the group owner. A corresponding currency discount is granted to the balance of the member who generated the event.

  Non-currency-based shared discounts are also used to aggregate service usage when the owner of a discount sharing group receives a discount based on the aggregated usage of the group members. When a member generates a usage event and the discount is applied, the discount impacts the aggregation counter of the

group owner. See "About Using Discounts to Aggregate Usage".

To set up discount sharing:

1. Define the shared discounts including all necessary components such as versions, configurations, masters, and so on. See "Common Setup Tasks".

   When creating the discount balance impacts associated with the discount rule:

   - For currency discounts, choose **Event Owner** in the **Applies To** field of the Discount/ChargeShare Balance Impact dialog box. The event owner is the account that generated the event.
   - For non-currency discounts, create two balance impacts:

     A balance impact that reduces the shared resource of the discount owner. Choose **Discount Owner** in the **Applies To** field of the Discount/ChargeShare Balance Impact dialog box.

     A balance impact that discounts the usage fee for the free resources used by the discount sharing group member. Choose **Event Owner** in the **Applies To** field of the Discount/ChargeShare Balance Impact dialog box.

2. Create a discount object in which you map an event type to the discount model you created. For more information, see "Defining a Discount Based on the Number of Subscriptions".
3. Create a discount sharing group by defining the group owner, the group members, and a list of the discounts that will be shared by the group. You create discount sharing groups by using Customer Center. See "Create Discount Sharing Groups".

For information about how BRM handles discount sharing groups, see "About Discount Sharing Groups" in *BRM Managing Accounts Receivable*.

Setting Up Charge Sharing

When you set up charge sharing, members of a charge sharing group have some of their charges sponsored by the charge sharing group owner. For example, a business can sponsor 100% of the GSM charges for employee calls made during normal working hours.

Setting up charge sharing is similar to setting up discounting. (See "Discounting Process Overview".) You set up components that work together to determine when and in what ways a particular charge is shared.

- A top-level object, called a chargeshare, is similar to a discount object. It includes a name and other basic information, along with validity dates. The chargeshare also includes a usage map that links particular event types to chargeshare models.
- A chargeshare model functions in the same way as a discount model. It filters to determine whether an event qualifies for charge share, checks conditions to determine whether a particular chargeshare model applies, and applies chargeshare rules to determine the actual charge sharing amounts and balance impacts. Chargeshare models and discount models share many of the same components, including masters, rules, and triggers.

In contrast to discounts, however, chargeshares are not purchasable items included in deals. They are linked to accounts and services via charge sharing groups.

To set up charge sharing, do the following:

1. Define a chargeshare model, including all necessary components such as versions, configurations, masters, and so on.

2. For each step in the chargeshare model, create two balance impacts for the currency resource:
   - A balance impact for charge sharing group members. This typically *reduces* the charge. For this balance impact, choose **Event Owner** in the **Applies To** field of the Discount/ChargeShare Balance Impact dialog box.
   - A balance impact for the charge sharing group owner. This balance impact typically *increases* the charge. For this balance impact, choose **ChargeShare Owner** in the **Applies To** field of the Discount/ChargeShare Balance Impact dialog box.

   Important: Both balance impacts must impact the same currency resource and must be for the same amount or percentage.

3. Create a chargeshare object in which you map an event type to the chargeshare model you created. You can configure the chargeshare to include additional event type to chargeshare mapping.
4. Create a charge sharing group by defining the group owner, group members, and a list of the sponsored charges that will be shared by the group. See "About Charge Sharing Groups" in *BRM Managing Accounts Receivable*.

Setting Up Snowball Discounts

You can configure a billing-time discount as a *snowball discount*. A snowball discount allows distribution of group discounts to discount sharing group members. A snowball discount can be distributed evenly among all accounts or based on the amount of usage for each account.

To create a snowball discount:

1. Create an aggregation counter resource to track the usage for all accounts. (See "Setting Up Resources" in *BRM Setting Up Pricing and Rating*.) Add this resource to the plans that are purchased by the owner and members of the discount sharing group.
2. Define two shared discounts. Set up their components as described in "Common Setup Tasks", following these guidelines:
   - Set up a usage discount that updates the aggregation counter when the accounts or services in the discount group generate usage. When you add this discount to a plan, map the discount model to the usage event for which the discount applies, for example, the Telco GSM session event. Each member in the discount sharing group should own this discount.
   - Set up a billing-time discount that applies the discount based on the amount in the aggregation counter. When you add the discount to a plan, check the **Snowball** option and map the discount model to the billing-time event (**Billing Time Discount Event**).
3. Set up a discount sharing group by defining the group owner, group members, and a list of the discounts that will be shared by the group. See "Create Discount Sharing Groups".
4. Specify how the discount is distributed in the **pin_snowball_distribution** file and load the file by running the **load_pin_snowball_distribution** utility. See "Defining How Snowball Discounts Are Distributed".

The distribution you specify in the **pin_snowball_distribution** file becomes the default distribution. You can customize this behavior by modifying the policy opcode PCM_OP_SUBSCRIPTION_POL_SNOWBALL_DISCOUNT.

5. If the discount is to be distributed unevenly, for example, if each member gets a percentage of the discount based on its own usage, configure sub-balance contributors. Specify the contributors in the **pin_sub_bal_contributor** file and load the file by running the **load_pin_sub_bal_contributor** utility. For information about configuring sub-balances, see "About Configuring Sub-Balances" in *BRM Setting Up Pricing and Rating*. For instructions on configuring the **pin_sub_bal_contributor** file, see "Configuring Sub-Balances" in *BRM Setting Up Pricing and Rating*.

Defining How Snowball Discounts Are Distributed

To specify how snowball discounts are distributed, edit the **pin_snowball_distribution** file, and then run the **load_pin_snowball_distribution** utility to load the contents of the file into the **/config/snowball_distribution** object in the BRM database.

You set two values in the **pin_snowball_distribution** file:

- The name of the discount (defined when you set up the discount)
- The record ID
  - If set to zero, the discount is distributed evenly among all members of the discount sharing group.
  - If set to greater than zero, this value is taken to be the contributing account's resource ID and the discount is distributed based on how much each account contributed to the amount in the aggregated balance. The resource ID identifies the resource in the member account that contributes to the aggregated balance.

Note: If the record ID is greater than zero, you must configure sub-balance contributors to enable BRM to track each account's contribution. For more information, see "About Configuring Sub-Balances" in *BRM Setting Up Pricing and Rating*.

For example, the following entries in the **pin_snowball_distribution** file specify that **snowball one** is distributed evenly and **snowball two** is distributed based on the amount each account contributed to the total usage counter (resource ID 1000501) at the end of the billing cycle.

```
snowball one : 0
snowball two : 1000501 # Total usage counter resource
```

Important: The **load_pin_snowball_distribution** utility needs a configuration (**pin.conf**) file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

> **Caution:** The **load_pin_snowball_distribution** utility overwrites existing snowball distribution rules. If you are updating snowball distribution rules, you cannot load new snowball distribution rules only. You must load complete sets of snowball distribution rules each time you run the **load_pin_snowball_distribution** utility.

To configure snowball distribution rules:

1. Edit the **pin_snowball_distribution** file in *BRM_Home*/**sys/data/pricing/example**. The **pin_snowball_distribution** file includes instructions.
2. Save the **pin_snowball_distribution** file.
3. Use the following command to run the **load_pin_snowball_distribution** utility:

   ```
   load_pin_snowball_distribution pin_snowball_distribution
   ```

   If you do not run the utility from the directory in which the file is located, you must include the complete path to the file, for example:

   ```
   load_pin_snowball_distribution BRM_Home/sys/data/pricing/example
   ```

   > **Tip:** If you copy the **pin_snowball_distribution** file to the directory from which you run the **load_pin_snowball_distribution** utility, you do not have to specify the path or file name. The file must be named **pin_snowball_distribution**.

4. Stop and restart the Connection Manager (CM). See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

To verify that the network elements were loaded, you can display the **/config/snowball_distribution** object by using the Object Browser, or use the **robj** command with the **testnap** utility. See "Discounting Process Overview" and "Reading an Object and Writing Its Contents to a File" in *BRM Developer's Guide*.

For more information, see "load_pin_snowball_distribution".

Setting Up Discounts that Consume Resource Grants

You can create discounts that reduce (consume) the amount of a non-currency resource granted by plans at the time the grant is applied. This is different than consuming granted resources when subscribers use their services. A discount that consumes a resource grant applies to the events that grants the resource, such as cycle fee and purchase events.

By default, discounts can consume a non-currency resource from any sub-balance that contains that resource, consuming the sub-balance with the earliest start date first. However, when a discount consumes a resource at the time the resource is granted, the consumption is restricted to that particular grant only; the discount does not impact any other sub-balances for that resource, even when existing sub-balances have earlier start times. For an example of this discount, see "Example of Consuming a Resource Grant".

For information about the order in which non-currency resources are typically consumed, see "Specifying the Order in Which Resource Sub-Balances Are Consumed" in *BRM Setting Up Pricing and Rating*.

To create a discount that consumes a resource at the time the resource is granted, configure the discount to consume the resource that is granted and map the discount model to the event that grants the resource, such as the monthly cycle forward event.

Example of Consuming a Resource Grant

This example shows how a discount consumes the free minutes granted Plan X when the plan is purchased, and does not impact other free-minute balances.

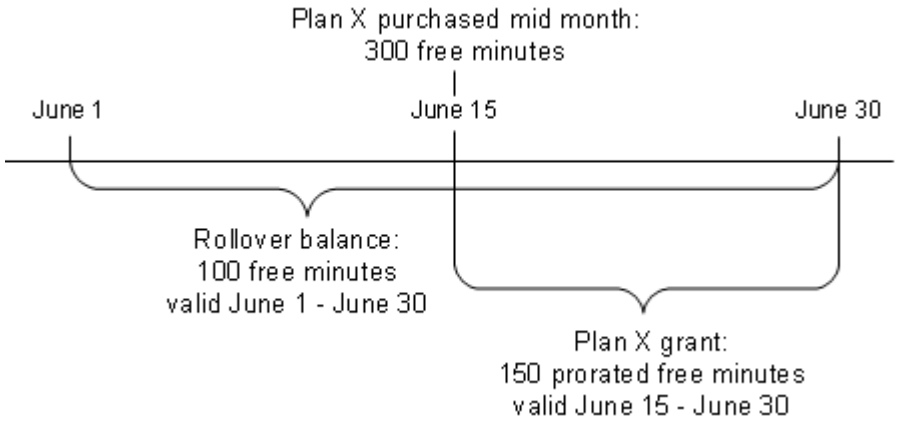
In this example, a subscriber purchases Plan X in mid cycle on June 15. Plan X grants 300 free minutes, which are prorated for the month, resulting in a grant of 150 free minutes valid from June 15 through June 30.

On June 15, at the time the subscriber purchases Plan X, the subscriber's account also includes:

- A cycle fee discount that reduces (consumes) free minutes by 10%.
- A balance of 100 free rollover minutes that are valid from June 1 to June 30.

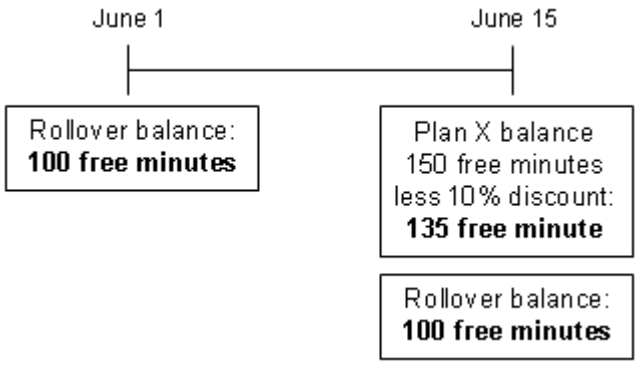
Figure 11–11 illustrates the subscriber's free minute resource balances.

Figure 11–11 Subscriber's Free Minutes Resource Balances



On June 15, when Plan X grants 150 free minutes, the cycle-fee discount is applied. The discount reduces the grant amount by 10% (15 minutes), resulting in a final grant of 135 free minutes. The discount does not impact the rollover balance of 100 free minutes. Figure 11–12 illustrates the subscriber's free minute resource balances after the discount is applied.

Figure 11–12 Subscriber's Free Minutes Resource Balances after Discount



Note: This example assumes the subscriber has not used any rollover minutes between June 1 and June 15.

Setting Up Volume-Based Discounts

Volume-based discounts are granted based on threshold values that define levels of usage. Volume-based discounts are generally granted as billing-time discounts or cycle fee discounts.

To set up volume-based discounts, see the following sections:

- Setting Up Discounts Based on Number of Subscriptions
- Setting Up Discounts Based on Contract Days
- Setting Up Discounts Based on Monthly Fees and Usage

To learn more about volume-based discounts, see "About Volume-Based Discounts".

Setting Up Discounts Based on Number of Subscriptions

To set up discounts based on the number of subscriptions, perform the tasks described in the following sections:

1. Configuring BRM to Track the Number of Subscriptions
2. Defining a Discount Based on the Number of Subscriptions
3. Setting Up Plans for the Number-of-Subscriptions Discount
4. Creating an Account Hierarchy
5. Setting Up a Discount Sharing Group for the Number-of-Subscriptions Discount
6. (Optional) Configuring and Defining Exclusion Rules.

Configuring BRM to Track the Number of Subscriptions

To enable BRM to track the number of subscriptions, you need to uncomment an entry in the Connection Manager (CM) configuration file:

1. Open the Connection Manager (CM) **pin.conf** file in *BRM_Home*/**sys/cm**.
2. Uncomment the following line in the file:

```
- fm_subscription  non_currency_lc 1000101
```

3. Stop and restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Defining a Discount Based on the Number of Subscriptions

How you set up this discount depends on:

- The amount of discount you want to offer.
- Which resources you will grant or impact.
- Whether the discount is a billing-time discount or a cycle fee discount.

  If you want the number-of-subscriptions discount to be a billing-time discount, you also need to set up an aggregation discount to aggregate the service usage fees. For more information, see "Setting Up Billing-Time Discounts".

To set up a discount based on number of subscriptions, use Pricing Center to define the discount components (discount master, triggers, rules, and model) as described in "Common Setup Tasks". When defining the discount components, follow these guidelines:

- In the Discount/ChargeShare Rule dialog box, specify the following values:
  - Enter **Bal(1000101)** in the **Drum Expression** field on the **Discount/ChargeShare Rule** tab. This specifies the Line Counter balance that contains the total number of subscriptions.
  - Select **Threshold** in the **Rule Type** field on the **Discount/ChargeShare Rule** tab.
  - Select **Quantity** in the **Drum Type** field.
  - In the **Discount/ChargeShare Step** tab, create a step for each subscription level to discount by specifying the lower and upper limits of the threshold in the **Threshold From** field and **Threshold To** field.

    For example, to set up three levels: 0 to 100 subscriptions, 101 to 200 subscriptions, and 201 or more subscription, set up three steps:

    Step 1: **Threshold From** = **0**, **Threshold To** = **100**

    Step 2: **Threshold From** = **101**, **Threshold To** = **200**

    Step 3: **Threshold From** = **201**, **Threshold To** = **Infinity**
- In the Balance Impact dialog box for each step:
  - Select **Event Owner** in the **Applied To** field.
  - In the **Base Expression** field, specify the balance that contains the aggregated service fees; for example, **Bal(***aggregation_counter_resource_ID***)**.

Setting Up Plans for the Number-of-Subscriptions Discount

To set up price plans that offer discounts based on the number of subscriptions, use Pricing Center to perform the following tasks:

- Setting up Subscription Services
- Adding the Line Counter Resource
- Adding the Discount to Plans and Deals

Setting up Subscription Services

Each child account must purchase a plan that includes a subscription service. Create a subscription service group in every plan that will be purchased by a child account in the account group. You create subscription service groups in the **Plan Attributes** tab of the Plan Attributes dialog box. See "About Subscription Services" in *BRM Managing Customers*.

Adding the Line Counter Resource

You add the Line Counter resource to the plan that includes the number-of-subscriptions discount. This is the plan that is purchased by the parent account. The number of active subscriptions is then stored in the parent account's Line Counter balance.

In the **Track Balances** tab of the Plan Attributes dialog box, select the account's balance group and add the **Line Counter** resource in the Credit Limits area.

Important: The Line Counter resource must be associated with only one plan in the parent account.

For more information, see "About Setting Limits and Thresholds" in *BRM Managing Accounts Receivable*.

Adding the Discount to Plans and Deals

After you define the discount components (the discount master, triggers, rules, and model), you create the discount by adding it to your price list.

Insert a new discount in the price list and follow these guidelines when configuring the discount in the Discount Attributes dialog box:

- Select **Subscription** in the **Discount Type** field.
- Select **Account** in the **Applies To** field. You can apply the discount to a service or an account.
- In the Map an Event to a Discount Model section:
  - Map the discount to the billing-time event (**Billing Time Discount Event**) to make it a billing-time discount, or to a cycle fee event to make it a cycle fee discount.
  - Select the discount model you set up for the number-of-subscriptions discount from the list of discount models.

For more information, see "Defining a Discount Based on the Number of Subscriptions".

You can then add the number-of-subscriptions discount to a deal in the Deal Attributes dialog box. Add the discount to the deal that will be purchased by the corporate parent account.

Creating an Account Hierarchy

Set up a hierarchical account group. If you are offering corporate discounts, make the corporate account the parent and add a child account to the group for each employee that has a subscription.

Note: The parent account can also be the child of another account.

You create account hierarchies by using Customer Center. See "Creating Hierarchical Groups" in *BRM Managing Accounts Receivable*.

Setting Up a Discount Sharing Group for the Number-of-Subscriptions Discount

You set up a discount sharing group to share the number-of-subscriptions discount. When the discount is shared, it is granted to each account in the discount sharing group.

Make the account group's parent account the owner of the discount sharing group. Add the owner's number-of-subscriptions discount as the shared discount. Add the subscription services in the child accounts as members of the discount sharing group.

Important: The discount sharing group must include all child accounts' subscription services in the account group to ensure the number of active subscription services is accurately counted.

At the end of the cycle, each child account receives a discount on its fees based on the total number of subscriptions counted.

You set up discount sharing groups by using Customer Center. See "Create Discount Sharing Groups".

Setting Up Discounts Based on Contract Days

Important: Support for discounts based on contract days is included in Advanced Discounting Manager; an optional feature that requires a separate license.

To provide a discount based on the cumulative number of contract days, the charges to discount need to be associated with the contract days counter (CDC) and contract days counter for discount (CDCD) resources.

To do this, you create a real-time discount that aggregates the cycle fee charges into a non-currency total charge counter. The other main configuration task is creating a billing-time discount that discounts the customer's charge balance, using the total charge counter created in the real-time discount. Basically, you are using:

- CDC as the discount ratable usage metric (DRUM) for usage fee discounts.
- CDCD as the DRUM for billing-time discounts.

This bases the discount tier for the billing-time discount on CDCD, and then applies the discount to the currency charges.

To set up discounts based on contract days:

1. Enable the discount based on contract days feature. See "Enabling Support for Discounts Based on Contract Days".
2. Use BRM Pricing Center to complete these procedures:
   a. Configuring BRM to Track the Number of Contract Days
   b. Creating a Real-Time Aggregation Discount
   c. Creating a Billing-Time Discount

After completing the above tasks, associate the discount objects with a deal at the subscription service level, associate the deal with a plan, and so on. See "Grouping Discount Components into Discount Models" and "Grouping Discount Components into Discount Models" in *BRM Managing Accounts Receivable*.

You can configure BRM to count or exclude the day in which a subscription service changes status. See "Specifying whether to Count the Days on which Subscription Status Changes".

Enabling Support for Discounts Based on Contract Days

By default, support for discounts based on contract days is disabled in BRM. You can enable this feature by modifying the **subscription** instance of the **/config/business_params** object.

To enable support for discounts based on the number of contract days:

1. Use the following command to create an editable XML file from the **subscription** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsSubscription bus_params_subscription.xml
   ```

   This command creates the XML file named **bus_params_subscription.xml.out** in your working directory. If you do not want this file in your working directory, specify the path as part of the file name.

2. Search the XML file for following line:

   ```
   <DiscountBasedOnContractDaysFeature>disabled</DiscountBasedOnContractDaysFeatur
   e>
   ```

3. Change **disabled** to **enabled**.

   Caution: BRM uses the XML in this file to overwrite the existing **subscription** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM subscription configurations.

4. Save the file and change the file name from **bus_params_subscription.xml.out** to **bus_params_subscription.xml**.

5. Use the following command to load this change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_subscription.xml
   ```

   You should execute this command from the *BRM_Home***/sys/data/config** directory, which includes support files used by the utility. To execute it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.

6. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   See "Using testnap" in *BRM Developer's Guide* for general instructions on using **testnap**. See "Reading Objects by Using Object Browser" in *BRM Developer's Guide* for information on how to use Object Browser.

7. Stop and restart the Connection Manager (CM). See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Configuring BRM to Track the Number of Contract Days

To enable BRM to track the number of contract days, you need to uncomment entries in the Connection Manager (CM) configuration file:

1. Open the Connection Manager (CM) **pin.conf** file in *BRM_Home***/sys/cm**.

2. Uncomment the following lines in the file:

   ```
   - fm_subscription   non_currency_cdc  1000099
   - fm_subscription   non_curr_cdcd     1000100
   ```

3. Stop and restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Creating a Real-Time Aggregation Discount

To create a discount to aggregate customer fees and usage:

1. Create a discount master using the default values.
2. Create a discount rule:
   a. From the Pipeline Toolbox, select **Discount/ChargeShare Rule**.
   b. Select the following values:

      Expression: TotalQ

      Rule Type: Threshold

      Drum Type: Quantity
3. Create a discount step:
   a. In the **Discount/ChargeShare Step** tab of the Discount/ChargeShare Rule dialog box, click **New**. The Discount/ChargeShare Step dialog box opens.
   b. Select the following values:

      Threshold From: 0

      Threshold To: infinite

      Prorate Purchase: Full Discount

      Prorate Cancel: Full Discount
4. Define balance impacts for the discount step, specifying the following values:
   - **Grant/Consume: TotalCharges**
   - **Applied To: Event Owner**
   - **Percentage: 100**
   - **Base Expression: StepC**
   - **Grant: Current Cycle** (under Resource Validity Period)
5. Create the discount trigger:
   a. From the Pipeline Toolbox, select **Discount/ChargeShare Trigger**. The Discount/ChargeShare Trigger box opens.
   b. Specify the following values:

      Condition Expression: 1.0

      Condition operator: Greater than

      Condition Value: 0
6. Create the discount model:
   a. From the Pipeline Toolbox, select **Discount Model**. The Discount Model window opens.
   b. Specify the names of the discount version, triggers, and rules you defined in steps 1 through 5.
   c. Select the appropriate discount setting in relation to the other discounts you offer.
7. Create the discount share object, and map the Delayed Session Event to the model created in step 6.

Creating a Billing-Time Discount

To create a billing-time discount for a service based on the contract days counter:

1. Create a discount master using the default values.
2. Create a discount rule:
   a. From the Pipeline Toolbox, select **Discount/ChargeShare Rule**.
   b. Select the following values:

      Drum Expression: BAL(CDCD *resource_ID***)**

      Rule Type: Tier

      Drum Type: Quantity
3. Create a discount step:
   - **Threshold From: 1**
   - **Threshold To: 11**
   - **Prorate Purchase: Full Discount**
   - **Prorate Cancel: Full Discount**
4. Define balance impacts for the discount step, specifying the following values:
   - **Grant/Consume: Currency**
   - **Applied To: Event Owner**
   - **Percentage:** type the percentage of discount

     For example, entering **-10** specifies a 10% discount to be subtracted from the total charge.
   - **Base Expression: BAL(TotalCharges)**
   - **Grant: Current Cycle** (under Resource Validity Period)
5. Create steps for all the discounts rates you want to offer.

   For example:
   - **11-30** for 15% off
   - **31-100** for 20% off
   - **101-MAX** for 25% off
6. Create the discount trigger:
   a. From the Pipeline Toolbox, select **Discount/ChargeShare Trigger**. The Discount/ChargeShare Trigger dialog box opens.
   b. Specify the following values:

      Condition Expression: 1.0

      Condition operator: Greater than

      Condition Value: 0
7. Create the discount model:
   a. Specify the names of the discount version, triggers, and rules you defined in steps 1 through 6.
   b. Select the appropriate discount setting in relation to the other discounts you offer.

Associate this discount object with a deal at the subscription service level, the deal with a plan, and so on.

Specifying whether to Count the Days on which Subscription Status Changes

The timestamp for creation, activation, inactivation, and closure for a subscription service is recorded as midnight. You can configure BRM to include or exclude the days on which the subscription service status changes. The default is to include the days.

1. Open the Connection Manager (CM) **pin.conf** file in *BRM_Home*/**sys/cm**.
2. Uncomment the following lines in the file and, if necessary, change their value: **1** to include the day of the status change or **0** to exclude the day.
   - The day a subscription is created:

     ```
     - fm_subscription    cdc_line_create_day_include 1
     ```

   - The day a subscription changes status from inactive to active or from closed to active:

     ```
     - fm_subscription    time_stamp_cdc 1
     ```

   - The day a subscription is canceled or suspended:

     ```
     - fm_subscription     cdc_line_cancel_day_include 1
     ```

3. Stop and restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

For general information about editing the configuration file, see "Using Configuration Files to Connect and Configure Components" in *BRM System Administrator's Guide*.

Setting Up Discounts Based on Monthly Fees and Usage

Important: Support for discounts based on monthly fees and usage is included in Advanced Discounting Manager; an optional feature that requires a separate license.

To enable discounts based on the monthly fees and usage of all accounts in an account group, you associate the parent service of the discount share group with the aggregation counter, and the child service with the *copy* of the aggregation counter and the child-level aggregation counter.

To set up discounts based on the monthly fees and usage, perform the following tasks:

- Configuring BRM to Track Monthly Fees and Usage
- Configuring the NET_EM Module
- Set up the required discounts in Pricing Center:
  - Creating a Usage Discount to Aggregate All Cycle Fees to the Parent Service Counter
  - Creating a Billing-Time Discount to Copy the Parent Service Counter to the Child Service Counter
  - Creating a Usage Discount to Aggregate Monthly Fee and Usage on the Child Service Level
  - Creating a Billing-Time Discount for Discounting

These procedures are summarized in this section and described fully in "Grouping Discount Components into Discount Models".

To be able to apply discounts based on an account group's monthly fees and usage, you must also perform the following tasks:

1. Create a plan that includes subscription services.

   See "Managing Customers' Subscription-Level Services" in *BRM Managing Customers*.

2. Create an account hierarchy in a customer management application, such as Customer Center.

   See "Creating and Managing Hierarchical Account Groups" in *BRM Managing Accounts Receivable*.

3. Purchase plans for the parent and child accounts.

4. Create a discount sharing group in a customer management application, such as Customer Center.

   See "Create Discount Sharing Groups".

Configuring BRM to Track Monthly Fees and Usage

To use discounts based on monthly fees and usage, you need to enable BRM to track the total monthly fees and usage. See "Setting Up Discounts Based on Monthly Fees and Usage".

Uncomment the **non_currency_mfuc** entry in the Connection Manager (CM) configuration file.

1. Open the Connection Manager (CM) **pin.conf** file in *BRM_Home***/sys/cm**.

2. Uncomment the following line in the file:

   ```
   - fm_subscription non_currency_mfuc 1000097
   ```

3. Stop and restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Configuring the NET_EM Module

To correctly copy the value of the MFUC to the CMFUC, the number of real-time pipelines must be greater than or equal to the number of threads specified for the **pin_billd** application in its configuration (**pin.conf**) file.

If the NET_EM module does not have enough pipelines, follow these steps to configure the correct number of pipelines:

1. Use the **wirelessRealtime.reg** registry file to configure the NET_EM module as follows:

   a. In the **DiscountOpcode** and **ReRatingOpcode** sections, change the **NumberofRTPipelines** entries from **2** to **7**.

   b. Change the number of threads in the **ThreadPool** section from **2** to **14**.

   c. Increase the number of discounting pipeline blocks in the file from **2** to **7**. The pipeline names should be consecutively numbered from **DiscountPipeline0** to **DiscountPipeline6**.

   d. Increase the number of rerating pipeline blocks in the file from **2** to **7**. The pipeline names should be consecutively numbered from **ReratingPipeline0** to **ReratingPipeline6**.

2. Use these SQL statements to modify the Pipeline Manager database:

```
INSERT INTO IFW_PIPELINE ( PIPELINE, NAME, EDRC_DESC, ENTRYBY, MODIFIED, RECVER ) VALUES
('ReratingPipeline2', 'Wireless Sample Re-rating Realtime', 'ALL_RATE', 0 ,1, 1);

INSERT INTO IFW_PIPELINE ( PIPELINE, NAME, EDRC_DESC, ENTRYBY, MODIFIED, RECVER ) VALUES
('ReratingPipeline3', 'Wireless Sample Re-rating Realtime', 'ALL_RATE', 0 ,1, 1);

INSERT INTO IFW_PIPELINE ( PIPELINE, NAME, EDRC_DESC, ENTRYBY, MODIFIED, RECVER ) VALUES
('ReratingPipeline4', 'Wireless Sample Re-rating Realtime', 'ALL_RATE', 0 ,1, 1);

INSERT INTO IFW_PIPELINE ( PIPELINE, NAME, EDRC_DESC, ENTRYBY, MODIFIED, RECVER ) VALUES
('ReratingPipeline5', 'Wireless Sample Re-rating Realtime', 'ALL_RATE', 0 ,1, 1);

INSERT INTO IFW_PIPELINE ( PIPELINE, NAME, EDRC_DESC, ENTRYBY, MODIFIED, RECVER ) VALUES
('ReratingPipeline6', 'Wireless Sample Re-rating Realtime', 'ALL_RATE', 0 ,1, 1);
```

For more information, see "NET_EM" and "Configuring the NET_EM Module for Real-Time Processing" in *BRM System Administrator's Guide*.

Creating a Usage Discount to Aggregate All Cycle Fees to the Parent Service Counter

To set up the monthly fee and usage counter to aggregate all cycle fees:

1. Create a discount master, accepting all of the default values in the Discount/ChargeShare Detail dialog box.
2. In the Discount/ChargeShare Rule dialog box, enter these values:
   - For **Code**, type the name.
   - For **Drum Expression**, select **TotalC**.
   - For **Rule Type**, select **Threshold**.
   - For **Drum Type**, select **Charge**.
3. Create a discount step using these values:
   - **Threshold From: 0**
   - **Threshold To: 0**
   - **Prorate Purchase: Full Discount**
   - **Prorate Cancel: Full Discount**
4. In the Discount/ChargeShare Step dialog box, click **Actions** to open the Discount/ChargeShare Balance Impact dialog box, and enter these values:
   - For **Grant/Consume**, select **1000097 MFUC**. This is the resource for the parent service counter.
   - For **Applied To**, select **ChargeShare Owner**.
   - For **Percentage**, type **100**.
   - For **Base Expression**, type: **StepC**.
   - For **Grant: Current Cycle**.
5. Create the discount trigger.

   For the discount sharing group, use default values so that the charge packet is always considered. From the Pipeline Toolbox, select **Discount/ChargeShare Trigger**. When, the Discount/ChargeShare Trigger box opens, enter these values.
   - **Condition Expression: 1.0**

- **Condition operator: Greater than**
- **Condition Value: 0**

6. Create a discount model for the model, the trigger, and the rule that you created in steps 1 through 5.
7. Create the discount object, including:
   a. Mapping a cycle forward discount event to the discount model to aggregate the monthly cycle forward fees in MFUC.
   b. Mapping the session event to the discount model to aggregate the usage charges in MFUC.

Creating a Billing-Time Discount to Copy the Parent Service Counter to the Child Service Counter

This procedure copies the parent monthly fee and usage counter (MFUC) to the child monthly fee and usage counter (CMFUC).

1. Create a discount master, accepting all of the default values in the Discount/ChargeShare Detail dialog box.
2. Create a discount rule using these values:
   - **Drum Expression: BAL(1000097)**
   - **Rule Type: Threshold**
   - **Drum Type: Quantity**
3. Create a discount step using these values:
   a. In the **Discount/ChargeShare Step** tab of the Discount/ChargeShare Rule dialog box, click **New**. The Discount/ChargeShare Step dialog box opens.
   b. Select the following values:

      Threshold From: 0

      Threshold To: 0

      Prorate Purchase: Full Discount

      Prorate Cancel: Full Discount
4. Define balance impacts for the discount step, specifying the following values (this copies the aggregation of MFUC to CMFUC at the child service level):
   - **Grant/Consume: CMFUC**
   - **Applied To: Event Owner**
   - **Percentage: 100%**
   - **Base Expression: BAL (MFUC)**
   - **Grant: Current cycle** (under Resource Validity Period)
5. Create a discount trigger using these values:
   - **Condition Expression: 1.0**
   - **Condition operator: Greater than**
   - **Condition Value: 0**
6. Complete a discount model for the entries the model, the trigger, and the rule that you created in steps 1 through 5.

7. Create the discount object, including a billing-time discount for the discount model to copy the MFUC (the parent service-level counter) to CMFUC (the child service-level counter).

Creating a Usage Discount to Aggregate Monthly Fee and Usage on the Child Service Level

To create a real-time discount that stores the monthly fee and usage charges for all the services of each child or user account:

1. Create a discount master, accepting all of the default values in the Discount/ChargeShare Detail dialog box.
2. In the Discount/ChargeShare Rule dialog box, enter these values:
   - For **Drum Expression**, select: **TotalC**.
   - For **Type**, select either **Threshold** or **Tier**.

     This setting determines whether the DRUM can overlap a threshold or must fall entirely within it.
   - For **Drum Type**, select **Charge**.

     This setting determines whether the DRUM and the threshold values are based on monetary charges or on quantities, such as bytes.
3. Create a discount step using these values:
   - **Threshold From: 0**
   - **Threshold To: 0**
   - **Prorate Purchase: Full Discount**
   - **Prorate Cancel: Full Discount**
4. In the Discount/ChargeShare Step dialog box, click **Actions** to open the Discount/ChargeShare Balance Impact dialog box, and enter these values:
   - For **Grant/Consume**, select **CHAGC**.
   - For **Applied To**, select **Event Owner**.
   - For **Percentage**, type **100**.
   - For **Base Expression**, type: **StepC**.
   - For **Grant: Current Cycle**.
5. Create a discount trigger using these values:
   - **Condition Expression: 1.0**
   - **Condition operator: Greater than**
   - **Condition Value: 0**
6. Create a discount model for the model, the trigger, and the rule that you created in steps 1 through 5.
7. Create the discount object, including:
   a. Mapping a cycle forward event to the discount model to aggregate cycle fees to the child service level counter.
   b. Mapping a usage event to the discount model to aggregate usage fees to child service level counter.

Creating a Billing-Time Discount for Discounting

The discounts created in the previous procedures are for non-currency resources. This discount can be purchased by discount group users.

1. Create a discount master.
2. In the Discount/ChargeShare Detail dialog box, enter the date range and other criteria for granting the discount.
3. Create a discount rule using these values:
   - **Drum Expression: BAL(1000098)** (The resource for CMFUC.)
   - **Rule Type: Threshold**
   - **Drum Type: Charge**
4. Create a discount step using these values:
   - **Threshold From: 0**
   - **Threshold To:** Type the upper limit, for example, **15000**.
   - Prorate Purchase: Full Discount
   - Prorate Cancel: Full Discount
5. Define balance impacts for the discount step, specifying these values:
   - **Grant/Consume:** select the currency you want to associate with the discount, for example: **840, US Dollar**.
   - **Applied To: Event Owner**
   - **Percentage:** Type the discount for this threshold range.
   - **Base Expression: TotalC**
   - **Resource Validity Period: Available resource** (under Consume)
6. Create a discount trigger using these values:
   - **Condition Expression: 1.0**
   - **Condition operator: Greater than**
   - **Condition Value: 0**
7. Complete a discount model for the entries the model, the trigger, and the rule that you created in steps 1 through 6.
8. Create the discount object, including a billing-time discount for the discount model to grant discounts based on the discount steps.
9. Bundle this purchasable discount object with a product in a deal, created at the service level.
10. Add this deal to a plan, which can be purchased by subscriber accounts.

    For more information about plans, see "Strategies for Creating Plans" in *BRM Setting Up Pricing and Rating*.

Setting Up Discounts Based on Query Values

You can create discounts based on query values. See "About Discounts" based on query values for introductory information. You use iScripts to implement query-based discounts. The iScripts retrieve data, either from the BRM database via opcode calls or from the EDR.

Note: Although discounts based on query values are billing-time discounts, they are configured differently from normal billing-time discounts. They do not require the use of aggregation counters.

To use iScripts for discounts, you specify one or more iScript file names in the registry for DAT_Discount. Each iScript file can include one or more functions that return values to be used in the discount.

To use iScript functions in discounts, you perform the following tasks:

1. Write one or more iScripts that retrieve the data you want to use for discounting. See "Writing iScripts for Query-Based Discounts".

   Note: BRM includes an iScript to support most-called-number discounts. See "Discounts Based on Most-Called Numbers".

2. Define the iScript in the DAT_Discount module registry. See "Configuring the DAT_Discount Module for Query-Based Discounts".
3. If the discount will be implemented in a batch pipeline, ensure that the pipeline includes a connection pool. See "DAT_ConnectionPool".
4. If necessary, modify the pipeline input mapping and EDR container to support the data returned by your iScript functions. See "Setting Up Pipeline Price List Data" in *BRM Setting Up Pricing and Rating*.

   Note: The data returned by the sample most-called-number iScript has already been mapped into the EDR container.

5. Configure a discount model that includes one or more expressions with **EVAL** tokens referring to the iScript. See "Understanding the EVAL Token" and "Example Discount Configuration".
6. Define a discount object that maps the discount model to a billing-time discount event and the appropriate service. See "Example Discount Configuration".

BRM includes a sample iScript for most-called-number discounts based on call duration, total cost, and total occurrences. You can use this iScript as is or modify it for your business needs. See "Discounts Based on Most-Called Numbers" for more information.

BRM also includes a sample iScript for discounts based on past usage. See "Example of Implementing Discounts Based on Past Usage".

Writing iScripts for Query-Based Discounts

An iScript for a query-based discount must include one or more named functions that can be referenced by an **EVAL** token in a discount expression. These named functions can retrieve data in two ways:

- **With opcode calls**. When the pipeline processes an EDR, the iScript calls one or more opcodes that retrieve and aggregate data from the BRM database. The data returned by the opcodes can be used directly or mapped into the EDR container to be retrieved by another **EVAL** token. You can use standard opcodes, customized policy opcodes, or newly-created custom opcodes to retrieve the information used

for discounting. You use iScript flist extension functions to manipulate data sent to and returned from opcodes. See "Calling Opcodes from iScript Functions".

- **From the EDR**. The iScript retrieves data that has been added to the EDR by customized opcodes. For example, you can modify a policy opcode that is triggered during billing-time discount processing to search for or aggregate data. The data returned by the opcode can be mapped into the EDR container. The iScript retrieves this data and uses it for discount calculations. You use standard iScript functions to retrieve the EDR data. See "Creating iScripts and iRules" in *BRM Developer's Guide*.

To create iScripts for query-based discounts, you can use the standard iScript functions, except for those described in iScript control structures. These special functions are not supported by the DAT_Discount module. See "Creating iScripts and iRules" in *BRM Developer's Guide* for more information.

In addition to the standard iScript functions, you use extension functions to manipulate flist data for query-based discounts. See "Loading External Modules for Extension Functions".

You can include query functions in multiple iScript files configured in a single DAT_Discount module. See "Configuring the DAT_Discount Module for Query-Based Discounts" for information about including iScript files in the module registry.

If you modify an iScript after the pipeline has been started, you can use a semaphore command with the DAT_Discount module to reload the iScript. See "DAT_Discount".

Loading External Modules for Extension Functions

The iScripts that you write for query-based discounts require the use of extension functions in two external modules that you load separately:

- EXT_FList, which includes functions for manipulating flists. See "Flist Manipulation Functions" in *BRM Developer's Reference*.
- EXT_Opcode, which includes functions for calling opcodes and opening a CM connection from a batch pipeline. See "Opcode Calling Functions" in *BRM Developer's Reference*.

To use functions from these module in an iScript, you must explicitly load them by including the following statements in the first lines of the iScript file:

```
Use EXT_FList;
Use EXT_Opcode;
```

Opening a CM Connection

Before an opcode can be called from an iScript, there must be a connection through which the opcode is called and data is returned. How this connection is established depends on whether the discounting module is running in a batch or real-time pipeline:

- In a real-time pipeline, a CM context exists already in the EDR object. This pre-existing connection is used automatically for opcode calls. You use the **opcodeExecuteInternal** function to call opcodes in this case.
- In a batch pipeline, you must use the **opcodeGetConnection** function to establish a CM connection. You can then use **opcodeExecute** to call opcodes. The iScript must call **opcodeGetConnection** before calling **opcodeExecute**.

  The **opcodeGetConnection** function obtains a connection from the connection pool you specify as a parameter. The connection pool must already have been

configured in the pipeline registry. See "DAT_ConnectionPool".

Calling Opcodes from iScript Functions

You use the **opcodeExecute** and **opcodeExecuteInternal** functions to call opcodes from iScripts. These functions do the same thing and have the same syntax: you specify the opcode number and any necessary flags in the function's parameters. You use them in different situations, however:

- You use **opcodeExecuteInternal** in iScripts with discounting modules in real-time pipelines. The function uses the already existing context to communicate with the CM.
- You use **opcodeExecute** in iScripts with discounting modules in batch pipelines. You must call the **opcodeGetConnection** function first to open a CM connection. See "Opening a CM Connection".

When you execute opcodes from iScripts, you must pass an input flist in the call to the opcode. To build the opcode flist, you use the flist extension iScript functions. For more information about the flist extension functions, see the following:

- "Creating iScripts and iRules" in *BRM Developer's Guide*
- "Flist Manipulation Functions" in *BRM Developer's Reference*
- "Opcode Calling Functions" in *BRM Developer's Reference*.

For example, to search for events for a specific account, you can create an flist for the PCM_OP_SEARCH opcode by using flist extension functions to set the search template and arguments.

The output flist returned by the opcode call can be accessed by the iScript using the flist extension functions. The return of the function that includes the opcode call should always be a decimal value.

See "iScript Example" for an example of calling an opcode.

Handling Errors in Opcodes Called from iScripts

If the opcode called returns an error, the error information can be accessed from the error buffer by using the **fListGetErrorText** function. The iScript function that included the opcode call should return INVALID_DECIMAL.

See "iScript Example" for sample error-handling code.

iScript Example

In this example, the **getLastSixMonthsCharge** function returns the total cost of calls during the last six months. The iScript includes examples of error handling and using the flist extension functions to manipulate flist data.

This iScript is for use with a real-time pipeline, so it does not include the **opcodeGetConnection** function to establish a CM connection.

See "Example of Implementing Discounts Based on Past Usage" for information about using this iScript.

```
use EXT_FList;
use EXT_Opcode;

//Global Variables

Bool DEBUG = false;
```

```
function Decimal getLastSixMonthsCharge
{
  logPipeline("getLastSixMonthsCharge() " +"\n");

  String s1 = "";
  String s2 = "";
  fListCreateNew();

  //Add the poid, flags and template to the flist
  fListSetPoid( "PIN_FLD_POID", "0.0.0.1 /search -1 0" );
  fListSetLong( "PIN_FLD_FLAGS", 256 );
  fListSetString( "PIN_FLD_TEMPLATE", "Select X from /item where F1 = V1 AND F2 >= V2 " );

  //Arg1 is the service object.
  String svcID = edrString(DETAIL.CUST_A.DL.DISCOUNT_OWNER_ID, 0, 0);
  String svcType = edrString(DETAIL.CUST_A.DL.DISCOUNT_OWNER_TYPE,0, 0);
  if(DEBUG == true){
     logPipeline("Service Type:\n" +svcType +" "+svcID +"\n");
  }
  //The service Id is in the format 1_12345. Need to separate the
  //database and the poid id.
  String ListArray[];
  Long nbElem = strSplit( ListArray, svcID, "_" );

  String svcObjStr = "0.0.0."+ListArray[0]+" "+svcType+" "+ListArray[1]+" 0";
  fListPushElem( "PIN_FLD_ARGS", 1 );
  fListSetPoid( "PIN_FLD_SERVICE_OBJ", svcObjStr);
  fListPopElem();

  //Arg2 is the effective_t. This is current date - 6 Months
  Date now = sysdate();
  Date past = dateAdd(now,0,-6,0,0);
  fListPushElem( "PIN_FLD_ARGS", 2 );
  fListSetDate( "PIN_FLD_EFFECTIVE_T", past );
  fListPopElem();

  //set the results array
  Decimal iTotal = 0.0;
  fListPushElem( "PIN_FLD_RESULTS",  0  );
  fListSetDecimal( "PIN_FLD_ITEM_TOTAL", iTotal);
  fListPopElem();

  if(DEBUG == true) {
     String fStr = fListToString();
     logPipeline("Printing the INPUT FLIST:\n");
     logPipeline(fStr);
  }

  //Execute the opcode PCM_OP_SEARCH
  if(opcodeExecuteInternal(7, 0) == false )
  {
    // Opcode failed
    fListGetErrorText( s1, s2 );
    logPipeline("PCM_OP_SEARCH failed: " + s1 + " " + s2 + "\n");
    return INVALID_DECIMAL;
  }

  if(DEBUG == true) {
     fListGetErrorText( s1, s2 );
```

```
    logPipeline("PCM_OP_SEARCH done: " + s1 + " " + s2 + "\n");

    String fRStr = fListToString();
    logPipeline("Printing the RETURN FLIST:\n");
    logPipeline(fRStr);
  }

  //Loop through the results and aggregate the item total
  Long resultCount = fListNumElem("PIN_FLD_RESULTS");
  Long i;
  Decimal tempTotal = 0.0;
  Decimal itemTotal = 0.0;

  for ( i = 0; i < resultCount; i = i + 1 )
  {
    tempTotal = fListDecimal ("PIN_FLD_RESULTS", i, "PIN_FLD_ITEM_TOTAL");
    itemTotal = itemTotal + tempTotal;

  }

  if(DEBUG == true) {
     logPipeline("getLastSixMonthsCharge() return:\n" +decimalToStr(itemTotal) +"\n");
  }
  return itemTotal;
}
```

Configuring the DAT_Discount Module for Query-Based Discounts

To configure the DAT_Discount module for query-based discounts, you specify the names of one or more iScript files in the module registry. These iScript files contain the functions for queries triggered by **EVAL** tokens in discount models.

When the module is initialized during pipeline startup, the iScript files defined in the registry are passed to the iScript interpreter for compilation. You can use a semaphore command to reload the iScript if you modify a file after pipeline startup. See "DAT_ Discount" for more information.

You specify the iScript file names in the **EvalScriptFiles** registry entry.

The following registry example includes two iScript files:

```
DiscountModelDataModule
{
ModuleName = DAT_Discount
   Module
   {
      ...
      #Customizable iScript files supporting EVAL function
      EvalScriptFiles
      {
         ScriptFile1=./iScript/CustomEval1.isc
         ScriptFile2=./iScript/CustomEval2.isc
      }
      ...
   }
...
}
```

Example Discount Configuration

This section includes an example of using Pricing Center to configure a query-based billing-time discount for international calls to the U.S. This example is based on the use of an **EVAL** token to invoke an iScript that returns the cost of calls to the U.S.

For detailed information about configuring discounts in Pricing Center, see Pricing Center Help and "Getting Started".

1. Create a discount master, leaving all fields at their default values.
2. Create a discount rule with the following characteristics:
   - DRUM Expression: **1**
   - Rule Type: **Tiered**
   - DRUM Type: **Charge**
3. Create a discount step for the new rule with the following characteristics:
   - Threshold Range: **0** to **infinity**
   - Prorate Purchase: **Prorate Discount**
   - Prorate Cancel: **Prorate Discount**
4. Create a balance impact for the new step with the following characteristics:
   - Impact/Consume: **978 (**Euro)
   - Applied To: **Event Owner**
   - Percentage: **-10%**
   - Base Expression: **EVAL(getInterUSCost)**
   - Resource Validity Period: **Current Cycle**
   - Leave other fields at their default values
5. Create a discount trigger.
6. Create a discount condition for the new trigger with the following characteristics:
   - Condition Expression: **1.0**
   - Condition Operator: **Greater than**
   - Condition Value: **0**
7. Create a discount model.
8. Create a discount model version for the new model with the following characteristics:
   - Version: **1**
   - Valid from: enter a validity date
   - Status: **Active**
9. Create a discount model configuration for the new version with the following characteristics:
   - Discount Version: select the version you created
   - Discount Trigger: select the trigger you created
   - Discount Rule: select the rule you created
   - Multiple discounts per event: **Parallel**

10. Create a discount object with the following characteristics:
    - Applies to: **/service/telco/gsm/telephony**
    - Map an Event to a Discount Model: select the **Billing Time Discount** event and the discount model you created
    - Leave all other fields at their default values
11. Include the discount in a deal and plan that applies to **/service/telco/gsm/telephony**.

Discounts Based on Most-Called Numbers

BRM includes iScripts and other components that enable you to implement discounts based on a subscriber's most-called numbers. Most-called-number discounts can be based on the total duration of calls, total charge for calls, or total times a number is dialed.

Implementing a most-called-number discount requires the use of the following components:

- The PCM_OP_RATE_POL_PRE_RATING policy opcode. This opcode searches for and aggregates most-called-number information from the BRM database.
- A sample provisioning tag definition that you can customize to define criteria for the type of most-called-number discount you want to use. This provisioning tag defines a **/profile/mostcalled** object that PCM_OP_RATE_POL_PRE_RATING uses to determine how to construct the list of most-called numbers.
- An iScript ("ISC_GetMostCalledInfo.isc") that retrieves data from the EDR that was returned by the customized policy opcode.
- The **discount_event.xml** file, which defines the flist-to-EDR container mapping for the INP_Realtime module.
- Sample pricing data in the **GSMPricePlan.ipl** price list file.

BRM also includes a sample iScript that you can use as the basis of discounts based on a subscriber's usage for the previous six months. See "Example of Implementing Discounts Based on Past Usage".

Most-Called-Number Discount Workflow

This section provides an overview of how BRM processes most-called-number discounts.

1. At billing time, if the account owns a most-called-number discount, Subscription Manager generates an event of type **/event/billing/cycle/discount/mostcalled**. Subscription Manager calls the PCM_OP_ACT_USAGE, which in turn calls the PCM_OP_RATE_POL_PRE_RATING policy opcode. The input to the opcodes includes the **/event/billing/cycle/discount/mostcalled** object.
2. PCM_OP_RATE_POL_PRE_RATING uses the POID of the **/purchased_discount** object to find the **/profile/mostcalled** object associated with the discount. This profile includes information about the criteria used to determine most-called numbers. It also includes information about how many numbers will be included in the list.
3. PCM_OP_RATE_POL_PRE_RATING uses the criteria in the **/profile/mostcalled** object to construct a search template that finds most-called-number information.

4. Based on the data returned by search, PCM_OP_RATE_POL_PRE_RATING creates a most-called-number list. The list includes four items of information:
   - A semicolon-delimited list of the most-called numbers. The length of the list depends on the contents of the PIN_FLD_COUNT field in the **/profile/mostcalled** object.
   - The total duration of all calls to the most-called numbers.
   - The total charges of all calls to the most-called numbers.
   - The total number of calls to the most-called numbers.
5. The opcode adds the data to the flist for the **/event/billing/cycle/discount/mostcalled** object. The data is stored in a substruct with fields corresponding to the four items of data returned by the opcode.
6. The flist is passed to the input module of the real-time discounting pipeline. The data is mapped into the following fields in the EDR container:
   - DETAIL.MOST_CALLED.LIST
   - DETAIL.MOST_CALLED.QUANTITY
   - DETAIL.MOST_CALLED.AMOUNT
   - DETAIL.MOST_CALLED.COUNT
7. When the EDR, now containing the most-called-number information, is processed by the discounting module, the **EVAL** token in the base expression of the discount model calls a function in the GetMostCalledInfo iScript to retrieve the total duration, cost, or number of calls.
8. The data retrieved by the specified function is substituted into the discount expression and becomes part of the discount calculation.

Implementing Most-Called-Number Discounts

You must complete the following tasks to implement most-called-number discounts using the customized components supplied with this feature:

1. If you plan to use the sample price plan (**GSMPricePlan.ipl**) to implement the discount, create the sample pricing data in the pipeline database.
2. Configure DAT_Discount in the real-time pipeline to use the "ISC_GetMostCalledInfo.isc" iScript file. See "Configuring the DAT_Discount Module for Query-Based Discounts".
3. Implement a MOST_CALLED provisioning tag. You must configure the provisioning tag to determine whether the list of most-called numbers is based on cost, duration, or occurrences. See "Using Provisioning Tags for Most-Called-Number Discounts".
4. In Pricing Center, configure a usage discount based on most-called numbers:
   a. Create a discount model that includes a base expression with an **EVAL** token. The token must refer to the function in "ISC_GetMostCalledInfo.isc" that corresponds to the type of most-called-number discount you configured in the MOST_CALLED provisioning tag.

      The **GSMPricePlan.ipl** sample price list file includes a most-called number discount. This file is located in the **Pricing Center/Sample_Price_Plans/Optional_Manager_Plans** directory.

b. Create a discount object that maps the discount model to the **/event/billing/cycle/discount/mostcalled** event and the appropriate service, such as **/service/telco/gsm/telephony**. Configure the discount object to include the MOST_CALLED provisioning tag.

5. Include the discount in a plan that can be purchased by a subscriber.

Using Provisioning Tags for Most-Called-Number Discounts

The most-called-number discounts use the MOST_CALLED provisioning tag to create profiles. These profiles determine the criteria by which the PCM_OP_RATE_POL_PRE_RATING policy opcode determines the list of most-called numbers.

To define the MOST_CALLED provisioning tag, you append its definition to the **pin_config_provisioning_tags.xml** file and load the file using the **load_config_provisioning_tags** utility. See "Working with Provisioning Tags" in *BRM Setting Up Pricing and Rating* for more information.

This feature includes a sample XML file, *BRM_Home***/sys/data/config/pin_config_provisioning_tags_mostcalled.xml.** The file includes a definition of a MOST_CALLED provisioning tag that you can edit. You add this definition to the **pin_config_provisioning_tags.xml** file.

When you define the MOST_CALLED provisioning tag, there are four options you can set in addition to the standard information required for all provisioning tags:

- The number of items in the most-called-number list. This value determines how many numbers are affected by the discount. For example, you can define the discount to apply to the five most-called numbers or the ten most-called numbers. The maximum is 22 numbers. You enter this as a value for the PIN_FLD_COUNT parameter.
- The impact category or categories to which the discount applies. For example, you can determine that the discount applies only to calls made on the Orange network. You enter the impact category as a value for the PIN_FLD_IMPACT_CATEGORY parameter. To enter multiple impact categories, separate values with a semicolon.
- The event type to consider when searching for data on most-called numbers. You enter the event type as a value for the PIN_FLD_EVENT_TYPE parameter. Only **/event/delayed/session/telco** is supported.
- The criterion by which the list of most-called numbers is calculated. You enter this option as a value for the PIN_FLD_CRITERION parameter. There are three possible values:
  - OCCURRENCES: The list is defined based on the number of calls.
  - DURATION: The list is based on call duration.
  - COST: The list is based on the cost of calls.

The following excerpt from the XML file shows a MOST_CALLED provisioning tag configured for a cost-based discount that applies to the Orange impact category, the five most-called numbers, and GSM calls.

```
<OpcodeParamElement>
   <OpcodeParamName>PIN_FLD_INHERITED_INFO.PIN_FLD_MOST_CALLED_INFO.PIN_FLD_COUNT
   </OpcodeParamName>
   <OpcodeParamValue>5</OpcodeParamValue>
</OpcodeParamElement>

<OpcodeParamElement>
   <OpcodeParamName>PIN_FLD_INHERITED_INFO.PIN_FLD_MOST_CALLED_INFO.PIN_FLD_IMPACT_CATEGORY
```

```
   </OpcodeParamName>
   <OpcodeParamValue>ORANGE</OpcodeParamValue>
</OpcodeParamElement>

<OpcodeParamElement>
   <OpcodeParamName>PIN_FLD_INHERITED_INFO.PIN_FLD_MOST_CALLED_INFO.PIN_FLD_CRITERION
   </OpcodeParamName>
   <OpcodeParamValue>COST</OpcodeParamValue>
</OpcodeParamElement>

<OpcodeParamElement>
   <OpcodeParamName>PIN_FLD_INHERITED_INFO.PIN_FLD_MOST_CALLED_INFO.PIN_FLD_EVENT_TYPE
   </OpcodeParamName>
   <OpcodeParamValue>/event/delayed/session/telco/gsm</OpcodeParamValue>
</OpcodeParamElement>
```

Example Most-Called-Number Discount

A 10% discount on most-called numbers is included in a sample price plan provided with BRM. To use the sample, open the **GSMPricePlan.ipl** file in Pricing Center. This file is located in the **Pricing Center/Sample_Price_Plans/Optional_Manager_Plans** directory.

The following example illustrates an additional most-called discount that you can implement. This example provides a 10% discount on the most-called numbers (based on duration) when the total duration of calls to those numbers is greater than 200 minutes.

The example configuration is similar to that of the **MostCalled GSM Discount** included in the sample price plan, except that it adds an **EVAL** token to the condition expression in the discount condition. This token invokes the **getMostCalledDuration** function to retrieve duration information so that it can be compared to the condition value.

This procedure assumes that you have created a provisioning tag called MOST_CALLED_DURATION that specifies DURATION in the PIN_FLD_CRITERION parameter. See "Using Provisioning Tags for Most-Called-Number Discounts".

1. Create a discount master, leaving all fields at their default values.
2. Create a discount rule with the following characteristics:
   - DRUM Expression: **1**
   - Rule Type: **Tiered**
   - DRUM Type: **Charge**
3. Create a discount step for the new rule with the following characteristics:
   - Threshold Range: **0** to **infinity**
   - Prorate Purchase: **Prorate Discount**
   - Prorate Cancel: **Prorate Discount**
4. Create a balance impact for the new step with the following characteristics:
   - Impact/Consume: **978** (Euro)
   - Applied To: **Event Owner**
   - Percentage: **-10%**
   - Base Expression: **EVAL(getMostCalledCost)**

- Resource Validity Period: **Current Cycle**
- Leave other fields at their default values

5. Create a discount trigger.
6. Create a discount condition for the new trigger with the following characteristics:
   - Condition Expression: **EVAL(getMostCalledDuration)**
   - Condition Operator: **Greater than**
   - Condition Value = **12000**
7. Create a discount model.
8. Create a discount model version for the new model with the following characteristics:
   - Version: **1**
   - Valid from: enter a validity date
   - Status: **Active**
9. Create a discount model configuration for the new version with the following characteristics:
   - Discount Version: select the version you created
   - Discount Trigger: select the trigger you created
   - Discount Rule: select the rule you created
   - Multiple discounts per event: **Parallel**
10. Create a discount object with the following characteristics:
    - Applies to: **/service/telco/gsm/telephony**
    - Map an Event to a Discount Model: select the **MostCalled Billing Time Discount** event and the discount model you created
    - Detail Discount Info tab: Select the MOST_CALLED_DURATION provisioning tag
    - Leave all other fields at their default values
11. Include the discount in a deal and plan that applies to **/service/telco/gsm/telephony**.

Example of Implementing Discounts Based on Past Usage

BRM includes a sample iScript file called **ISC_GetLastSixMonthCharg.isc** that you can use to provide discounts based on past usage. The iScript is intended to be used with a a billing-time discount that is processed in a real-time pipeline.

The iScript includes a function, **getLastSixMonthCharge**, that retrieves an account's total charges for a specific service during the previous six months. It creates a template for PCM_OP_SEARCH, which finds all the matching bill items. The function then aggregates the total charges and returns a decimal value. To implement a discount, you specify this function in an **EVAL** token.

The following example discount configuration uses the data returned by **getLastSixMonthCharge** as the basis of a 20% discount.

1. Create a discount master, leaving all fields at their default values.

2. Create a discount rule with the following characteristics:
   - DRUM Expression: **1**
   - Rule Type: **Tiered**
   - DRUM Type: **Charge**
3. Create a discount step for the new rule with the following characteristics:
   - Threshold Range: **0** to **infinity**
   - Prorate Purchase: **Prorate Discount**
   - Prorate Cancel: **Prorate Discount**
4. Create a balance impact for the new step with the following characteristics:
   - Impact/Consume: **978** (Euro)
   - Applied To: **Event Owner**
   - Percentage: **-20%**
   - Base Expression: **EVAL(getLastSixMonthCharge)**
   - Resource Validity Period: **Current Cycle**
   - Leave other fields at their default values
5. Create a discount trigger.
6. Create a discount condition for the new trigger with the following characteristics:
   - Condition Expression: **1.0**
   - Condition Operator: **Greater than**
   - Condition Value: **0**
7. Create a discount model.
8. Create a discount model version for the new model with the following characteristics:
   - Version: **1**
   - Valid from: enter a validity date
   - Status: **Active**
9. Create a discount model configuration for the new version with the following characteristics:
   - Discount Version: select the version you created
   - Discount Trigger: select the trigger you created
   - Discount Rule: select the rule you created
   - Multiple discounts per event: **Parallel**
10. Create a discount object with the following characteristics:
    - Applies to: **/service/telco/gsm/telephony**
    - Map an Event to a Discount Model: select the **Billing Time Discount** event and the discount model you created
    - Leave all other fields at their default values

11. Include the discount in a deal and plan that applies to **/service/telco/gsm/telephony**.

Sample iScripts for Query-Based Discounts

This section describes sample iScripts for query-based discounts.

ISC_GetMostCalledInfo.isc

This iScript includes three functions to retrieve total cost, total duration, and total occurrences values from the EDR container. These values are used to calculate most-called-number discounts. All three functions return decimal values.

You insert the return values of the functions into discounts by specifying the function names in the **EVAL** token. See "Understanding the EVAL Token". The three functions are:

- **getMostCalledCost**
- **getMostCalledDuration**
- **getMostCalledOccurrences**

You implement these functions by specifying the file name in the **EvalScriptFiles** registry entry for DAT_Discount. When the iScript is loaded and compiled, the functions are available to the **EVAL** token in discount expressions.

The file is located in the **iScriptLib/iScriptLib_Samples** directory.

Registry example

```
DiscountModelDataModule
{
ModuleName = DAT_Discount
   Module
   {
      ...
      #Customizable iScript files supporting EVAL function
      EvalScriptFiles
      {
         iScriptFile = ./iScriptLib/iScriptLib_Samples/ISC_GetMostCalledInfo.isc
      }
      ...
   }
...
}
```

ISC_GetLastSixMonthCharge.isc

This is a sample iScript that retrieves the total charges for a specific service in an account during the last six months. You can use this iScript to implement a billing-time discount on usage during this period.

The iScript includes the **getLastSixMonthsCharge** function. You insert the return values of this function into discounts by specifying the function name in the **EVAL** token. See "Understanding the EVAL Token".

The file is located in the **iScriptLib/iScriptLib_Samples** directory.

See "Example of Implementing Discounts Based on Past Usage" for more information.

Registry example

```
DiscountModelDataModule
```

```
{
ModuleName = DAT_Discount
   Module
   {
      ...
      #Customizable iScript files supporting EVAL function
      EvalScriptFiles
      {
         iScriptFile = ./iScriptLib/iScriptLib_Samples/ISC_GetLastSixMonthCharge()
      }
      ...
   }
...
}
```

Implementing Discount Validity Rules

When a discount is activated or canceled in the middle of a cycle, discount validity rules govern whether the discount is granted for the whole cycle, a part of the cycle, or no part of the cycle in which it is activated or canceled. For more information, see "About Applying Discounts Activated or Canceled in Mid-Cycle".

To implement discount validity rules, you must do the following:

- Set discount validity rules when you add discounts to plans by using Pricing Center. You specify a separate set of discount validity rules for usage discounts and for cycle discounts. See "About Discount Validity Rules" and "Creating Discounts".
- If you use a batch-rating pipeline, configure the batch pipeline for discount validity rules. See "Configuring the Batch Rating Pipeline for Discount Validity Rules".
- Set the proration rules in discount models so that they do not override discount validity rules. See "Discount Validity Rule Dependencies".
- Run a utility to change the status of expired discounts to canceled. See "Changing the Status of Discounts Canceled in Mid-Cycle".
- Rerate events to apply or back out usage discounts. For more information, see "Rerating Usage Events when Discount Validity Rules Apply".

Configuring the Batch Rating Pipeline for Discount Validity Rules

If you set the validity rule for usage discounts to grant the discount for the entire cycle in which it is activated (**Valid from middle of cycle: Full discount**), you must also set the **UseLatestProductAndDiscount** entry to **True** in the DAT_AccountBatch module registry. When set to **True**, the batch pipeline retrieves products and discounts based on their activation date instead of their purchase date. This ensures that delayed events are properly rated when their start date is earlier than the discount's purchase date.

See "DAT_AccountBatch".

Discount Validity Rule Dependencies

You set discount validity rules when you add discounts to plans in Pricing Center. These validity rules apply to usage discounts (discounts on usage events) and to cycle discounts (discounts on cycle events). You set these rules in the **Detailed Discount Info** tab of the Discount Attributes dialog box.

There are also cycle-event proration settings at the discount model level, in discount rules. These apply to cycle-fee discounts only.

To apply the cycle-fee discount validity rules that you set when adding discounts to plans, you must set the cycle event proration setting in the discount rule to **Prorate Discount**. When setting up discount rules, select **Prorate Discount** for the cycle event purchase and cancel proration settings in the **Discount/ChargeShare Step** tab of the Discount/ChargeShare Rule dialog box.

If you specify a value other than **Prorate Discount** in the discount rule, that value overrides the discount validity rules set at the discount level.

Note: To prorate fixed cycle-event discounts, before starting the real-time pipeline, you must set the **ProrateFixedDiscount** registry entry to **True** in the FCT_Discount module registry. See "FCT_Discount".

Managing Discount End Dates during Mid-Cycle Cancellations

When the discount validity rule is set to **Full Discount** and a discount is canceled in the middle of a cycle, BRM, by default, cancels the discount at the end of the accounting cycle. You can optionally configure BRM to cancel the discount immediately.

To specify how BRM sets the discount end date during mid-cycle cancellations, use the **CancelFullDiscountImmediate** parameter in the **subscription** instance of the **/config/business_params** object:

- When the parameter is disabled, BRM sets the discount end date to the accounting end date. This is the default.
- When the parameter is enabled, BRM sets the discount end date to the cancellation date.

To configure how BRM manages discount end dates during mid-cycle cancellations when the discount validity rule is set to **Full Discount**:

1. Go to *BRM_Home*/**sys/data/config**, which includes the support files used by the **pin_bus_params** utility.
2. Use the following command to create an editable XML file from the **business_params_subscription** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsSubscription bus_params_subscription.xml
   ```

   This command creates the XML file named **bus_params_subscription.xml.out** in your working directory. If you do not want this file in your working directory, specify the path as part of the file name
3. Search the file for the following line:

   ```
   <CancelFullDiscountImmediate>disabled</CancelFullDiscountImmediate>
   ```

4. Set the parameter to the appropriate value:
   - **enabled** to set the discount end date to the cancellation date.
   - **disabled** to set the discount end date to the accounting end date.

   BRM uses the XML in this file to overwrite the existing subscription instance of the **/config/business_params** object. If you delete or modify any other parameters in

the file, these changes affect the associated aspects of the BRM subscription configurations.

5. Save the file as **bus_params_subscription.xml**.
6. Use the following command to load this updated file into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_subscription.xml
   ```

7. Read the object with the **testnap** utility or Object Browser to verify that all fields are correct.
8. Stop and restart the CM.

Changing the Status of Discounts Canceled in Mid-Cycle

When a discount's validity rule is set to **Full discount** for discounts that are canceled in the middle of a cycle, BRM sets the discount to expire at the end of the cycle, but its status remains active.

To change the status of expired discounts from active to canceled, you must run the **pin_discount_cleanup** utility with the **-m** parameter.

You can run this utility daily or add it to the **pin_bill_day** utility to be run automatically. For more information, see "pin_discount_cleanup".

Rerating Usage Events when Discount Validity Rules Apply

For usage discounts (discounts that apply to usage events), you need to run rerating to apply or back out the discounts in these circumstances:

- When you grant a full discount for discounts that are:
  - Activated in the middle of a cycle
  - Activated and canceled in the same cycle

  Rerate the usage events for the accounting cycle in which the discount is activated. This ensures the discount is applied to events that occurred between the cycle start time and the discount purchase time. For example, if a discount is purchased on 1/15 and its activation time is set to 1/1, rerating ensures the discount is applied to events that occurred between 1/1 and 1/15.
- When you grant no discount for discounts that are:
  - Canceled in the middle of a cycle
  - Activated and canceled in the same cycle

  Rerate the usage events for the accounting cycle in which the discount is canceled. This ensures the discounts that were applied to events between the cycle start time and the cancellation time are backed out.

Validity rules for discounts on usage events also apply to discounts that aggregate usage for the purpose of applying billing-time discounts. Rerating usage events adjusts aggregation amounts accordingly.

For information on rerating, see "About Rerating Events" in *BRM Setting Up Pricing and Rating*.

Setting Up Discount Exclusion Rules

Exclusion rules establish a mutually exclusive relationship between discounts and also between a discount and a plan.

You define exclusion rules in Pricing Center. You can also view the discounts or plans that already have exclusion rules set.

For information on setting up exclusion rules for usage discounts, see "Setting Up Exclusion Rules for Usage Discounts".

Configuring and Defining Exclusion Rules

Before you can use exclusion rules, you must do the following:

- Configure the **/config/business_params** storable class object to support exclusion rules.
- Define an exclusion rule in Pricing Center.

Configuring Exclusion Rules

You can set up dependencies between two discounts or between a discount and a plan. Setting up an exclusion rule in Pricing Center enables the feature, but does not specify the dependency type. When you set up an exclusion rule in Pricing Center, the **/config/business_params** object is called by PCM_OP_SUBSCRIPTION_VALIDATE_DISCOUNT_DEPENDENCY.

Before you can set up exclusion rules, you need to set certain values in the /config/business_params storable class object. By default, exclusion rules are disabled in BRM. You can enable this feature by modifying a field in the **billing** instance of the **/config/business_params** object. You use the pin_bus_params utility to perform this task.

To enable and specify discount exclusion rules:

1. Use the following command to create an editable XML file from the **billing** instance of the **/config/business_params** objects:

   ```
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```

   This command creates the XML file named **bus_params_billing.xml.out** in your working directory. If you do not want this file in your working directory, specify the full path as part of the file name.

2. Search the XML file for following line:

   ```
   <ValidateDiscountDependency>disabled</ValidateDiscountDependency>
   ```

3. Change **disabled** to any of the following:
   - **discToDiscExcl:** Enable discount-to-discount exclusion rules.
   - **discToPlanExcl:** Enable discount-to-plan exclusion rules.
   - **enableBothExcl**: Enable both discount-to-discount and discount-to-plan exclusion rules.
   - **disableDiscToPlanExclAndNoPurTimeValidation:** Disable exclusion rules between price plans and discounts system wide. When this flag is set, at purchase time no dependency validations are performed.

- **enableBothExclAndNoPurTimeValidation:** Enable both discount-to-discount and discount-to-plan exclusion rules, and do not support dependency validations at purchase time.
- **returnOnFirstExcl**: Return the first mutually exclusive discounts or discount/price plan if BRM finds any such conflicts.

Caution: BRM uses the XML in this file to overwrite the existing **billing** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM billing configuration.

4. Save the file and rename it from **bus_params_billing.xml.out** to **bus_params_billing.xml**.
5. Use the following command to load the change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_billing.xml
   ```

   You should execute this command from the *BRM_Home*/**sys/data/config** directory, which includes support files used by the utility. To execute it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.
6. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   See "Using testnap" in *BRM Developer's Guide* for general instructions on using **testnap**. See "Reading Objects by Using Object Browser" in *BRM Developer's Guide* for information on how to use Object Browser.
7. Stop and restart the Connection Manager (CM). See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.
8. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see "pin_multidb" in *BRM System Administrator's Guide*.

Setting Up Exclusion Rules for Usage Discounts

After you have configured for standard exclusion rules (see "Configuring Exclusion Rules"), you must perform the following tasks to be able to apply exclusion rules for usage discounts:

- For the batch pipeline only, set the **ReadPlans** registry entry in the Customer Data module. See "Configuring the DAT_AccountBatch module".
- For both the real-time and batch pipelines, set the PortalConfigDataModule registry entry in the Discount Data module. See "DAT_Discount".

Configuring the DAT_AccountBatch module

To support discount-to-plan exclusion in a batch pipeline, *before starting the pipeline*, set the **ReadPlans** registry entry in the DAT_AccountBatch module to **True**.

For more information, see "DAT_AccountBatch".

Configuring the DAT_Discount module

To enable mutual exclusion in both the real-time and batch pipelines, configure the "DAT_Discount" module to retrieve business parameter settings from the "DAT_PortalConfig" module by using the **PortalConfigDataModule** entry.

For more information, see "Using Business Parameter Settings from the BRM Database" in *BRM System Administrator's Guide*.

Defining Exclusion Rules for Plans and Discounts

You can define exclusion rules between a plan and a discount in two ways:

- By defining an exclusion rule for a plan, which prohibits certain discounts from being owned by an account if the plan is also owned.
- By defining an exclusion rule for a discount, which prohibits any specified plans from being owned by an account if the discount is also owned.

Either way creates an exclusion rule between a discount and a plan. When you prohibit a plan from being used with a discount, that also prohibits any deals, services, or other discounts owned by that plan from being used with that discount. In the same way, when you prohibit a discount from being used by a plan, the exclusion rule you set also prohibits any discounts, services, or deals associated with that plan from being used with that discount. The method you use depends on your situation and whether you are most interested in controlling the plan or the discount.

You can also define an exclusion rule between two discounts, which excludes one from being applied when both are already owned or prohibits the two of them from being purchased in the same deal. See "Setting Up Discount Exclusion Rules".

Updating Discount Data

If you change discount pricing data, to make the changes take effect, you must refresh the Pipeline Manager data. To do this, use the **Reload** semaphore file entry for the DAT_Discount module. See "Reloading Data into a Pipeline Manager Module" in *BRM System Administrator's Guide* and "DAT_Discount".

When you reload discount data, Pipeline Manager retrieves discount model information from the Pipeline Manager database and account balance information from the BRM database. If any new non-currency resources are specified in the discount model, Pipeline Manager reloads the balance data for those resources. While Pipeline Manager is reloading discount data, event processing is suspended until reloading is complete.

12

Configuring Discounting Modules and Components

This chapter describes how to configure discounting in the Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager.

For information about Pipeline Manager discounting, see "About Discounts".

For information about discounting pipelines, see "Understanding the Discounting Architecture".

Configuring a Batch Discounting Pipeline

Batch discounting is typically performed in a separate discounting pipeline.

Configure the following data modules:

- **DAT_AccountBatch**. This module provides account data to Pipeline Manager. This module loads account data into memory when you start Pipeline Manager, and updates it when it is changed in the BRM database. See "DAT_AccountBatch" and "Adding Customer Balance Impact Data to EDRs" in *BRM Setting Up Pricing and Rating*.
- **DAT_BalanceBatch**. This module provides balance data for the FCT_Discount module when discounting is run in batch. This module loads balance data into memory when you start Pipeline Manager, and keeps the balance data synchronized between the Pipeline Manager database and the BRM database. See "DAT_BalanceBatch".
- **DAT_ModelSelector**. This module provides discount model selector data to the FCT_DiscountAnalysis module. See "DAT_ModelSelector".
- **DAT_Discount**. This module supplies discount model information to the discount function modules. See "DAT_Discount".

Configure the following function modules:

- **FCT_Discount**. This module performs the discount calculations and adds discounting data to the event data record (EDR). See "FCT_Discount".
- **FCT_DiscountAnalysis**. This module selects applicable discounts and prioritizes them. See "FCT_DiscountAnalysis".
- **FCT_Rounding**. This module rounds the balance impacts of discounting. Use the **Mode** registry entry to specify **Discounting**. See "FCT_Rounding".
- **FCT_ApplyBalance**. This module is used only for batch discounting. This module adds the discount balance impact to the EDR and updates the Pipeline Manager

memory. See "FCT_ApplyBalance".

About settIng the Validity of Resources Impacted by Discounts

The effective period of a granted resource can start when a subscriber first consumes the resource balance.

For more information, see "About Balance Impacts That Become Valid on First Usage" in *BRM Setting Up Pricing and Rating*.

The following modules are used to set the validity period of resources that start on first usage when they are impacted for the first time:

- **DAT_BalanceBatch**. This module calculates the resource validity period based on the EDR timestamp and initializes the validity period in memory.

  If the validity periods of all first-usage resources in the deal should be synchronized, DAT_BalanceBatch adds information about those resources to the EDR. (For information about synchronizing the first-usage validity of resources, see "About Synchronizing First-Usage Validity of Resources in Deals" in *BRM Setting Up Pricing and Rating*.)

- **FCT_ApplyBalance**. This module sets the validity period information in the EDR for all first-usage resources whose validity needs to be set. It then sends the entire EDR to an output stream.

  You specify the first-usage validity output stream in the FCT_ApplyBalance registry. See "FCT_ApplyBalance".

  You configure the output stream in the batch rating pipeline. See "Configuring Pipeline Output for First-Usage Products, Discounts, and Resources".

To set the validity period of first-usage resources sent to the output stream, you configure Universal Event (UE) Loader. See "About Updating Validity Period Information in the BRM Database".

Configuring Batch Discounting to Restrict Resource Validity End Time

When the validity period of a granted resource starts on first usage and ends relative to the start time, you can restrict the resource end time to ensure the resource balance cannot continue to be consumed after the product or discount expires.

For more information, see "About Restricting The End Time Of Granted Resources That Start On First Usage" in *BRM Setting Up Pricing and Rating*.

To restrict the validity end time of first-usage resources, configure "DAT_BalanceBatch" to use the **RestrictResourceValidityToOffer** business parameter setting from the BRM database.

You configure DAT_BalanceBatch to use business parameter settings from the BRM database by performing the following:

- Configuring the "DAT_PortalConfig" module in your registry file. This module must be listed before all other data modules in the registry file.
- Connecting the "DAT_BalanceBatch" module to DAT_PortalConfig by using the PortalConfigDataModule registry entry.

See "Using Business Parameter Settings From The BRM Database" in *BRM System Administrator's Guide*.

When you restrict resource validity end time, DAT_BalanceBatch sets the end time of the resource validity period to the end time of the product or discount that grants the resource if it is earlier than the resource validity end time.

Important: If you restrict resource validity for pipeline rating, you must also restrict resource validity for real-time rating. See "Configuring Real-Time Rating To Restrict Resource Validity End Time" in *BRM Setting Up Pricing and Rating*.

Calculating the Match Factor of Parallel and Sequential Discounts

The *match factor* in discounting is the percentage of usage that is discounted by a single discount when more than one discount is applied. The match factor is used with cascading discounts in which a discount can be applied only to the portion of usage that has not already been discounted.

For example, if an account owns two cascading discounts and the first one discounts 75% of the usage, the match factor is .75. The second discount, therefore, can be applied only to 25% of the usage.

By default, discounting does not calculate the match factor for parallel and sequential discounts. Should you need to calculate the match factor for parallel and sequential discounts, set the **AvoidMatchFactorCalculation** entry to **False** in the FCT_Discount module registry. See "FCT_Discount".

Configuring a Real-Time Discounting Pipeline

Important: Real-Time Discounting is an optional feature that requires a separate license.

You can use a real-time discounting pipeline to calculate discounts for events that are rated by real-time rating. This allows you to discount all events in real time, so customer discount balances are always current.

For information about real-time discounting pipelines, see "Real-Time Discounting Architecture".

To configure real-time discounting:

1. Configure a real-time discounting pipeline. See "Configuring a Real-Time Discounting Pipeline".
2. Configure the Input registry section. See "Configuring the Input Registry Section".
3. Configure the Connection Manager (CM) to send discounting requests to the NET_EM module. See "Configuring The NET_EM Module For Real-Time Processing" in *BRM System Administrator's Guide*.

Configuring a Real-Time Discounting Pipeline

Configure a real-time discounting pipeline that includes the following real-time discounting modules:

- **INP_Realtime**. This module handles flist-to-EDR format translation for a real-time pipeline. See "INP_Realtime".

  Use this entry for the **OpcodeMapping** entry:

```
OpcodeMapping = ./formatDesc/Formats/Realtime/discount_event.xml
```

- **OUT_Realtime**. This module handles EDR-to-flist format translation for a real-time pipeline. See "OUT_Realtime".
- **NET_EM**. This module provides an interface to the CM for the INP_Realtime and OUT_Realtime modules and an interface to the BRM database for the DAT_AccountRealtime and DAT_BalanceRealtime modules. See "NET_EM" and "Configuring The NET_EM Module For Real-Time Processing" in *BRM System Administrator's Guide*.
- **DAT_AccountRealtime**. This module provides account cycle data to the FCT_Discount module. The DAT_AccountRealtime module gets data from the BRM database by connecting with the NET_EM module. It does not store data in memory, so it does not load data when you start Pipeline Manager. See "DAT_AccountRealtime".
- **DAT_BalanceRealtime**. This module provides balance data for the FCT_Discount module for real-time discounting. The DAT_BalanceRealtime module gets data from the BRM database by connecting with the NET_EM module. It does not store data in memory, so it does not load data when you start Pipeline Manager. See "DAT_BalanceRealtime".
- **FCT_CreditLimitCheck**. This module is used only for real-time discounting. This module is used during the prepaid authorization process to determine whether event owners have enough resources in their account balance to use a requested service. For more information, see "FCT_CreditLimitCheck" and "How BRM Authorizes Users To Access Prepaid Services" in *BRM Telco Integration*.

In addition, configure the following standard discounting modules:

- **FCT_Discount**. This module performs the discount calculations and adds discounting data to the EDR. See "FCT_Discount".
- **FCT_DiscountAnalysis**. This module selects applicable discounts and prioritizes them. See "FCT_DiscountAnalysis".

Configuring the Input Registry Section

To manage real-time discounting efficiently, you must enable the Input registry **UnitsPerTransaction** entry. Use the following entry in the Input registry section:

```
Input
{
   UnitsPerTransaction = 1
   NoThread = true
...
```

For more information on the **UnitsPerTransaction** entry, see "Combining Multiple CDR Files Into One Transaction" in *BRM System Administrator's Guide*.

About Dumping Discount Model Information during Run Time

You can dump discount model configuration information during run time. Dumping the information writes it to a file or to the terminal. This is useful should you need to verify, compare, or provide discount configuration information during run time for troubleshooting purposes.

You can use the DAT_Discount and DAT_ModelSelector modules to write discount model configuration information. You can write discount configuration information for one or all discount models in your system by using the **DiscountModel** and

DataFileName semaphores. See "DAT_Discount" and "DAT_ModelSelector".

About Discount Transaction Management

To maintain data integrity in a pipeline, discounting uses transactional processing on two levels:

- Standard pipeline transactions, managed by the Transaction Manager (TAM). If a transaction fails, the input is stopped and all open transactions are rolled back. After rolling back the data, the input is restarted.

  Important: To enable transaction management, the **redoEnable** registry parameter of the TAM must be set to **True**. If the redo mechanism is not enabled, discounting will block any other sequential transactions.

- EDR transactions, managed by the discounting modules. An EDR might contain multiple charge packets that are manipulated by the FCT_Discount module. The module might find errors in some charge packets and not be able to finish processing the EDR. Therefore, all changes made in one EDR are logged. If there are no errors, the data is committed, otherwise the changes made by the module are rolled back and the EDR is not committed.

About Processing Balance Groups Locked by Other Transactions

During discount calculations in BRM, when discounting data is added to the event data record (EDR), the associated balance group is locked by the transaction. By default, there is a dependency between concurrent transactions involving the same balance group. If transaction A locks a balance group, then, by default, transaction B waits for transaction A to commit or roll back before it locks the same balance group.

You can enhance pipeline throughput performance in BRM by configuring the **IgnoreEDROnLock** entry in the FCT_Discount module to ignore an event data record, if the associated balance group object is locked by another transaction.

For more information on the **IgnoreEDROnLock** registry entry in the FCT_Discount module, see "FCT_Discount".

To configure the FCT_Discount module for:

- Batch operations, set the **IgnoreEDROnLock** registry entry to **True**. See the description for using registry files to control Pipeline Manager in *BRM System Administrator's Guide*.
- Run-time operations, set the **IgnoreEDROnLock** semaphore entry to **True**. See the description for using semaphore files to control Pipeline Manager in *BRM System Administrator's Guide*.

When the FCT_Discount module processes concurrent transactions involving the same locked balance group objects, if **IgnoreEDROnLock** entry is set to **True**, it places the ignored or rejected EDRs with the locked balance group in the **discountError** directory.

13

Discount Sharing Configuration Example

This chapter describes an example of configuring Oracle Communications Billing and Revenue Management (BRM) discounts for sharing free minutes among several accounts.

Before reading this document, you should read the following documents:

- About Discounts
- About Implementing Discounts

Sharing Free Minutes Among Several Accounts

This example shows how you set up discounts to share free minutes within a discount sharing group.

In this example, there is one sharing group owner account and two sharing group member accounts. The owner account shares its free seconds with the member accounts. The purpose of this configuration is to allow the member accounts to consume different quantities of the owner account's free seconds and to limit the amount of free seconds the members can consume.

The Scenario

The owner account is granted free minutes each month by purchasing a product that includes free seconds. These free seconds are shared with two member accounts. Each member account is granted a quota amount by purchasing a product that includes a quota resource. The quota resource granted is the maximum number of seconds the member is allowed to use from the owner's balance of free seconds. The quota is a counter that is increased whenever free seconds are used. The member accounts themselves are not granted free seconds in this scenario.

The free seconds and quota resource are granted as negative amounts that are increased with usage until the balance equals 0.

When a member account makes a call, the quantity used is deducted from the balance in the member's quota counter and from the owner account's free seconds balance. The free seconds are consumed until the owner's free seconds or the member's quota resource is depleted. Logically, the free seconds granted to the owner should be at least the sum of the quotas granted to the members.

Discount Elements

This discount sharing configuration example uses the following elements:

- Discount sharing groups to distribute the owner's free seconds to the member accounts.
- These resources:
  - **Free Seconds (Resource ID 1000095)**. This resource is granted to the discount sharing group owner account.
  - **Quota (Resource ID 1000036)**. This resource is a free minute counter that is granted to the member accounts in the discount sharing group. Each member can have a different quota amount. In this example, this resource is granted at the beginning of each cycle, and unused amounts are not rolled over.

    Note: Quota (resource ID 1000036) is not a default BRM resource. You configure this resource when you set up your price plans.

  - **Euro (Resource ID 978)**. Charges for calls impact this resource. This resource is discounted when the calls qualify for free seconds.
- Event balances to store available resources and the quantity of resource used.

  Event balances are required in this example for two reasons: Balances cannot be directly retrieved across accounts during the discounting process, and the impact of a discount that belongs to one account can't be applied to another account unless the discount is part of the discount sharing group. Not all discounts in this example are included in the discount sharing group.

You define event balances when you configure the discount balance impacts.

This example uses the following event balances:

- **EBal(109)** to store the discount sharing group owner's balance of free seconds.
- **EBal(110)** to store the difference between the owner's balance of free seconds and the member's free seconds quota.
- **EBal(111)** to store the number of free seconds used by the member account.

Three discounts: two purchased by the discount sharing group owner account, and one purchased by each member account in the discount sharing group:

- **Discounts purchased by owner account:**
  - **Owner Quota discount** - This discount essentially copies the balance of free seconds in the owner's account to a temporary event balance (EBal(109)). This event balance is needed by the members' discounts to calculate the quantity of usage that can be applied to the owner's free seconds.
  - **Owner Quota Free Seconds discount** - This discount deducts the free seconds that were consumed from the owner's balance. This discount doesn't provide any free seconds to the owner account because the balance update is based only on the amount of free seconds consumed by the member accounts.
- **Discount purchased by members:**
  - **Member Quota discount** - This discount updates the member account's quota balance and discounts the charges for free seconds used. It also stores the quantity of free seconds consumed in an event balance (EBal(111)). This event balance is used by the Owner Quota Free Seconds discount to update the owner's balance of free seconds.

Additional Discount Configuration

To set up this discount sharing scenario, you must perform the following tasks as well as configure the discounts:

- Set up a quota counter resource to grant the quota amount. See "Setting Up Resources" in *BRM Setting Up Pricing and Rating*.
- Create products that includes the free seconds and quotas.
- Create offer profiles for policy-driven charging. The offer profile name is the provisioning tag in the **/discount** object with which the offer profile is associated. See "Policy-Driven Charging" in *BRM Setting Up Pricing and Rating*.
- Create discount sharing groups that include the owner and member accounts. In this example, you create two discount sharing groups--one group for each member account. See "Create Discount Sharing Groups".

Defining the Discounts

This example shows how to set up the discounts in the order in which they are processed by the discount sharing groups:

1. Owner Quota Discount
2. Member Quota Discount
3. Owner Quota Free Seconds Discount

This section describes the values that you enter when creating the discount components in Pricing Center. Only those components and values relevant for this specific example are covered. Other, non-relevant values are not specified. For example, to filter EDRs for discounting, you might use the same discount master in all discount configurations. The discount master doesn't impact how discounts are shared in a discount sharing group so configuring the discount master isn't significant for this example.

Owner Quota Discount

The purpose of this discount is to simply record the owner's balance of free seconds in a temporary event balance. The owner's free seconds are granted at the beginning of each cycle by the product in the price plan. As the subscribers make calls, the free seconds balance changes so each time this discount is evaluated, a different number of free seconds is available.

This discount specifies that if the owner has a balance of free seconds (defined by the trigger), apply the discount balance impact (defined in the rule). The balance impact applies a free second to the event balance for every free second in the owner's balance.

This discount doesn't actually impact the account's balance because the balance impact is stored in a temporary event balance instead. You discount the owner's balance of free seconds in order to pass the quantity in the balance to the next discount (the Member Quota discount). The next discount uses the event balance to compare the owner's available free seconds with the amount the member is allowed to consume (the member's quota balance).

Discount Model Configuration: Owner Has Free Seconds to Share

Discount/Chargeshare Trigger = DTOQ (Discount Trigger for Owner Quota discount):

- Discount/Chargeshare Condition:
  - **Condition Expression = Bal(1000095)**

    The resource ID for free seconds is 1000095. The discount expression Bal(1000095) references the discount owner's free seconds balance.
  - **Condition Operator = Less Than**
  - **Condition Value = 0**

  This condition specifies that the discount owner has free seconds available (the balance of free seconds is less than 0). The operator is **Less than** because the free seconds in the plan are granted as a negative value that is increased with usage until the balance equals zero.

Discount/Chargeshare Rule = DROQ (Discount Rule for Owner Quota discount):

In this rule, you only need to record the owner's balance of free seconds into an event balance. This means you want the discount applied regardless of the amount of usage. Therefore, you need to make sure the DRUM expression always falls within the step's threshold. A sure way to do this is to make the threshold unlimited and the DRUM a finite quantity.

- DRUM:
  - **DRUM Expression = 1.0**

    This specifies that the minimum quantity to consider for discounting is one unit. The unit is non-currency because the DRUM type is Quantity. The resource to impact in the balance impact is free seconds, which means this DRUM evaluates to one second.
  - **DRUM Type = Quantity**
  - **Rule Type = Tiered**

    A tiered rule type means that the balance impact is applied when any part of one second (the DRUM) falls within the discount step threshold.
- Discount/Chargeshare Step:
  - **Threshold From = 0**
  - **Threshold To = Infinity**

    The threshold is unlimited so that any positive amount in the DRUM expression causes the balance impacts to be applied. In this case, 1.0 (the DRUM) falls between 0 and infinity so the following balance impact is applied.
- Discount/Chargeshare Balance Impact:

  This balance impact applies a free second to the event balance for every free second in the owner's free seconds balance.
  - **Impact/Consume = 1000095, Free Seconds**

    This balance impact is applied to the free seconds resource. Recall that the balance impact will be stored in an event balance, where this resource ID will be recorded.
  - **Applied To = Discount/Chargeshare Owner**

    This specifies that the free seconds balance that is impacted belongs to the discount owner account. However, because the balance impact will be stored

in an event balance and not applied to the account balance, the value in Applied To is not considered.

– **Amount = 1; Beat = 1; no proration**

The amount and beat are both 1. This specifies a one-to-one correlation between the discount amount and the quantity that is discounted, making them equal. For example, if the amount to discount is 20 minutes (1200 seconds), the discount amount is also 20 minutes.

Discounting a portion of one second is not desired so the beat is not prorated.

– **Base Expression = Bal(1000095)**

The base expression represents the amount to discount. The discount expression Bal(1000095) references the discount owner's current balance of free seconds that was granted by the owner's plan. (This balance changes as the free minutes are used.)

Because the amount and beat are equal, for every available second in the owner's balance, a second is added to the balance impact. In this way, you copy the available balance into the event balance.

– **Consume: Available resource**

The balance impact is stored in an event balance and is not applied to the account balance so it doesn't matter whether you specify **Consume** or **Impact**. **Consume** is the default.

– **Event Balance ID = 109**

This balance impact is stored in the temporary event balance with the ID 109. This event balance is referenced by using the expression EBal(109).

Discount Model Configuration

This model associates the trigger and rule that you just created.

Discount Model = DMOQ (Discount Model for Owner Quota discount):

- Discount Model Configurations:
  – **Trigger = DTOQ (Discount Trigger for Owner Quota discount)**
  – **Rule = DROQ (Discount Rule for Owner Quota discount)**
  – **Cascading = No**

  Because this model contains only one discount configuration, cascading is not relevant.

Member Quota Discount

The purpose of this discount is to update the member's quota balance, discount the member's charges for used free seconds, and store the number of free seconds used in another event balance (EBal(111)).

This discount uses the discount sharing group owner's balance of free seconds (stored in EBal(109) by the previous discount) to determine the maximum free seconds that can be consumed by the member.

The discount sharing group member cannot use more free seconds than its quota specifies, even though the group owner account may have additional seconds to share. It's also possible that the member's quota exceeds the number of free seconds available in the owner's balance. The number of free seconds allowed depends on which balance

is greater: the owner's or the member's. Therefore, this discount uses two discount model configurations to select the maximum number of free seconds allowed:

- If the discount sharing group owner's balance of free seconds (EBal(109)) exceeds the member's quota, the member's quota is the maximum allowed. The associated discount rule stores the quota balance in a temporary event balance (EBal(110)).
- If the member's quota balance exceeds the owner's balance of free seconds (EBal(109)), the owner's balance is the maximum allowed. The associated discount rule stores the owner's balance in a temporary event balance (EBal(110)).

Separate model configurations are used because the above scenarios are mutually exclusive.

A third discount model configuration is used to discount the usage charges and update the member's quota balance. The member's account balances are updated based on the maximum free seconds allowed, which was stored in EBal(110) by the previous configuration.

The number of free seconds consumed by the member is stored in another temporary event balance (EBal(111)). This event balance is required because this discount is not part of the discount sharing group and can't be used to update a balance in another account (the discount sharing group owner's balance of free seconds). EBal(111) will be used by the next discount to adjust the owner's balance of free seconds.

Discount Model Configuration 1: Owner's Balance of Free Seconds Exceeds Member's Quota

In this configuration, if the owner's balance of free seconds is greater than the member's quota, the quota balance is stored in a temporary event balance (EBal(110)). This balance represents the maximum amount of free seconds that can be consumed. This event balance is used in discount model configuration 3, which updates the member's balances based on the number of seconds used and the maximum free seconds allowed.

Discount/Chargeshare Trigger 1 = DTUseUQ (discount trigger to store member's quota balance):

- Discount/Chargeshare Condition:
  - **Condition Expression = EBal(109) - Bal(1000036)**
  - **Condition Operator = Less Than or Equal To**
  - **Condition Value = 0**

    Recall that the owner's balance of free seconds was stored in an event balance with the ID 109 by the Owner Quota discount, and that available minutes and quota are negative quantities.

    This condition specifies that the owner's balance of free seconds (EBal(109)) exceeds the member's available quota (Bal(1000036)). For example, if the owner has a balance of -300, and the member has a quota of -100, the condition is **(-300) - (-100) is less than or equal to 0**. In other words, **-200 is less than or equal to 0**.

Discount/Chargeshare Rule 1 = DRUseQ (discount rule to store member's quota balance):

In this rule, you only need to copy the member's available quota balance into an event balance. This means you want the discount applied regardless of the amount of usage. Therefore, you need to make sure the DRUM expression always falls within the step's threshold. A sure way to do this is to make the threshold unlimited and the DRUM a finite amount.

- DRUM:
  - **DRUM Expression = 1.0**
  - **DRUM Type = Quantity**

    This specifies that the minimum quantity to consider for discounting is one unit. The unit is non-currency because the DRUM type is Quantity. The resource to impact in the balance impact is the quota counter, which means this DRUM is equivalent to one second.
  - **Rule Type = Tiered**

    A tiered rule type means that the balance impact is applied when any part of one second (the DRUM) falls within the discount step's threshold.
- Discount/Chargeshare Step:
  - **Threshold From = 0**
  - **Threshold To = Infinity**

  The threshold is unlimited so that any positive amount in the DRUM expression causes the balance impacts to be applied.
- Discount/Chargeshare Balance Impact:

  In this balance impact, for every free second count in the current balance of the member's quota, a free second count is applied to the event balance. This balance impact is used to tell the next balance impact how many free seconds are available for consumption.
  - **Impact/Consume = 1000036 (quota counter)**

    This balance impact is applied to the quota resource. Recall that the balance impact will be stored in an event balance, where this resource ID will be recorded.
  - **Applied To = Event Owner**

    This specifies that the quota balance that is impacted belongs to the account that generated the usage. However, because the balance impact will be stored in an event balance and not applied to the account balance, the value in Applied To is not considered.
  - **Amount = 1; Beat = 1; no proration**

    The amount and beat are both 1 so the discount amount will equal the quantity that is discounted (specified in the base expression).
  - **Base Expression = Bal(1000036)**

    The base expression represents the amount to discount. The discount expression Bal(1000036) references the event owner's (the discount sharing group member's) current balance of the quota resource that was granted by the member's plan.

    Because the amount and beat are equal, for every available second in the member's quota balance, a second is added to the balance impact. In this way, you copy the available balance into the event balance.
  - **Consume: Available resource**

    The balance impact is stored in an event balance and is not applied to the account balance so it doesn't matter whether you specify **Consume** or **Impact**. **Consume** is the default.

  - **Event Balance ID = 110**

    This balance impact is stored in the temporary event balance with the ID 110. This event balance is referenced by using the expression EBal(110).

Discount Model Configuration 2: Member's Quota Exceeds Owner's Balance of Free Seconds

In this configuration, if the member's quota balance is greater than the owner's balance of free seconds, the owner's balance of free seconds is stored in a temporary event balance. This balance represents the maximum amount of free seconds that can be consumed. The event balance is used in discount model configuration 3, which updates the member's balances based on the number of seconds used and the maximum free seconds allowed.

Discount Trigger 2 = DTUseFS (discount trigger to store owner's free seconds balance):

- Discount/Chargeshare Condition:
  - **Condition Expression = EBal(109) - Bal(1000036)**
  - **Condition Operator = Greater Than or Equal To**
  - **Condition Value = 0**

  This condition specifies that the member's quota (Bal(1000036)) exceeds the owner's balance of free seconds (EBal(109)). For example, if the owner's balance is -100 and the member's balance is - 150, the condition is **(-100) - (-150) is greater than 0**, or **50 is greater than 0**.

Discount Rule 2 = DRUseFS (Discount Rule to store owner's free seconds balance)

In this rule, you only need to copy the owner's balance of free seconds into an event balance. This means you want the discount applied regardless of the amount of usage. Therefore, you need to make sure the DRUM expression always falls within the step's threshold. A sure way to do this is to make the threshold unlimited and the DRUM a finite amount.

- DRUM:
  - **DRUM Expression = 1.0**
  - **DRUM Type = Quantity**

    Again, the minimum quantity of usage to consider for discounting (the DRUM) is one second (1.0).
  - **Rule Type = Tiered**

    A tiered rule type means that the balance impact is applied when any part of one unit (the DRUM) falls within the discount step's threshold.
- Discount/Chargeshare Step
  - **Threshold From = 0**
  - **Threshold To = Infinity**
- Discount/ChargeShare Balance Impact:

  In this balance impact, for every free second in the owner's current balance of free seconds, a free second is applied to an event balance. This balance impact is used to tell the next balance impact how many free seconds are available for consumption.

- **Impact/Consume = 1000095, Free Seconds**

  This balance impact is applied to the free seconds resource. The balance impact will be stored in an event balance, where this resource ID will be recorded.

- **Applied To = Event Owner**

  This specifies that the balance impacted belongs to the account that generated the usage. However, because the balance impact will be stored in an event balance and not applied to the account balance, the value in Applied To is not considered.

- **Amount = 1; Beat = 1; no proration**

  The amount and beat are both 1 so the discount amount will equal the quantity that is discounted (specified in the base expression).

- **Base Expression = EBal(109)**

  The amount to discount is the discount sharing group owner's current balance of free minutes that was granted by the owner's plan. Recall that the owner's balance of free minutes was stored in the event balance with ID 109 by the previous Owner Quota discount.

  Because the amount and beat are equal, for every second in EBal(109), a second is added to the balance impact. In this way, you copy this available balance into the new event balance.

- **Consume: Available resource**

  The balance impact is stored in an event balance and is not applied to the account balance so it doesn't matter whether you specify **Consume** or **Impact**. **Consume** is the default.

- **Event Balance ID = 110**

  This balance impact is stored in the temporary event balance with the ID 110. This event balance is referenced by using the expression EBal(110).

Discount Model Configuration 3: Consume Free Seconds and Store Quantity Consumed

Thus far, all balance impacts have been stored in event balances and no account balances have yet been modified. This configuration impacts the member account's quota balance and adjusts the currency balance for the free minute charges. The quantity of free seconds consumed by the member is stored in another event balance (EBal(111)). This event balance is used by the next discount to update the free seconds balance in the discount sharing group owner account.

In this configuration, the balance impact of the previous configuration (stored in EBal(110)) determines the maximum free seconds that can be consumed by the discount sharing group member account.

Discount Trigger 3 = DTQuota - Discount trigger for quota usage

- Discount/Chargeshare Condition:
  - **Condition Expression = Bal(1000036)**
  - **Condition Operator = Less Than**
  - **Condition Value = 0**

This condition specifies that the member has an available quota balance. If the resource in the quota balance has been used up, this discount is not applied.

Discount Rule 3 = DRQuota - Discount Rule for Quota usage

In this rule, the balance impacts are based on the actual amount of usage. Therefore, the quantity to consider for discounting (the DRUM) is the total usage in the charge packet. If the total usage exceeds the available free seconds, only the usage up to the available amount should be discounted. Therefore, you use the maximum free seconds that can be consumed (stored by the previous configuration in EBal(110)) as the upper limit of the threshold.

- DRUM:
  - **DRUM Expression = TotalQ**

    The minimum quantity to consider for discounting specifies the discount expression **TotalQ**. This expression evaluates to the total quantity (the total length of the member's call) in the EDR's charge packet.
  - **DRUM Type = Quantity**
  - **Rule Type = Tiered**

    If any part of the total quantity in the charge packet falls within the step's threshold, the discount is applied.
- Discount/Chargeshare Step:
  - **Threshold From = 0**
  - **Threshold To = EBal(110)**

    Threshold To specifies the event balance, EBal(110), which was stored by the previous configuration. This event balance specifies the maximum amount of free minutes that can be consumed. The quantity in this balance is either the owner's balance of free seconds or the member's quota balance, whichever is less.

    When the balance impacts are calculated, the quantity of usage (the DRUM) that falls within 0 and the maximum allowed (the step) is the amount that is discounted.

There are three discount balance impacts for this discount rule:

- Discount/Chargeshare Balance Impact 1:

  This balance impact credits the member account's currency balance for charges applied to usage that qualify for free seconds. This is necessary because charges are applied during rating, before events are discounted.
  - **Impact/Consume = 978, Euro**

    This balance impact is applied to the discount owner's euro resource.
  - **Applied To = Event Owner**

    The euro balance that is impacted belongs to the account that generated the usage (the discount sharing group member account).
  - **Percentage = -100%; no proration**

    A -100% discount credits the account for the entire charge specified in the following base expression.
  - **Base Expression = StepC**

The amount to discount is the discount expression **StepC**. This expression evaluates to the *charge* for the amount of resource (the member's euro balance) that falls within the step's threshold (0 to the available balance of free seconds or quota). The amount to discount, then, is the charge for seconds used, up to the available balance.

For example, if the DRUM is 180 seconds, and the maximum free seconds allowed in the step is 6000 seconds, the entire 180 seconds qualifies as free seconds. Therefore, StepC is the euro (the resource) amount charged for 180 seconds.

– **Consume: Available resource**

This is the default value for currency resources. It specifies that the account's resource can be discounted up to the available amount in the balance.

– **Event Balance ID = 0**

No event balance is specified (ID = 0) so this balance impact is applied to the account balance.

- Discount/Chargeshare Balance Impact 2:

This balance impact reduces the member's quota balance by the number of free seconds used.

– **Impact/Consume = 1000036, Quota**

This balance impact is applied to the quota resource.

– **Applied To = Event Owner**

The quota balance that is impacted belongs to the account that generated the usage.

– **Amount = 1; Beat = 1; no proration**

The amount and beat are both 1 so the discount amount will equal the quantity that is discounted (specified in the base expression).

– **Base Expression = StepQ**

The amount to discount is the expression **StepQ**. This expression evaluates to the *quantity* of the resource (the member's quota balance) that falls within the step's threshold (between 0 and the balance of free seconds or quota). The amount to discount, then, is the number of seconds used, up to the available balance.

For example, if the DRUM is 180 seconds, and the maximum free seconds allowed in the step is 6000 seconds, the entire 180 seconds qualifies as free seconds. Therefore, StepQ is a count (the quota resource) of 180 seconds.

– **Consume: Available resource**

This discount can consume the member's quota resource up to the available amount in the balance.

In this rule, the available balance was used as the upper limit of the threshold. The amount that falls within this threshold is the amount that is discounted (specified by the base expression). Therefore, you know the quota balance contains at least the amount that was consumed.

– **Event Balance ID = 0**

- Discount/Chargeshare Balance Impact 3:

This balance impact stores the number of free seconds consumed in another temporary event balance. This event balance will be used by the next discount to reduce the discount sharing group owner's balance of free seconds. Recall that the owner's balance cannot be directly updated by this discount because this discount is not shared.

- **Impact/Consume = 1000095, Free Seconds**

  This balance impact is applied to the free seconds resource. Recall that this balance impact will be stored in an event balance, where this resource ID will be recorded.

- **Applied To = Discount/Chargeshare Owner**

  This specifies that the free-second balance that is impacted belongs to the discount owner account. However, because the balance impact will be stored in an event balance and not applied to the account balance, the value in Applied To is not considered.

- **Amount = 1; Beat = 1; no proration**

  The amount and beat are both 1 so the discount amount will equal the quantity that is discounted (specified in the base expression).

- **Base Expression = StepQ**

  The amount to discount is the expression **StepQ**. This expression evaluates to the *quantity* of the resource (free second) that falls within the step's threshold (between 0 and the balance of free seconds or quota). The amount to discount, then, is the number of seconds used, up to the available balance.

- **Consume: Available resource**

  The balance impact is stored in an event balance and is not applied to the account balance so it doesn't matter whether you specify **Consume** or **Impact**. **Consume** is the default.

- **Event Balance ID = 111**

  This balance impact is stored in the temporary event balance with the ID 111. This event balance is referenced by using the expression EBal(111).

Discount Model = DMQuota (Discount Model for Quota)

Create a discount model configuration to associate each trigger with a rule.

- Discount Model Version = 1
  - Discount Model Configuration 1: Owner's Balance of Free Seconds Exceeds Member's Quota:

    Trigger = DTUseUQ - Discount trigger to store member's quota balance

    Rule = DRUseQ - Discount rule to store member's quota balance

    Cascading = No

  - Discount Model Configuration 2: Member's Quota Exceeds Owner's Balance of Free Seconds:

    Trigger = DTUseFS - Discount trigger to store owner's free seconds balance

    Rule = DRUseFS - Discount rule to store owner's free seconds balance

    Cascading = No

– "Discount Model Configuration 3: Consume Free Seconds and Store Quantity Consumed":

Trigger = DTQuota - Discount trigger for quota usage

Rule = DRQuota - Discount rule for quota usage

Cascading = No

Owner Quota Free Seconds Discount

The purpose of this discount is to reduce the discount sharing group owner's balance of free seconds by the number of free seconds the member account consumed. This discount takes the amount of free seconds used that was stored in the event balance (EBal(111) by the previous discount, and applies it to the owner's balance of free seconds.

Discount Model Configuration: Update Owner's Free Seconds Balance

Discount Trigger = DTFS - Discount Trigger for Free Seconds discount:

- Discount/Chargeshare Condition:
  - **Condition Expression = EBal(111)**

    EBal(111) contains the number of free seconds consumed by the member account, which was stored by the Member Quota discount in "Discount Model Configuration 3: Consume Free Seconds and Store Quantity Consumed".
  - **Condition Operator = Greater Than**
  - **Condition Value = 0**

  This condition specifies that there were available free seconds that were used.

Discount Rule = **DRFS** - Discount Rule for Free Seconds:

- DRUM:
  - **DRUM Expression = 1.0**
  - **DRUM Type = Quantity**

    The minimum amount of usage to consider for discounting (the DRUM) is one second (1.0).
  - **Rule Type = Tiered**

    A tiered rule type means that the balance impact is applied when any part of one unit (the DRUM) falls within the discount step's threshold.
- Discount/Chargeshare Step:
  - **Threshold From = 0**
  - **Threshold To = EBal(111)**

    Threshold To specifies the event balance (EBal(111)) from the previous discount configuration. The quantity in this balance is the number of free seconds used by the discount sharing group member account.
- Discount/Chargeshare Balance Impact:
  - **Impact/Consume = 1000095, Free Seconds**

    This balance impact is applied to the free seconds resource.
  - **Applied To = Discount/Chargeshare Owner**

The free-second balance that is impacted belongs to the discount sharing group owner account.

- **Amount = 1; Beat = 1; no proration**

  The amount and beat are both 1 so the discount amount will equal the quantity that is discounted (specified in the base expression).

- **Base Expression = StepQ**

  The amount to discount is the discount expression **StepQ**. This expression evaluates to the *quantity* of the resource (the owner's free seconds) that falls within the step's threshold (0 and the number of free seconds used (EBal(111)). The amount to discount, then, is the number of seconds used, up to the available balance.

- **Consume: Available resource**

  This discount can consume the discount sharing group owner's balance of free seconds up to the available amount in the balance.

- **Event Balance ID = 0**

Discount Model = DMPQFS (Discount Model for Free Seconds)

- Discount Model Configuration:
  - **Trigger = DTFS** - Discount Trigger for Free Seconds discount
  - **Rule = DRFS** - Discount Rule for Free Seconds
  - **Cascading = No**

  Because this model contains only one discount configuration, cascading is not relevant.

Create Discount Sharing Groups

You set up a discount sharing group by creating a discount sharing group object (**/group/sharing/discounts**) for the owner account that shares its discount, and ordered balance group object (**/ordered_balgrp**) for each account that consumes the owner's free minutes.

For information about discount sharing groups, see "About Discount Sharing Groups" in *BRM Managing Accounts Receivable*.

To set up the discount sharing groups:

1. Create a discount sharing group by using the PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE opcode. This opcode links the accounts in the group and specifies the discounts that are shared. See "Creating Resource Sharing Groups" in *BRM Managing Accounts Receivable*.
2. Set up an ordered balance group for each member account by using the PCM_OP_SUBSCRIPTION_ORDERED_BALGRP opcode. (The discount sharing group owner doesn't need an ordered balance group). This opcode specifies the order in which the discounts are applied to the member accounts. See "Managing Ordered Balance Groups" in *BRM Managing Accounts Receivable*.

   In the opcode input flist, for this example, you specify the following values:

   - In the PIN_FLD_ACCOUNT_OBJ field, specify the POID of the member account for which this object is created.

– In the PIN_FLD_ORDERED_BALGRP array, specify the POID of the discount objects that are shared in the following order:

  a. The Owner Quota discount

  b. The discount owned by the member.

  c. Owner Quota Free Seconds discount

For more information, see "About Ordered Balance Groups" in *BRM Managing Accounts Receivable*.

14

Global Charge Sharing Configuration Example

This chapter provides an example of how to configure Oracle Communications Billing and Revenue Management (BRM) charge sharing for toll free phone numbers.

Before reading this document, you should be familiar with the following concepts:

- Charge sharing. See "About Charge Sharing Groups" in *BRM Managing Accounts Receivable*.
- Zone models. See "About Zone Models" in *BRM Setting Up Pricing and Rating*.

About Charging Calls Made to a Toll Free Number to a Special Account

The example in this chapter shows how you use BRM charge sharing to split charges between a company and anyone calling its toll free number.

Toll Free Number Scenario

ABC Tones & Co. is a third-party content provider that sells custom ring tones to wireless phone users. The company provides a toll free number (1-800-555-1234) that people can call to hear, purchase, and download ring tones that feature popular rock tunes. ABC Tones & Co. wants anyone to be able to call its 1-800 phone number for free.

Before you set up BRM charge sharing for this example, you need to define the following:

- The criteria for an event to qualify for charge sharing. In this example, only calls to 1-800-555-1234 qualify.
- How to split the charges between the company and the caller. In this example, 100% of the charges are applied to ABC Tones & Co. and 0% of the charges are applied to the caller.

About Setting Up BRM Charge Sharing

To set up BRM charge sharing for this example, you create the following:

- A zone model that finds and flags all calls to 1-800-555-1234.
- A chargeshare that specifies to charge 100% of all calls that meet the criteria to the global charge sharing group owner and 0% to the event owner.
- A global charge sharing group that has ABC Tones & Co. as the owner and all GSM telephony services in your system as members.

Note: You use BRM Content Manager to handle any ring tone purchases and downloads. See "Understanding Content Manager" in *BRM Content Manager*.

Setting Up BRM to Process Calls Made to the Toll Free Number

To set up this charge sharing scenario, perform these tasks:

1. Configuring BRM to Detect and Flag Calls to the Toll Free Number
2. Defining when and how to Split Charges between the Owner and Members
3. Specifying the Eligible Accounts and Services

Configuring BRM to Detect and Flag Calls to the Toll Free Number

You configure BRM to detect and flag calls made to 1-800-555-1234 by creating a zone model. Zone models map event attributes to an impact category. In this example, you create a zone model that maps events with a B number of 1-800-555-1234 to a custom impact category named TOLL_FREE_IMPACT.

To detect and flag calls to the toll free number, perform these tasks in Pricing Center:

1. Create an impact category named TOLL_FREE_IMPACT.
2. Create a zone model named ABC_Zone_Model, and choose Standard as the zone model type.
3. Define the standard zone with the following information:
   - Destination area code: 01-1-800-555-1234
   - Wholesale impact category: TOLL_FREE_IMPACT
   - Retail impact category: TOLL_FREE_IMPACT

   Note: An area code must include the international code and can include the country code, region code, city code, phone number prefix, or entire phone number.

For more information about creating impact categories and zone models, see "Setting up Zones for Batch Pipeline Rating" in *BRM Setting Up Pricing and Rating*.

Defining when and how to Split Charges between the Owner and Members

You define the conditions for an event to qualify for charge sharing and how to split charges between ABC Tones & Co. and any caller by creating a chargeshare.

For the toll free example in this document, you create the following:

- A chargeshare model named ABC_Model that applies 100% of all charges to the global charge sharing group owner and 0% of the charges to the event owner.
- A chargeshare that maps ABC_Model to GSM telephony events.

To create a chargeshare, perform these tasks in Pricing Center:

1. Start a chargeshare model named ABC_Model.
2. Create a chargeshare master named ABC_Master with the following attributes:

- Impact category: TOLL_FREE_IMPACT
- RUM: Duration

3. Create a chargeshare rule named ABC_Rule with the following attributes:
   - Chargeshare master: ABC_Master
   - DRUM expression: TotalC
   - DRUM type: Charge
4. Create a chargeshare trigger named ABC_Trigger with the following attributes:
   - Condition expression: TotalC
   - Condition operator: Greater than
   - Condition value: 0
5. Finish the ABC_Model chargeshare model by specifying the rule and trigger to use:
   - Trigger: ABC_Trigger
   - Rule: ABC_Rule
6. Create a chargeshare named ABC_Chargeshare that maps **/event/session/telco/gsm** events to ABC_Model.

Specifying the Eligible Accounts and Services

You specify the accounts or services that are eligible for charge sharing by creating a global charge sharing group. For the toll free example in this document, you create a global charge sharing group that includes all GSM telephony services in your system as members.

You use Customer Center or a third-party client application to create a global charge sharing group with the following attributes:

- Owner: ABC Tones & Co. (**0.0.0.1 /account 123456 10**)
- Members: A type-only POID for GSM telephony (**0.0.0.1 /service/telco/gsm/telephony -1 0**)
- Chargeshare: ABC_Chargeshare (**0.0.0.1 /sponsorship 123456 10**)

Important: Make sure the global charge sharing search is enabled before you create any global charge sharing groups. See "Enabling Global Charge Sharing Searches During Discounting" in *BRM Managing Accounts Receivable*.

Using a Third-Party Client Application to Specify the Eligible Accounts

To create the global charge sharing group by using a third-party client application, customize your application to send an flist to the PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE opcode, similar to the one shown below:

```
0 PIN_FLD_POID               POID [0] 0.0.0.1 /account 123456 10
0 PIN_FLD_GROUP_OBJ          POID [0] 0.0.0.1 /group/sharing/charges -1 0
0 PIN_FLD_BAL_GRP_OBJ        POID [0] 0.0.0.1 /balance_group 123456 10
0 PIN_FLD_NAME                STR [0] "ABC_Tones"
0 PIN_FLD_PARENT             POID [0] 0.0.0.1 /account 123456 10
0 PIN_FLD_MEMBERS           ARRAY [0] allocated 2, used 2
```

```
1   PIN_FLD_ACCOUNT_OBJ       POID [0] 0.0.0.1 /account -1 0
1   PIN_FLD_SERVICE_OBJ       POID [0] 0.0.0.1 /service/telco/gsm/telephony -1 0
0 PIN_FLD_SPONSORS           ARRAY [0] allocated 1, used 1
1   PIN_FLD_SPONSOR_OBJ       POID [0] 0.0.0.1 /sponsorship 123456 10
```

For more information, see "Using Third-Party Client Applications to Create, Modify, and Delete Global Charge Sharing Groups" in *BRM Managing Accounts Receivable*.

Using Customer Center to Specify the Eligible Accounts

To create a global charge sharing group for the example in this document, perform these tasks in Customer Center:

1. Create a charge sharing group by following the instructions in "Creating a Charge Sharing Group" in *BRM Managing Accounts Receivable*. Make sure you specify ABC_Chargeshare as the chargeshare to use.
2. Select the global charge sharing group owner's account.
3. Click the **Sharing** tab.
4. Select **Charge Sharing** from the **View** box.
5. In the **Members in this Group** column (**Charge Sharing Ownership** table), click the **Add Members** link.

   The **Sharing** tab displays the **Members** panel.
6. Click **Add** on the **Members** panel.

   The Add Members dialog box opens.
7. Select the **All accounts are members** check box.

   Customer Center disables the search criteria fields and clears any search results.
8. Click **Next**.
9. Select **Selected service types**.

   The **Service Types** panel is displayed and lists all service types currently supported in your system.
10. Select the **/service/telco/gsm/telephony** service.
11. Click **Finish**.

    The Add Members Confirmation dialog box opens.
12. Click **Yes** to confirm the GSM telephony service as a member.

For more information, see "Adding Members to a Discount or Charge Sharing Group" in *BRM Managing Accounts Receivable*.

15

Discounting Utilities

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager discounting utilities.

pin_discount_cleanup

Use this utility to change the status of expired discounts from **active** to **canceled** and to delete canceled discounts.

You use this utility to close or delete the discounts that are canceled in the middle of a cycle and the discount's validity rule is set to Full discount. For more information, see "Changing the Status of Discounts Canceled in Mid-Cycle".

For information about discount validity rules, see "About Applying Discounts Activated or Canceled in Mid-Cycle".

You can run this utility daily or add it to the **pin_bill_day** script to be run automatically. See "Running Billing Utilities" in *BRM Configuring and Running Billing*.

Location

BRM_home/**bin**

Syntax

```
pin_discount_cleanup -m [close|delete] [-n days] [-d date] [-v] [-t] [-help]
```

Parameters

-m close|delete
Specifies whether to delete discounts when they are canceled:

- **close**

  Changes the status of all active, expired discounts to **canceled** without deleting the discounts.

- **delete**

  Deletes all expired discounts.

-n days
The number of days prior to **-d** *date* for which the expired discounts are retained. The utility changes the status to **canceled** for the discounts that expired more than **-n** *days* prior to **-d** *date*.

- d date
The end date (in the format *MM/DD/YYYY*) of the period in which discounts that expired are retained.

For example, if **-n** is **5** and **-d** is **07/15/2015**, the status of the discounts that expired before 7/10/2015 are changed to canceled.

Note:

The expiry date cannot be greater than the current date. For instance, in the example above, if 7/10/2015 is greater than the current date, **pin_discount_cleanup** returns an error. Similarly, if only **-d** is specified, and is greater than the current date, **pin_discount_cleanup** returns an error.

If neither -**n** nor **-d** parameter is specified, the current date is used.

-v
Displays information about successful or failed processing as the utility runs.

> **Note:** This parameter is always used in conjunction with other parameters and commands. It is not position dependent. For example, you can enter -v at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace filename.log with the name of the log file:
>
> **pin_discount_cleanup** *other_parameters* **–v** > *filename*.**log**

-t
Displays the number of records processed (the number of discounts that were canceled.

-help
Displays the syntax and parameters for this utility.

Results

To check results of running this utility, look in the log file (normally **default.pinlog**) for error messages. The log file is located in the directory from which the utility was started or in a directory specified in the utility's configuration file (**pin.conf**).

load_pin_snowball_distribution

Use this utility to load snowball discount distribution rules into the **/config/snowball_distribution** object in the BRM database. You define how snowball discounts are distributed in the **pin_snowball_distribution** file in *BRM_home*/**sys/data/pricing/example**.

For more information, see "About Snowball Discounts".

Caution: This utility overwrites existing distribution rules. If you are updating distribution rules, you cannot load new distribution rules only. You must load a complete set of distribution rules each time you run this utility.

Important:

- To connect to the BRM database, this utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.
- Before you load snowball distribution rules, you must first load the price lists in the **pin_beid** file.

Location

BRM_home/**bin**

Syntax

```
load_pin_snowball_distribution pin_snowball_distribution_file [-d] [-v]
```

Parameters

pin_snowball_distribution_file
The name and location of the file that defines the snowball distribution rules. The default **pin_snowball_distribution** file is in *BRM_home*/**sys/data/pricing/example**.

If you do not run the utility from the directory in which the file is located, you must include the complete path to the file. For example:

```
load_pin_snowball_distribution BRM_home/sys/data/pricing/example
```

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors, but the data has not been loaded into the database.

-v
Displays information about successful or failed processing as the utility runs.

Note: This parameter is always used in conjunction with other parameters. To redirect the output to a log file, use the following syntax:

load_pin_snowball_distribution *other_parameter* **–v >** *filename*.**log**

Part III

Suspending and Recycling EDRs

Part III describes how to suspend and recycle EDRs in an Oracle Communications Billing and Revenue Management (BRM) system. It contains the following chapters:

16

About the EDR Recycling Features

This chapter provides an overview of the Oracle Communications Billing and Revenue Management (BRM) features used to manage suspended (failed) call records (EDRs).

About the EDR Recycling Features

BRM offers these tools for managing EDRs that are not successfully rated by Pipeline Manager:

- **Standard recycling**. BRM provides the standard recycling tools as the default EDR recycling mechanism. Using standard recycling, you use the **pin_recycle** utility to test recycle, recycle, or delete EDRs that failed processing the first time through the pipeline.

  The standard recycling tools include:

  - FCT_Reject
  - FCT_PreSuspense
  - FCT_Suspense
  - **pin_recycle** utility

  For details on standard recycling and configuring the standard recycling tools, see "About Standard Recycling".

- **Suspense Manager**. Suspense Manager is a service integration component that you purchase separately. It offers the most comprehensive and flexible set of tools for managing:

  - Individual failed CDRs
  - Large numbers of individual failed CDRs at once (bulk processing)
  - CDR files containing multiple CDRs (batch processing)

  Suspense Manager includes the Suspense Management Center GUI application to:

  - Analyze, edit, recycle, test recycle, write off, archive, and delete CDRs, either individually or in bulk.
  - Analyze, resubmit, write off, and delete batch files of CDRs.

  Suspense Manager also includes a set of BRM reports for analyzing suspended call records. For details, see "About Suspense Manager".

- **Recycling EDRs for pipeline-only systems**. This feature is used by customers that use Pipeline Manager, but do not store suspended EDRs in the BRM database. This feature includes the FCT_Recycle and FCT_PreRecycle pipeline modules that you

use to recycle suspended EDRs. For details see "Recycling EDRs in Pipeline-Only Systems".

17

About Standard Recycling

This chapter provides an overview of the Oracle Communications Billing and Revenue Management (BRM) standard recycling features.

For information on the other BRM recycling features, see "About the EDR Recycling Features".

Before using standard recycling, you should be familiar with Pipeline Manager. For details, see "About Pipeline Rating".

About Standard Recycling

You use standard recycling to recycle, test recycle, or delete failed event data records (EDRs).

Standard recycling mainly relies on these BRM tools to suspend and recycle EDRs:

- The "FCT_Reject" pipeline module
- The "FCT_PreSuspense" pipeline module
- The "FCT_Suspense" pipeline module
- The Suspended Event (SE) Loader application
- The "pin_recycle" utility

You use "pin_recycle" to recycle, test recycle, or delete suspended call records. EDRs are often suspended because of a pipeline configuration problem. You then fix the problem, and test recycle a CDR file of suspended call records. If they pass the recycle test, then you recycle all of the CDR files of suspended calls. **pin_recycle** also has a delete option to remove call records that have been successfully processed, or call records that cannot be rated.

Standard Recycling Workflow

Overview of the standard recycling process:

1. You start the pipeline with the **FCT_PreSuspense**, **FCT_Suspense**, and **FCT_Reject** modules active.
2. **FCT_PreSuspense** appends suspense-related information to all EDRs that come through the pipeline.
3. As an EDR is processed, a module finds an error in the EDR. The error is appended to the EDR, and a flag is set to indicate that the EDR has an error.
4. The EDR is sent to the next module. Each module adds errors, if any more are found.

5. The **FCT_Reject** module analyzes an EDR's errors to determine whether it has failed. **FCT_Reject** also routes EDRs to the appropriate output stream to be stored in the database by Suspended Event (SE) Loader. SE Loader stores suspended EDRs in **/suspended_usage** objects

   By default, FCT_Reject fails call records with an error level of **Warning** or **Error**. However, you configure the error level or other conditions that causes EDRs to fail. Call records also "fail" if they cannot otherwise be processed by the pipeline. These failures can be intentional or inadvertent. For example:

   - A call record may arrive with invalid data and fail a Pipeline Manager validity rule.
   - The call record may fail custom validity checking set up in a custom iScript.
   - The Pipeline Manager database tables may be set up incorrectly.

6. During recycling operations, **FCT_Suspense** routes EDRs from **SuspenseCreateOutput** to **SuspenseUpdateOutput**.
7. You examine the errors and determine how to reconfigure Pipeline Manager to prevent the errors.
8. Run the **pin_recycle** utility with the **-f** *filename* option to start the recycling process. This sends the rejected EDRs through the pipeline again for another attempt to rate them.

   pin_recycle can recycle EDRs in test mode or real mode. Typically, you run the recycling processes in test mode first, to see if the problems causing the EDR errors have been fixed. When there are no longer any errors, you recycle in real mode.

   You usually run **pin_recycle** (as part of a **cron** job) periodically.

   - In test mode, this utility creates a report about the processing, but does not make any state changes.
   - In recycle mode, this utility sends the results to an output file, and attaches a sequence number to the output file.

   Note: This utility sends an entire CDR file to the error directory. You can configure the threshold for the number of errors allowed per file. See "Specifying the Maximum Errors Allowed in an Input File".

9. Run **pin_recycle** again with the delete option to remove any remaining unratable EDRs.

For details on the **pin_recycle** utility, see "pin_recycle".

For details on configuring Pipeline Manager to use standard recycling, see "Configuring Standard Recycling".

For details on using standard recycling to recycle, and delete EDRs, see "Using Standard Recycling to Recycle Suspended EDRs".

Suspended EDR States

As suspended EDRs are processed by standard recycling, they are assigned one of the following states:

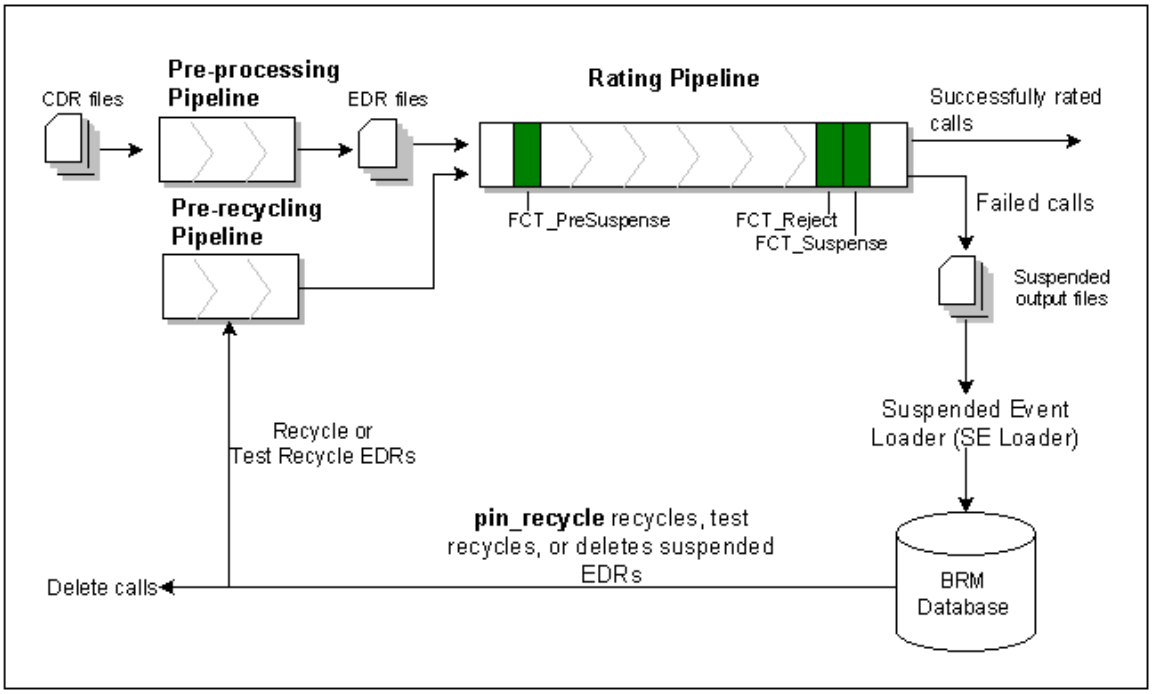
- **Suspended**. The call record could not be processed by the pipeline and has been stored in the BRM database as a suspended call record.

- **Recycling**. The call record is being sent through the rating pipeline again to be rated
- **Succeeded**. The call record has been successfully recycled and rated.
- **Written off**. The EDR is set to this state automatically just before being deleted to generate revenue assurance data.

About the Standard Recycling Pipelines

Figure 17–1 shows how standard recycling fits into your BRM system.

Figure 17–1 Standard Recycling in BRM



Call records first enter standard recycling through the *preprocessing pipeline*. The preprocessing pipeline converts call records (CDRs) to EDRs used by BRM. Calls only go through this pipeline once, so only a few modules are appropriate for it. **FCT_DuplicateCheck** and **FCT_CallAssembling** are candidates.

The *rating pipeline* is a normal rating pipeline. Most of your pipeline function modules are included in this pipeline. It is in this pipeline where you configure call "success" and "failure" policies. If calls "fail" in this pipeline, they are sent to a Suspended Event Loader (SE Loader).

SE Loader converts the failed calls to **/suspended_usage** objects in the BRM database.

After these objects are stored in the BRM database, you check your Pipeline Manager log files to see what caused the calls to fail. The problem can frequently be fixed by reconfiguring the pipeline.

Suspended calls that you recycle or test recycle are processed by the *pre-recycle pipeline* before they go through the rating pipeline again. The pre-recycle pipeline converts the suspended call objects back into files that the pipeline can process, and routes the suspended call records back through their original pipeline for recycling.

If you are test recycling calls, the pipeline tries to rate the calls, but does not make any changes to the database.

What's Next

The next step is to configure standard recycling. For details, see "Configuring Standard Recycling".

18

Configuring Standard Recycling

This chapter explains how to set up the Oracle Communications Billing and Revenue Management (BRM) standard recycling feature.

Before you read this document, you should be familiar with:

- Pipeline Manager and how to set up pipeline modules. See these documents:
  - About Pipeline Rating
  - Configuring Pipeline Rating
  - Configuring EDR Input Processing
  - Configuring EDR Output Processing
  - Configuring EDR Preprocessing
- Editing **pin.conf** configuration files and using file loading utilities. For details, see "Using Configuration Files to Connect and Configure Components" in *BRM System Administrator's Guide*.

> **Important:** Suspense Manager customers must complete the configuration instructions in this chapter first, and then follow the instructions in "Configuring Suspense Manager".

About Configuring Standard Recycling

Table 18–1 lists the tasks required for configuring standard recycling:

Table 18–1 Standard Recycling Configuration Tasks

Task	Description
Configure the pre-processing pipeline See "Configuring a Preprocessing Pipeline".	▪ Configure your input module. ▪ Configure the OUT_GenericStream pipeline module. ▪ Configure MultiDB routing logic (optional).

Table 18–1 (Cont.) Standard Recycling Configuration Tasks

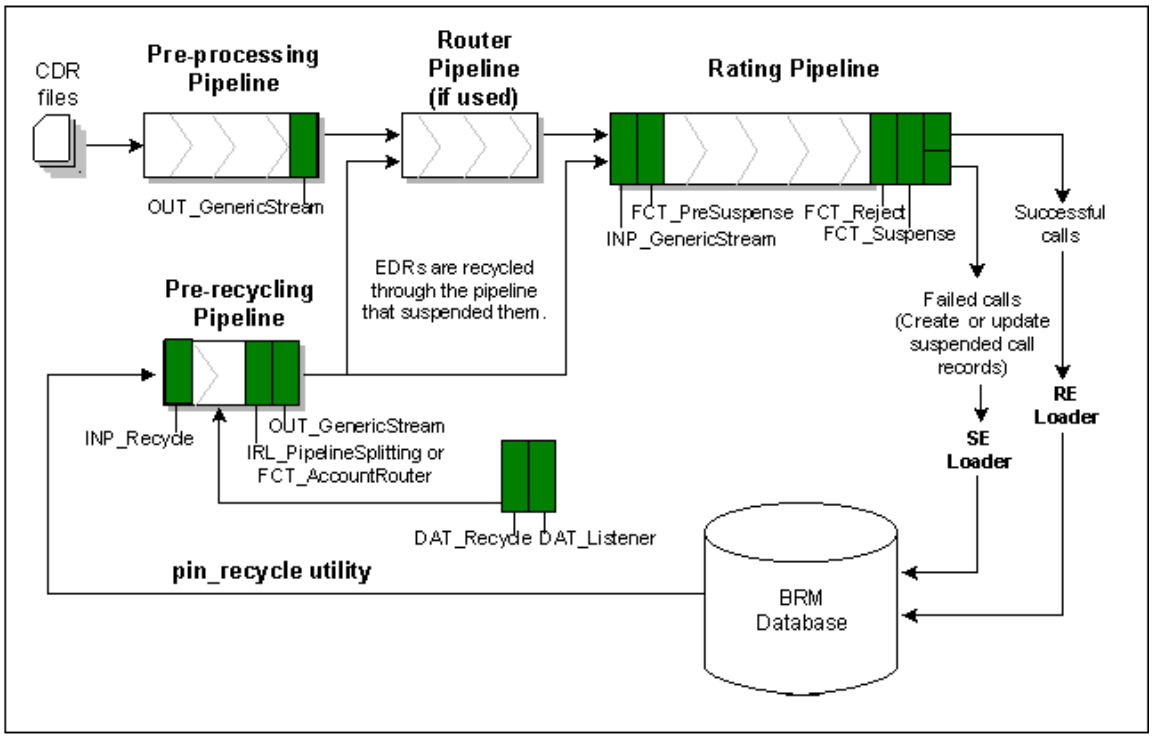
Task	Description
Configure the rating pipeline See "Configuring Standard Recycling in a Rating Pipeline".	■ Configure the INP_GenericStream pipeline module. ■ Configure the FCT_PreSuspense pipeline module. ■ Configure the FCT_Reject pipeline module. ■ Set RejectStream to SuspenseCreateOutput. ■ Configure the FCT_Suspense or pipeline module. ■ Configure the SuspenseCreateOutput registry entry. ■ Configure the SuspenseUpdateOutput registry entry.
Configure the pre-recycling pipeline See "Configuring a Pre-Recycling Pipeline".	■ Configure the INP_Recycle pipeline module. ■ Edit the IRL_PipelineSplitting.data pipeline module. ■ Configure the OUT_GenericStream pipeline module.
Configure recycle request handling See "Configuring Recycle Request Handling".	■ Configure the DAT_Listener pipeline module. ■ Configure the DAT_Recycle pipeline module.
Configure a pipeline module to add recycle keys to EDRs (if needed) See "Configuring a Pipeline Module to Add Recycle Keys to EDRs".	■ Used by features that temporarily suspend rating.
Configure **pin_recycle** See "Configuring the pin_recycle Utility".	■ Set up **pin_recycle** to recycle suspended EDRs.
Set up Suspended Event (SE) Loader See "Configuring SE Loader for Standard Recycling".	■ Configure the Batch Controller. ■ Edit the **Infranet.properties** file.
Confirm that **pin_rel** is configured.	■ Ensure that **pin_rel** is configured for standard recycling.
Map EDR field values to brand information (if needed) See "Mapping EDR Fields to Brand Information".	■ Edit the **pin_suspense_edr_fld_map** file. ■ Run the **load_pin_suspense_edr_fld_map** utility.

Configuring Pipeline Modules for Standard Recycling

Standard recycling requires you to configure the rating pipeline to correctly handle suspended call records. For an example of a complete sample pipeline, see the *Pipeline_Home***/conf/wireless.reg**. *Pipeline_Home* is the directory where you installed Pipeline Manager.

Figure 18–1 shows in green the pipeline modules that you need to configure:

Figure 18–1 Pipeline Modules to Configure for Standard Recycling



Configuring a Preprocessing Pipeline

Standard recycling requires a preprocessing pipeline, and this section explains how to set it up.

For a complete example of a preprocessing pipeline, see *Pipeline_Home*/**conf/wireless.reg**.

All call records coming into your system for rating go through the preprocessing pipeline only once. After the preprocessing pipeline, EDRs go to the rating pipeline. Failed calls may be recycled through the rating pipeline multiple times, but they skip the preprocessing pipeline after going through once.

To configure your preprocessing pipeline:

1. Define preprocessing pipelines in the registry. You need a separate preprocessing pipeline for each input format your system uses.
2. Set up the input module of each pipeline to process call records from the external system you are using. You need a different pipeline for each call record format.
3. Configure the **OUT_GenericStream** pipeline module as an output module of the preprocessing pipeline.
   - Add this entry to the DataDescription.StreamFormats section of each preprocessing pipeline:

     ```
     SOL42=./FormatDesc/Solution42/SOL42_V670_REL.dsc
     ```

   - Add this entry to the DataDescription.OutputMappings section of each preprocessing pipeline:

     ```
     SOL42=./FormatDesc/Solution42/SOL42_V670_REL_OutMap.dsc
     ```

- Add this entry to the output module section of each preprocessing pipeline:

```
Grammar=./FormatDesc/Solution42/SOL42_V670_REL_OutGrammar.dsc
```

For complete examples of these registries, see *Pipeline_Home*/**conf/wireless.reg**. For details on this module, see "OUT_GenericStream".

4. (Optional) Multidatabase Manager users typically add a Multidatabase Manager routing pipeline after the preprocessing pipeline. For details on setting up Multidatabase Manager, see "Installing a Multischema System" in *BRM Installation Guide*.

The preprocessing pipeline is now configured. Follow the steps in the next section to configure the rating pipeline.

Configuring Standard Recycling in a Rating Pipeline

All calls go through a rating pipeline at least once, and suspended calls may be recycled through this pipeline multiple times.

For a complete example of a rating pipeline, see *Pipeline_Home*/**conf/wireless.reg**.

Note: You must use the input description file specified below. Customized description files are not supported.

To configure your rating pipeline:

1. Configure the **INP_GenericStream** pipeline module as the input module.
   - Add this entry to the DataDescription.StreamFormats section of each rating pipeline:

```
SOL42_INPUT=./FormatDesc/Solution42/SOL42_V670_REL_ForInput.dsc
```

   - Add this entry to the DataDescription.InputMappings section of each rating pipeline:

```
SOL42_INPUT=./FormatDesc/Solution42/SOL42_V670_REL_InMap.dsc
```

   - Add this entry to the input module section of each rating pipeline:

```
Grammar=./FormatDesc/Solution42/SOL42_V670_REL_InGrammar.dsc
```

   For examples of these entries, see *Pipeline_Home*/**conf/wireless.reg**. For details on this module, see "INP_GenericStream".
2. Configure **FCT_PreSuspense** as the first function module of the pipeline. For details, see "FCT_PreSuspense".
3. Configure **FCT_Reject** to route suspended calls to the suspense create output module (in Step 5.). For details, see "How the FCT_Reject Module Works" and "FCT_Reject".
4. Set the **RejectStream** entry to **SuspenseCreateOutput** in the rating pipeline:

```
...
ALL_RATE
{
   Active                       = true
```

```
   CountryCode                 = 49
   MobileCountryCode           = 262
   NationalAccessCode          = 0
   InternationalAccessCode     = 00
   InternationalAccessCodeSign = +
   NetworkDestinationCode      = 172
   RejectStream                = SuspenseCreateOutput
...
```

5. Configure **FCT_Suspense** as the last function module of the rating pipeline. You need to configure the registry section of this module. For details, see "FCT_ Suspense".
6. Confirm that your pipeline contains a **MaxErrorRates** output entry. If this entry is missing unexpected log file messages may result. For details on adding this entry, see "Specifying the Maximum Errors Allowed in an Input File".
7. Configure the suspense create output module as one of the output modules for this pipeline.

   The following example works as a suspense create output module. Add the **/suspended_usage** object produced by this pipeline in the **EventType** entry:

```
SuspenseCreateOutput

{
   ModuleName                  = OUT_GenericStream

   EventType = /suspended_usage

   Module
   {
      Grammar                  =
./formatDesc/Formats/SuspenseHandling/SuspendedUsageCreationGrammr.dsc
      DeleteEmptyStream        = True

      OutputStream
      {
      ModuleName              = EXT_OutFileManager
      Module
      {
         OutputPath            = ./data/reject
         OutputPrefix          = suspense_create_
         OutputSuffix          = .out
         TempPrefix            = tmp

         TempDataPath          = ./data/reject
         TempDataPrefix        = susp.create.tmp.
         TempDataSuffix        = .data

         Replace               = True
         AppendSequenceNumber  = False
         }
      }
   }
} # end of SuspenseCreateOutput
```

Important: To ensure output file integrity, specify a unique combination of **OutputPath**, **OutputSuffix**, and **OutputPrefix** values for each output stream defined in the registry.

8. Configure the suspense update output module as one of the output modules for this pipeline.

   This example implements a suspense output module:

```
#----------------------------
SuspenseUpdateOutput
{
   ModuleName                  = OUT_GenericStream

   EventType = /tmp_suspended_usage

   Module
   {
      Grammar                   =
./formatDesc/Formats/SuspenseHandling/SuspendedUsageUpdateGrammar.dsc
      DeleteEmptyStream         = True

      OutputStream
      {
         ModuleName              = EXT_OutFileManager
         Module
         {
         OutputPath            = ./data/reject
         OutputPrefix          = suspense_update_
         OutputSuffix          = .out
         TempPrefix            = tmp

         TempDataPath          = ./data/reject
         TempDataPrefix        = susp.update.tmp.
         TempDataSuffix        = .data

         Replace               = True
         AppendSequenceNumber  = False
         }
      }
   }
} # end of SuspenseUpdateOutput
```

Important: To ensure output file integrity, specify a unique combination of OutputPath, OutputSuffix, and OutputPrefix values for each output stream defined in the registry.

Configuring a Pre-Recycling Pipeline

When suspended call records are recycled, they are first processed by a pre-recycling pipeline and then reprocessed by the original rating pipeline.

The pre-recycling pipeline used the INP_Recycle module. This module is used by standard recycling and Suspense Manager. It reads suspended usage records from the BRM database, restores original EDRs, applies edits to them, and pushes EDRs into the pre-recycling pipeline.

For a complete example of a pre-recycling pipeline, see *Pipeline_Home*/**conf/wireless.reg**.

To configure your pre-recycling pipeline:

1. Configure INP_Recycle as the input module. For details, see "INP_Recycle".
   - In the **EXT_InEasyDB** module, change the **SqlDetail** entry to **StdRecycleDetail.sql**.
2. Add and configure a pipeline module to send call records to the correct stream.
   - **Single database schema systems:** Use the "IRL_PipelineSplitting" module. Add and configure the **IRL_PipelineSplitting** module (an iRules module). Add this module to the pipeline registry iRules (its order in the registry is not important). Edit the **IRL_PipelineSplitting.data** file (in the *Pipeline_Home*/**iScriptLib/iScriptLib_Suspense** directory), adding pipeline name/output stream pairs. For syntax details, see "FCT_IRules".
   - **Multischema systems that require Account Migration Manager:** Use the "FCT_AccountRouter" module. Add and configure this module.

     Important: AMM is not part of base BRM. Contact your BRM account manager for information about using AMM.

3. Configure the **OUT_GenericStream** pipeline module as an output module of the pre-recycling pipeline. Create a different "OUT_GenericStream" module for each rating pipeline used to recycle suspended calls.
   - Add this entry to the DataDescription.StreamFormats section of the pre-recycling pipeline:

     ```
     SOL42=./FormatDesc/Solution42/SOL42_V670_REL.dsc
     ```

   - Add this entry to the DataDescription.OutputMappings section of the pre-recycling pipeline:

     ```
     SOL42=./FormatDesc/Solution42/SOL42_V670_REL_OutMap.dsc
     ```

   - Add this entry to each output module section of the pre-recycling pipeline:

     ```
     Grammar=./FormatDesc/Solution42/SOL42_V670_REL_OutGrammar.dsc
     ```

   For details, see "OUT_GenericStream".

   Your pipelines are now ready to accept call records.

Configuring Recycle Request Handling

To configure recycle request handling, add **DAT_Recycle** to the registry data pool. Here is an example:

```
RecyclingData
{
   ModuleName=DAT_Recycle
   Module
   {
      Listener     =ifw.DataPool.Listener
      LogEvents    =True
      ControlPath  =./database/Oracle/Scripts/Suspense
      ParameterFile =parameter.isc
```

```
    }
}
```

For details, see "DAT_Recycle".

Configuring a Pipeline Module to Add Recycle Keys to EDRs

Programs and features that must temporarily interrupt and then restart rating use **pin_recycle** to recycle all EDRs after the interruption is over. The BRM features that add recycle keys to EDRs all have pipeline modules for doing this. See the feature documentation for details.

Configuring the pin_recycle Utility

To configure **pin_recycle** to recycle EDRs, set up a configuration file for **pin_recycle**. For details, see "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide* and "pin_recycle".

Configuring SE Loader for Standard Recycling

The procedures for installing, configuring, and using SE Loader are identical to those of RE Loader, except for the step listed here. For details see:

- Understanding Rated Event Loader
- Installing Rated Event Loader
- Configuring Rated Event Loader
- Loading Prerated Events
- "pin_rel" in *BRM Setting Up Pricing and Rating*

Standard recycling requires SE Loader configuration. Perform these tasks to set it up:

1. Add a separate instance of SE Loader to each pipeline.
2. Create a new *BRM_Home*/**apps/pin_rel/suspense** directory by copying the contents of *BRM_Home*/**apps/pin_rel/gsm/tel** to *BRM_Home*/**apps/pin_rel/suspense**. *BRM_Home* is the directory where you installed BRM components.
3. Confirm that these files are in the *BRM_Home*/**apps/pin_rel/suspense** directory:
   - **pin.conf**
   - **SampleRelHandler_config.values**
   - **SampleRelHandler.pl**
4. Add these entries to the *BRM_Home*/**apps/pin_rel/suspense/SampleRelHandler_config.values** file:

   ```
   $FILETYPE = "*.out.bc";
   $HANDLER_DIR = "BRM_Home/apps/pin_rel/suspense";#
   ```

5. Edit the *BRM_Home*/**apps/batch_controller/Infranet.properties** file, adding **SUSPENSE** and **RECYCLE_ROLLBACK** entries to **batch.random.events**:

   ```
   batch.random.events     TEL,SMS,FAX,DATA,GPRS,SUSPENSE,RECYCLE_ROLLBACK
   ```

   Add these parameters to the new entries:

   ```
   #for SUSPENSE events:
   SUSPENSE.name                    SUSPENSE Usage
   ```

```
SUSPENSE.handlers                suspHandler
SUSPENSE.file.location           Pipeline_Home/data/reject
SUSPENSE.file.pattern            suspense_ *.out

suspHandler.name                 suspHandler
suspHandler.max.at.highload.time 1
suspHandler.max.at.lowload.time  1
suspHandler.start.string           BRM_Home/apps/pin_rel/suspense
/SampleRelHandler.pl

#For RECYCLE_ROLLBACK events:
RECYCLE_ROLLBACK.name            RECYCLE_ROLLBACK Usage
RECYCLE_ROLLBACK.handlers        recycleRollbackHandler
RECYCLE_ROLLBACK.file.location   Pipeline_Home/data/error
RECYCLE_ROLLBACK.file.pattern    testDB*.err

recycleRollbackHandler.name      recycleRollbackHandler
recycleRollbackHandler.max.at.highload.time  1
recycleRollbackHandler.max.at.lowload.time   1
recycleRollbackHandler.start.string      BRM_Home/apps/pin_
rel/recycle/SampleRelHandler.pl
```

6. Confirm that these *BRM_Home***/apps/pin_rel/Infranet.properties** file entries are set to **false**:

```
infranet.rel.validate_dbnumber = false
infranet.rel.validate_indexes = false
```

> **Note:** The SE Loader architecture makes obsolete the database consistency checks and number validation controlled by these entries.

7. Create a new *BRM_Home***/apps/pin_rel/recycle** directory by copying the contents of *BRM_Home***/apps/pin_rel/gsm/tel** to *BRM_Home***/apps/pin_rel/recycle**.
8. Add these entries to the *BRM_Home***/apps/pin_rel/recycle/SampleRelHandler_config.values file**:

```
$FILETYPE = "*.err.bc";
$HANDLER_DIR = "BRM_Home/apps/pin_rel/recycle";#
```

9. Add this entry to each output stream of the pre-recycling pipeline in your *Pipeline_Home***/conf/wireless.reg** file:

```
EventType = /recycle_suspended_usage
```

Mapping EDR Fields to Brand Information

If you use Brand Manager, you need to follow these instructions to make standard recycling brand-aware.

The purpose of mapping unique brand values to EDR fields is to allow standard recycling to identify the brand for a call record if it does not contain enough information to identify the subscriber. This process involves mapping values unique to a brand (such as a range of phone numbers) to the corresponding EDR fields. You also give these mappings a valid time period.

You load this brand-to-EDR field mapping into **/config/suspense_edr_fld_map** objects in the BRM database by using the **load_pin_suspense_edr_fld_map** utility. For details, see "load_pin_suspense_edr_fld_map".

The **FCT_Suspense** module normally adds the brand POID to suspended call records during pipeline processing. If it is unable to do so, then FCT_Suspense uses the information in **/config/suspense_edr_fld_map** to determine the brand POID.

To load the necessary brand information into standard recycling:

1. Determine the EDR fields to map to each brand.
2. Determine the valid value ranges for each field.
3. Decide how long to make the number-to-brand mapping valid.
4. Look up the POIDs of brands.
5. Edit the **pin_suspense_edr_fld_map** file (in the *BRM_Home***/sys/data/config** directory) to map your brands to their POIDs.

   The syntax for an entry:

   ```
   edr_field Name   field_name
   edr_field_number     valid_from_date    valid_to_date    brand_POID
   ```

   The *valid_from_date* and *valid_to_date* use the YYYYMMDD format.

   Excerpt from a sample file with examples:

   ```
   ============================================================
   #
   # Sample file:
   #============================================================
   #
   edr_field_name  DETAIL.A_NUMBER
   881768*      20031231       20371231     0.0.0.5 /newtel 1 1
   881759*      20031231       20371231     0.0.0.5 /newtel 1 1

   edr_field_name  DETAIL.ASS_GSMW_EXT.PORT_NUMBER
   66325*        20031231      20371231     0.0.0.5 /newtel 1
   66324*        20031231       20371231     0.0.0.5 /newtel 1 1
   ```

   This example shows:

   - That all call records with DETAIL.A_NUMBERs that start with **881768** map to the brand with the POID of **0.0.0.5 /newtel 1 1**.
   - The mapping for this entry is valid from December 31, 2003 through December 31, 2037.
6. (Optional) Edit **pin_suspense_edr_fld_map** to map your EDR fields to brands for a specific period of time.
7. Run the **load_pin_suspense_edr_fld_map** utility (in the *BRM_Home***/bin** directory**)** to load the brand mapping information into the database. For details, see "load_pin_suspense_edr_fld_map".

   Important: The **load_pin_suspense_edr_fld_map** utility requires a configuration file to function correctly. For details, see "Using Configuration Files to Connect and Configure Components" in *BRM System Administrator's Guide*.

   Sample syntax:

```
%load_pin_suspense_edr_fld_map pin_suspense_edr_fld_map
```

This program loads your mapping information into a **/config/suspense_edr_fld_map** object.

What's Next

You have now set up BRM and Pipeline Manager to accept and recycle suspended EDRs. For details on day-to-day tasks necessary to recycle EDRs, see "About Standard Recycling" and "Using Standard Recycling to Recycle Suspended EDRs".

19

Using Standard Recycling to Recycle Suspended EDRs

This chapter explains how to use the Oracle Communications Billing and Revenue Management (BRM) **pin_recycle** utility to recycle suspended EDRs. EDRs are usually recycled for one of two reasons:

- They were suspended by the pipeline because of a problem with the pipeline or the EDR. Once the problem is fixed, you recycle the EDRs by using the BRM standard recycling tools in another attempt to rate them. The standard recycling tools recycle all EDRs from the same CDR file at the same time.
- They were suspended intentionally by a BRM program that required a temporary interruption in rating. These programs mark the EDRs with a recycle key and store them until the interruption is over. All EDRs with the same recycle key are recycled at the same time.

In both cases you use **pin_recycle** to recycle the suspended EDRs back through the pipeline to rate them and capture the revenue they represent. See "pin_recycle".

For information on how to configure Pipeline Manager to suspend calls see "Configuring Suspense Manager".

For details on the **pin_recycle** utility, see "pin_recycle".

About the Standard Recycling Mechanism

The BRM standard recycling feature uses the FCT_Reject, FCT_Suspense, and FCT_PreSuspense pipeline modules, along with the **pin_recycle** utility to suspend and recycle calls that originated in the same CDR input file. After examining the Pipeline Manager log files to determine why calls were suspended, Pipeline Manager administrators fix the pipeline, and then use this utility to attempt to rate these calls again. For details on setting up and using the standard recycling tools, see "Configuring Standard Recycling".

Configuring the pipeline requires system administration experience. You need to be familiar with:

- Modifying BRM pipeline modules to append EDRs with data. For details on setting up and administering pipeline rating, see "About Pipeline Rating".
- Creating a **crontab** file entry to run the **pin_recycle** utility to recycle or delete EDRs. See your operating system documentation for details on creating a **cron** command.

Setting Up EDR Recycling by CDR File

To set up BRM to recycle EDRs by CDR file:

1. Configure the pipeline to reject EDRs according to your business policies. For details, see "Configuring Standard Recycling".
2. Run **pin_recycle** utility with the **-f** option as needed to recycle suspended within a CDR file. You can test recycle, recycle, or delete all the failed EDRs contained in that CDR file. For the complete **pin_recycle** syntax, see "pin_recycle".

   You can run this utility like any other BRM utility, but you will probably want to run it manually as needed. How often you run this script depends on how many EDRs your pipeline rejects. When you make frequent or significant changes to your pipeline, you need to check your log files frequently. If a lot of EDRs are being rejected, you need to run **pin_recycle** often.

About Recycling Suspended EDRs after Rating Interruptions

Some BRM programs and features temporarily interrupt and then restart rating for certain accounts. These programs and features use **pin_recycle** to recycle calls for those accounts when the interruption is over. These features, such as account migration and pipeline-triggered billing, temporarily stop rating by directing the pipeline to suspend calls that come in during the interruption. As these call records arrive in the pipeline, they are appended with a recycle key. When the interruption is over, you use **pin_recycle** to rate all the stored calls that contain that recycle key. You can further configure this feature by using any number of different recycle keys to control when suspended EDRs get recycled.

Note: This feature is compatible with both Suspense Manager and standard recycling.

The **-k** *recycle key* option directs **pin_recycle** to search for all EDRs that contain a specific recycle key string and a status of **suspended**, and queues them for rating. The BRM feature that suspends EDRs determines which EDRs contain the same recycle key and need to be recycled together. This gives **pin_recycle** the flexibility to selectively restrict recycling to just the EDRs with specific characteristics.

For example, the account migration feature moves groups of accounts across databases, and must temporarily stop rating for each group of accounts while they are being moved. Account migration uses internal job IDs to keep track of the accounts being moved, and it also uses these job IDs in the recycle keys for suspended EDRs associated with those same accounts.

In contrast, the pipeline-triggered billing feature interrupts all rating for all accounts. Therefore, pipeline-triggered billing only needs to use one recycle key (**Trigger_Billing**) for all EDRs that arrive during the temporary suspension.

Before using **pin_recycle**, you must first configure a pipeline module to add the recycle key. For details, see "Setting Up EDR Recycling by Recycle Key".

Setting Up EDR Recycling by Recycle Key

To set up BRM to recycle suspended EDRs by recycle key:

1. Configure the pipeline to suspend EDRs according to your business policies. For details, see "Configuring Standard Recycling".

2. Configure BRM to add the recycle key to EDRs during the temporary interruptions. The feature requiring the temporary interruption has a pipeline module associated with it that does this. For example, the pipeline-triggered billing feature uses the FCT_TriggerBilling module, and Account Migration Manager uses the FCT_AccountRouter module.

   Important: AMM is not part of base BRM. Contact your BRM account manager for information about using AMM.

   The recycle key can be any string that corresponds to a set of EDRs to recycle. You configure a pipeline module to add your recycle key to the DETAIL.ASS_SUSPENSE_EXT.RECYCLE_KEY field of each EDR. The specific module to use depends on the program running billing and the strategy you use for recycling.

3. Configure **pin_recycle** to run periodically. You do this by adding it to a **cron** file. For details, see "Setting Up pin_recycle to Run Periodically".

4. Configure **pin_recycle** to run periodically with the **-d** option to remove the successfully recycled EDRs from the BRM database. You can do this by adding **pin_recycle** to a **cron** file. For details, see "Setting Up pin_recycle to Run Periodically".

Setting Up pin_recycle to Run Periodically

You need to run **pin_recycle** periodically both to queue the temporarily stored EDRs for rating, and to delete them. The **cron** command is the typical way to do this, although you can run **pin_recycle** like any other BRM command-line utility. This section explains how to set up **cron** command to run **pin_recycle**.

You need to add two **pin_recycle** entries to the **cron** command. One to search for and recycle EDRs, and the other to delete them after they are recycled. See "pin_recycle" for the syntax.

Adding EDR Recycle Entries

To run **pin_recycle** periodically, add entries like the following. The optimal frequency depends on your recycling strategy.

Running pin_recycle

Use a **cron** job with a **crontab** entry to run the **pin_recycle** script. The following **crontab** entry runs **pin_recycle** at 1:00 a.m. daily, and queues EDRs with a recycle key of **Trigger_Billing** for rating:

```
0 1 * * * BRM_Home/bin/pin_recycle -k Trigger_Billing &
```

BRM_Home is the directory where you installed BRM components.

Adding EDR Delete Entries

To remove EDRs from the BRM database, add an entry like the following. The optimal frequency depends on your recycling strategy.

Running pin_recycle

Use a **cron** job with a **crontab** entry to run the **pin_recycle** script. The following **crontab** entry runs **pin_recycle** at 1:00 a.m. daily, and deletes EDRs with a recycle key of **Trigger_Billing**:

```
0 1 * * * BRM_Home/bin/pin_recycle -k Trigger_Billing -d &
```

20

About Suspense Manager

This chapter provides an overview of the Oracle Communications Billing and Revenue Management (BRM) Suspense Manager features.

Important: Suspense Manager is an optional component, not part of base BRM.

Before using Suspense Manager, you should be familiar with the following topics:

- Pipeline Manager. See "About Pipeline Rating".
- Using Pipeline Manager to recycle EDRs. See "Configuring EDR Preprocessing".

Important: Suspense Manager is an extension of the BRM standard recycling feature. You must configure standard recycling first, before configuring Suspense Manager.

About Suspense Manager

You use Suspense Manager to:

- Analyze, edit, recycle, write off, archive, restore, and delete individual CDRs that have failed pipeline processing.
- Any number of individual call records at the same time (bulk processing).
- Analyze, resubmit, write off, and delete CDR files containing any number of individual call records. CDR files cannot be edited or archived.

Note: Suspense Management Center and certain BRM utilities and tools refers to CDR files as *batches* or *batch files*.

Suspense Manager includes the Suspense Management Center that allows you to perform these tasks using a graphical user interface.

Records "fail" if they cannot be processed by Pipeline Manager. These failures can be intentional or inadvertent. For example:

- A call record may arrive with invalid data, and fail a pipeline validity rule.
- The Pipeline Manager database tables may be set up incorrectly.
- The call record may fail custom validity checking that you have set up in a custom iScript, such as a size or time duration limit for individual records.

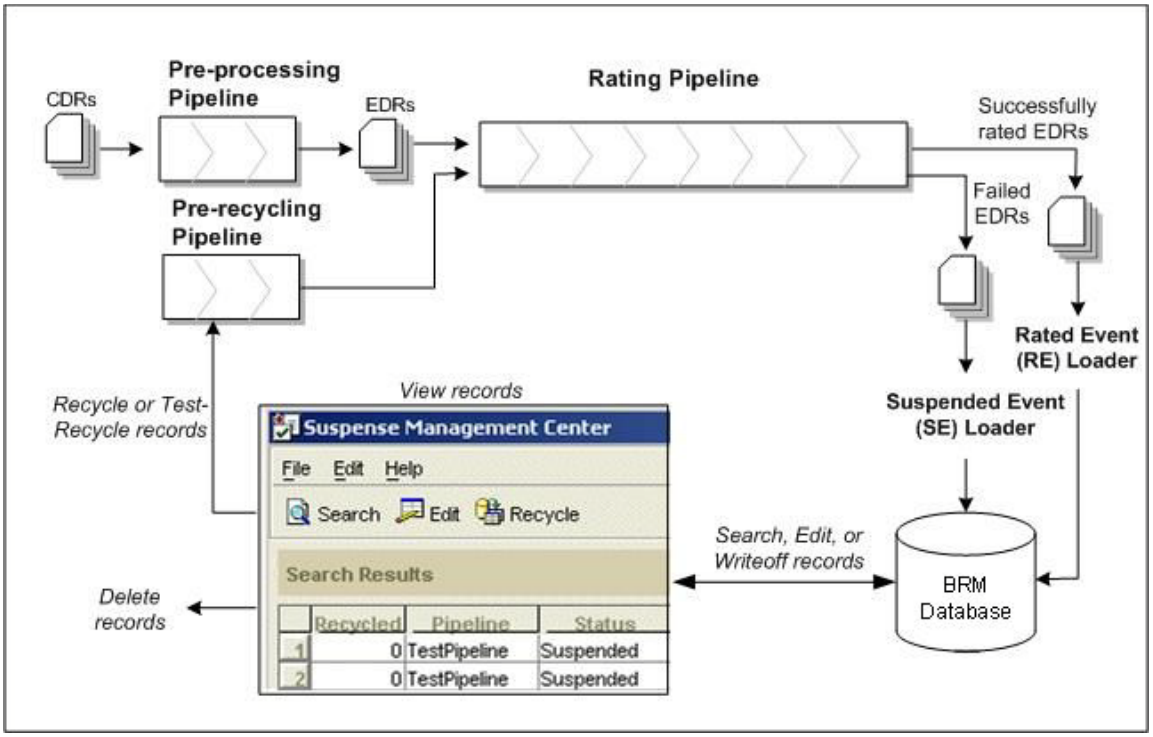
Suspense Manager replaces or augments the base BRM Standard Recycling feature for rejecting or recycling suspended calls.

Suspense Manager server components are available on the Solaris, Linux, HP-UX IA64, and AIX operating systems and require Oracle database software. The Suspense Management Center client application runs on Windows systems. For details on system requirements, see "Installing Suspense Manager".

Suspending Individual CDRs, or CDRs in Bulk

Figure 20–1 shows an example of how Suspense Manager can fit into your BRM system to manipulate individual failed CDRs, on groups of CDRs at once.

Figure 20–1 Suspense Manager and Individual Failed CDRs in BRM



In the example above, CDRs first enter Suspense Manager through a *preprocessing pipeline*. The preprocessing pipeline converts these records to the format used by Suspense Manager, EDRs. These records go through the preprocessing pipeline only once, and only a few modules are needed for it.

This example shows EDRs next going through a normal *rating pipeline*. Most pipeline function modules are included here. CDR "success" and "failure" policies are configured in this pipeline. If records "fail" in this pipeline, they are directed to the appropriate event loader to be loaded into the database.

The *SE Loader* converts the failed calls to objects in the BRM database.

Then, you manipulate these records by using the *Suspense Management Center* application. This application allows you to search for, edit, undo edits, test recycle, recycle, write off, or delete suspended CDRs or CDR files.

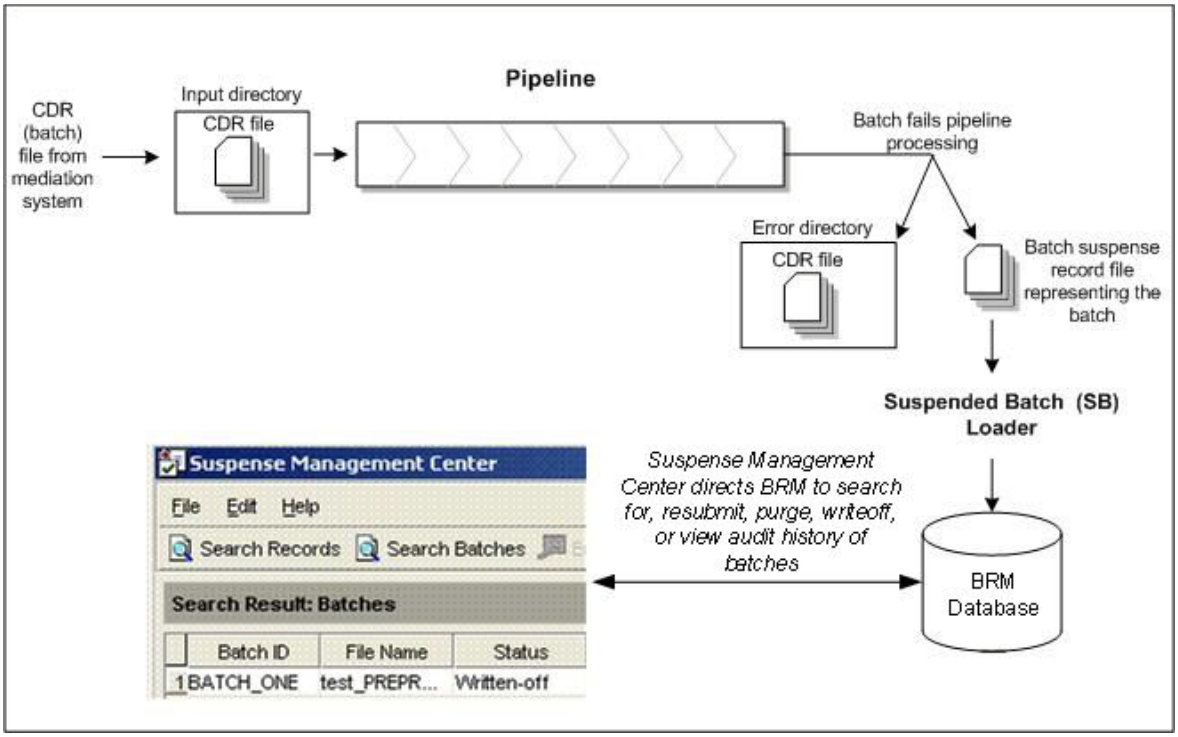
In this example, suspended records that get recycled are processed by the *pre-recycle pipeline* before they go through the rating pipeline again. Before recycling these records, you would probably make changes to the rating pipeline, or edit the records

so they are successfully rated when they go through the pipeline again. The pre-recycle pipeline converts the suspended record objects back into files that the pipeline can process, and routes the suspended records back through their original pipeline for recycling.

Suspending CDR Files

Figure 20–2 shows an example of how Suspense Manager can fit into your BRM system to manipulate files containing multiple CDRs.

Figure 20–2 Suspense Manager and Files Containing Multiple CDRs



This example shows a CDR file entering through a mediation system. The CDR file is first placed in the pipeline's input directory, and then processed by the pipeline. The pipeline contains "success" and "failure" policies based on file-level or record-level validation. If the CDR file fails pipeline processing, the CDR file itself is directed to the pipeline's error directory, and a **batch_suspense_create** file is created. SB Loader uses the information in this file to create an object which is stored in the BRM database.

You use the Suspense Management Center GUI to manipulate suspended CDR files by acting on the suspended CDR file objects. By using Suspense Management Center, you can resubmit CDR files through the pipeline, purge them, write them off, or view their audit histories.

If the problem with the CDR file is a bad pipeline policy or configuration, you can correct the pipeline and resubmit the CDR file for another attempt at processing.

If the problem is with the CDR file itself, you have several options. You can force the pipeline to ignore certain errors, and process the CDR file, or you can give up on the CDR file and purge it from the database. The list of pipeline errors to ignore is configurable and must be set up ahead of time. You cannot edit CDR files using Suspense Management Center.

Suspended Call Record States

As suspended records are processed by Suspense Manager, they are assigned one of the following states:

- **Suspended**: The call record could not be processed by Pipeline Manager and has been stored in the BRM database as a suspended call record.
- **Recycling**: The call record is being sent through the rating pipeline again to be rated.
- **Succeeded**: The call record has been successfully recycled and rated.
- **Written off**: The call will not be recycled, but is being stored for further use.

Table 20–1 lists the details about the states:

Table 20–1 Suspended Call Record States

State	PIN_FLD_ STATUS value in /suspended_ usage	Can be edited	Can be recycled	Can be written off	Can be deleted	Can be archived and deleted
Suspended	**0**	Yes	Yes	Yes	No	No
Recycling	**1**	No	No	No	No	No
Succeeded (Successfully processed)	**2**	No	No	No	Yes	Yes
Written off	**3**	No	No	No	Yes	Yes

About Suspended Event (SE) and Suspended Batch (SB) Loader

The Suspended Event (SE) and Suspended Batch (SB) loaders both load suspended records into the BRM database, but they operate on different types of records. The SE loader takes suspended (failed) CDRs as input and uses the **pin_rel** utility to load them into the BRM database as **/suspended_usage** objects. This utility is usually set up to run automatically, but can also be run manually as needed.

The SE loader is a special configuration of the Rated Event Loader (RE Loader), which loads prerated wireless events into the BRM database as objects.

The SB loader does not load CDR files directly into the BRM database. Instead, it accepts information from the **suspense_create_batch** file created for each failed CDR file, and creates **/suspended_batch** objects. The SB loader uses the **load_suspended_ batch_info.pl** script to create the **/suspended_batch** objects. This script is usually set up to run automatically, but can also be run manually as needed.

About the FCT_BatchSuspense Module

The FCT_BatchSuspense module adds suspense reason and suspense subreason codes to batches.

- If a resubmitted batch is successful, then FCT_BatchSuspense generates a batch_ suspense_update file with status as **Succeeded**. The SB loader reads this file and updates the corresponding /suspended_batch as **Succeeded** when you run **load_ suspended_batch_info.pl**.
- If a resubmitted batch fails again, FCT_BatchSuspense generates a batch_ suspense_update file with status as **Suspended**, new error code, and new suspense reason. The SB loader reads this file and updates the corresponding /suspended_batch object with a status of **Suspended**.

The specific errors that the FCT_BatchSuspense module adds are based on the error codes assigned to the EDR by the pipeline and the mapping information stored in the /config/batch_suspense_reason_code object. If no /config/batch_suspense_reason_code object is present, this module sets the suspense reason to **O** (other).

See "FCT_BatchSuspense".

Differences between the RE, SE, and SB Loaders

Table 20–2 explains the differences between the three event loaders:

Table 20–2 Differences Between Event Loaders

Task	RE Loader	SE Loader	SB Loader
Loads these types of records	Event file	CDRs	CDR files
Creates these objects	**/event**	**/suspended_usage**	**/suspended_batch**

Suspense Manager APIs

This section briefly describes the Suspense Manager components that you need to customize.

Suspense Manager Objects

Suspense Manager stores individual suspended CDRs in **/suspended_usage** objects, and suspended CDR file records in **/suspended_batch** objects. During configuration, you create a subclass of these objects for each type of call record you receive.

Every action performed by Suspense Management Center is recorded in these **/admin_action** objects:

- **/admin_action/suspended_usage/edit**
- **/admin_action/suspended_usage/recycle**
- **/admin_action/suspended_usage/writeoff**
- **/admin_action/suspended_batch**
- **/admin_action/suspended_batch/resubmit**
- **/admin_action/suspended_batch/writeoff**

For example, when you edit a number of suspended call records at the same time, Suspense Manager records the edits in an **/admin_action/suspended_usage/edit** object. All of the individual suspended CDRs have **/suspended_usage/***type* objects which reference that **/admin_action/suspended_usage/edit** object.

If you choose to override specific suspense reasons during recycling, the reasons available to override are stored in **/config/suspense_override_reason** objects.

For details on these Suspense Manager objects, see "Storable Class Definitions" in *BRM Developer's Reference*.

About Upgrading from Standard Recycling to Suspense Manager

To upgrade a system from standard recycling to the Suspense Manager features, follow the instructions in "Installing Suspense Manager" and "Configuring Suspense Manager".

Note: Suspense Manager is an optional component, not a part of base BRM.

For details on using Suspense Management Center with call records created using standard recycling, see "Using Suspense Management Center with Standard Recycling Call Records".

What's Next

The first step in setting up Suspense Manager is to install the Suspense Manager. software. For details see "Installing Suspense Manager".

21

Installing Suspense Manager

This chapter explains how to install the Oracle Communications Billing and Revenue Management (BRM) Suspense Manager software.

For information about Suspense Manager, see "About Suspense Manager".

Important: Suspense Manager is an optional feature that requires a separate license.

System Requirements

The Suspense Manager server component is supported on the HP-UX IA64, Linux, AIX, and Solaris operating systems. For information on disk space requirements for these operating systems, see "Disk space requirements" in *BRM Installation Guide*.

The Suspense Management Center application is supported on the Windows platform and requires approximately 120 MB of disk space.

Note: Suspense Management Center requires the Java Runtime Environment (JRE), which is included in the Suspense Management Center package and is approximately 50 MB. If the JRE was already installed with another BRM client application, it will not be reinstalled.

Software Requirements

Before installing Suspense Manager, you must install:

- Third-Party software, which includes the PERL libraries and JRE required for installing BRM components. See "Installing the Third-Party Software" in *BRM Installation Guide*.
- The base BRM software.
- Rated Event (RE) Loader, which is an optional BRM component. For details, see "About Loading Prerated Events".

If you are installing all Suspense Manager components on one system (typical for a test system), you need the information for that system. If you are installing the Suspense Manager Connection Manager (CM) on a different system (common for a production system), you need the appropriate information for that system.

To install the Suspense Management Center, you need information about the machine running the CM to which Suspense Management Center will connect such as:

- Computer or server name
- Port number
- Database number

Installing Suspense Management Center

To install the Suspense Management Center client software:

1. Download the software to a temporary directory *(temp_dir)*.
2. Extract the files in the **.zip** file to any temporary directory.
3. Go to the temporary directory where you extracted the client application download file.
4. Double-click **Setup.exe** to run it.
5. On the Choose Destination Location screen, select the destination folder for the program and click **Next**.
6. For **Install Type**, select the type of installation you want, **standalone** or **Web Start**.
   - **Standalone** is the default setting, click **Next** if that is the type of installation you need.
   - To use **Web Start** features, click **Web server** and click **Next**.
7. On the Connection Manager screen, enter the server, port, and database information and click **Next**.
8. On the Select Program Folder screen, accept the default program folder, **Portal**, or select a new one and click **Next**.
9. On the Start Copying Files screen, click **Next** to start installing the application.
10. Click **Finish** to complete the installation process.

Starting and Using Suspense Management Center

From the start options, choose **Suspense Management Center**, which is under **Portal**.

Note:

- When prompted, use the BRM default login, **root.0.0.0.1**, and default password, **password**. All BRM users can change their logins and passwords later.
- Click the Suspense Management Center **Help** button to display information on how to use Suspense Management Center.

Installing Java Web Start and Downloading Suspense Management Center

If you installed the Web Server version of Suspense Management Center, follow the instructions in this section.

To set up Suspense Management Center on client systems from the Web server, go to this URL in your Web browser:

http://*machine_name***/SuspenseManagement_en.html**

Replace *machine_name* with the name of the system running the Web server and Suspense Management Center. If Suspense Management Center is located in a subdirectory of your Web server's document root directory, include the full path to **SuspenseManagement_en.html**.

If your Web server uses a port number other than the default of 80, include the port number in the URL:

http://*machine_name***:***port_number***/SuspenseManagement_en.html**

For example, if Suspense Management Center is on a system called server1 using port 81, go to this URL: **http://server1:81/SuspenseManagement_en.html**.

Important: The first time you start Suspense Management Center on a Windows client system, you download and install the JRE, which includes Java Web Start.

Note:

- If your Web server uses default port 80, specifying the port number is optional. Otherwise, you must include the port number.
- If your Web server doesn't display the page correctly, make sure you set the JNLP MIME type in your Web server.
- If the MIME type is set in your Web server, try clearing the cache in your Web browser. For Internet Explorer, delete temporary Internet files.
- If you can't see Suspense Management Center displayed in Java Web Start, choose **File - Preferences** in Java Web Start Application Manager. Under the **Advanced** tab, change the **Remote Application URL** field to the Suspense Management Center URL.

For more information on using Java Web Start, see the Java Web Start documentation at http://www.oracle.com/technetwork/java/index.html.

Installing Suspense Manager Server Components

Note: If you have already installed the product, features that are already installed cannot be reinstalled without uninstalling them first. To reinstall a feature, uninstall it and then install it again.

To install the Suspense Manager server components:

1. Download the software to a temporary directory *(temp_dir)*.

Important:

- If you download to a Windows workstation, use **FTP** to copy the .bin file to a temporary directory on your UNIX server.
- You must increase the heap size used by the Java Virtual Machine (JVM) before running the installation program to avoid "Out of Memory" error messages in the log file. For information, see "Increasing heap size to avoid "Out of Memory" error messages" in *BRM Installation Guide*.

2. Go to the directory where you installed the Third-Party package and source the **source.me** file.

   Caution: You must source the **source.me** file to proceed with installation, otherwise "suitable JVM not found" and other error messages appear.

   Bash shell:

   ```
   source source.me.sh
   ```

   C shell:

   ```
   source source.me.csh
   ```

3. Go to the *temp_dir* directory and enter this command:

   ```
   7.5.0_SuspenseMgr_platform_opt.bin
   ```

   where *platform* is the operating system name.

   Note: You can use the **-console** parameter to run the installation in command-line mode. To enable a graphical user interface (GUI) installation, install a GUI application such as X Windows and set the DISPLAY environment variable before you install the software.

4. (Optional) If you want to install Suspense Manager server components separately, either on this computer or on another computer, select custom install when asked to specify the setup type. Select the components you are installing by typing their respective numbers and click **Next**. The components are:
   - **SuspenseMgr DM**
   - **SuspenseMgr CM**
5. Follow the instructions displayed during installation. The default installation directory for Suspense Manager is **opt/portal/7.5**.

   Note: The installation program does not prompt you for the installation directory if BRM or Suspense Manager is already installed on the machine and automatically installs the package at the *BRM_Home* location *BRM_Home* is the directory where you installed BRM components.

6. Go to the directory where you installed the Suspense Manager package and source the **source.me** file:

   Bash shell:

   ```
   source source.me.sh
   ```

   C shell:

   ```
   source source.me.csh
   ```

7. Go to the *BRM_Home*/**setup** directory and run the **pin_setup** script.

   Note: The **pin_setup** script starts all required BRM processes.

Your Suspense Manager server installation is now complete. See "What's Next?".

Uninstalling Suspense Manager

To uninstall Suspense Manager, run the *BRM_Home*/**uninstaller/SuspenseMgr/uninstaller.bin**.

What's Next?

The next step is to configure Suspense Manager for your business requirements. See "Configuring Suspense Manager".

22

Configuring Suspense Manager

This chapter explains how to set up Oracle Communications Billing and Revenue Management (BRM) Suspense Manager.

Before you read this document, you should be familiar with:

- BRM system administration.
- Pipeline Manager and how to set up pipeline modules. See these documents:
  - About Pipeline Rating
  - Configuring Pipeline Rating
  - Configuring EDR Input Processing
  - Configuring EDR Preprocessing
- Editing **pin.conf** configuration files and using file loading utilities. See "Using Configuration Files to Connect and Configure Components" in *BRM System Administrator's Guide*.

About Configuring Suspense Manager

The business decisions you make in the planning phase determine the details of your implementation in the configuration phase.

Table 22–1 shows the tasks required for configuring Suspense Manager:

Table 22–1 Tasks to Configure Suspense Manager

Task	Description
Plan and set up your database. See "Planning and Setting up Your Database for Suspense Manager".	▪ Decide whether to extend the **/suspended_usage** class. ▪ Select a list of queryable EDR fields. ▪ Add **/suspended_usage** subclasses with queryable fields.
Create a list of editable fields. See "Creating a List of Editable Fields Based on Your /suspended_usage Subclasses".	▪ Create a list of fields that you want the ability to correct using Suspense Manager.
Load editable fields into the database. See "Loading Editable Fields into the Database".	▪ Edit the **pin_suspense_editable_flds** file. ▪ Run the **load_pin_suspense_editable_flds** utility.

Table 22–1 (Cont.) Tasks to Configure Suspense Manager

Task	Description
Change suspense reasons and subreasons (optional). See "Changing the List of Suspense Reasons and Subreasons".	■ Decide whether to change the suspense reason lists. ■ Edit the **suspense_reason_code.en_US** file or the **batch_suspense_reason_code.en_US** file. ■ Edit the **pin_suspense_reason_code** or **pin_batch_suspense_reason_code** file. ■ Run the **load_localized_strings** utility. ■ Run the **load_pin_suspense_reason_code** or **load_pin_batch_suspense_reason_code** utility.
Configure the pipeline for Suspense Manager. See "Configuring Pipeline Manager for Suspense Manager".	■ Configure the standard recycling pipeline. ■ Configure the rating pipeline. ■ Configure the pre-cycling pipeline.
Configure SE or SB Loader. See "Setting Up Suspended Event (SE) Loader for Suspense Manager". See "Setting Up Suspended Batch (SB) Loader for Suspense Manager".	■ Edit the **Infranet.properties** file.
Create indexes for search templates. See "Creating Indexes for Search Templates".	■ Create indexes.
Configure and customize Suspense Management Center. See "Configuring and Customizing Suspense Management Center".	■ Set permissions using Customer Center. ■ Add custom fields (edit the **custom.properties** file). ■ Add custom fields to Web Start (edit **SuspenseManagement_en.jnlp**) (Optional). ■ Set up permissions for Suspense Management Center.
Configuring event notification for Suspense Manager. See "Configuring Event Notification for Suspense Manager".	■ Consolidate event notification operations.
Configure debugging (optional). See "Configuring Debugging (Optional)".	■ Set up Java PCM logging (edit the **Infranet.properties** file). ■ Edit **RunSM.bat**.
Configure the number of records to process in a transaction (optional). See "Configuring the Number of Suspended Records to Process in a Transaction".	■ Edit the **pin_suspense_params** file and run **load_suspense_params**.

Planning and Setting up Your Database for Suspense Manager

> **Important:** The planning process is critical to the successful operation of Suspense Manager. You will be making database schema changes during setup. Changing the database schema *after* you start using Suspense Manager requires a database upgrade that can be time consuming and painful. Be sure to follow the steps in this section carefully.

The process of setting up your database involves these two steps:

- Picking the EDR fields you will use to search for suspended EDRs (queryable fields). If the queryable fields your implementation requires are not in the default BRM objects, you need to create new objects subclasses for them.
- Picking a list of fields that you will allow your personnel to edit. BRM assumes that this list is a subset of the default EDR fields and any new queryable fields. If not, then you will need to create new objects for them.

Once you have decided on these fields to add and/or edit in EDRs, you will then load the list into the BRM database. This will allow suspense Manager to access them.

Deciding Whether You Need to Extend the Suspense Subclasses

Your business will require you to search for and analyze suspended call records (individual CDRs or CDR files). The first step in setting up Suspense Manager is to decide whether the default Suspense Manager storable classes meet your needs. The storable classes you use must contain the fields your business requires to search for the records you need, and analyze their problems. If not, you will need to modify, extend, or replace the default storable classes.

Selecting a List of Queryable EDR Fields

This is a list of EDR fields that you will use to search for and analyze call records. Suspense Management Center allows you to search for suspended calls based on values in these queryable fields, and displays these values in your search results.

> **Important:** After you specify and load the list of queryable fields into the database, modifying the list involves significant effort. Be sure to plan the list carefully.

Each BRM implementation requires a different list. Making all of your EDR fields queryable degrades performance by using a lot of unnecessary database storage. However, you do need to make enough fields queryable as necessary to meet your business needs.

Start by reviewing the object specifications for the sample Suspense Manager storable classes listed below. If these storable classes contain all the information your business requires, you won't need to make any changes and you can skip the next section. If you need to expand or extend these classes, make a list of the fields you want to make queryable. You will use this list to define custom extensions to **/suspended_usage** object types in the next section.

Note: You add one set of queryable fields representing one **/suspended_usage** subclass *per pipeline*. For example, for a single pipeline that accepts **/suspended_usage/telco/gsm** records, you can select queryable fields from the **/suspended_usage/telco** and **/suspended_usage/telco/gsm** subclasses. You cannot select queryable fields from **/suspened_usage/telco/gprs**, because it requires a separate pipeline.

/suspended_usage/telco

This storable class stores general wireless call record data in the fields listed in Table 22–2:

Table 22–2 /suspended_usage/telco Storable Class

Field	Description
PIN_FLD_BYTES_IN	Volume of data sent.
PIN_FLD_BYTES_OUT	Volume of data received.
PIN_FLD_CALLED_TO	Phone number being called.
PIN_FLD_CALLING_FROM	Phone number the call originated from.
PIN_FLD_CALL_DURATION	Call time duration.
PIN FLD PRIMARY MSID	Primary MSID.
PIN FLD SERVICE TYPE	Basic service type.
PIN_FLD_START_TIME	Time the call was initiated. **Note**: The start time is not stored in UTC format.
PIN_FLD_USAGE_TYPE	Describes the call scenario, for example, customer-to-customer call, birthday call, closed-user-group call, or friends & family call.

/suspended_usage/telco/gprs

This class stores call record data for generic GPRS (data) calls in the fields listed in Table 22–3:

Table 22–3 /suspended_usage/telco/gprs Storable Class

Field	Description
PIN_FLD_BYTES_IN	Volume of data sent.
PIN_FLD_BYTES_OUT	Volume of data received.
PIN_FLD_CALLED_TO	Phone number being called.
PIN_FLD_CALLING_FROM	Phone number the call originated from.
PIN_FLD_CALL_DURATION	Call time duration.
PIN FLD PRIMARY MSID	Primary MSID.
PIN FLD SERVICE TYPE	Basic service type.
PIN_FLD_START_TIME	Time the call was initiated. **Note**: The start time is not stored in UTC format.

Table 22–3 (Cont.) /suspended_usage/telco/gprs Storable Class

Field	Description
PIN_FLD_USAGE_TYPE	Describes the call scenario, for example, customer-to-customer call, birthday call, closed-user-group call, or friends & family call.
PN_FLD_APN	APN address.
PIN_FLD_GGSN_ADDRESS	GGSN address.
PIN_FLD_NODE_ID	Node ID.
PIN_FLD_SECONDARY_MSID	MSI number.
PIN_FLD_SGSN_ADDRESS	SGSN address.

/suspended_usage/telco/gsm

This class stores call record data for generic GSM (voice) calls in the fields listed in Table 22–4:

Table 22–4 /suspended_usage/telco/gsm Storable Class

Field	Description
PIN_FLD_BYTES_IN	Volume of data sent.
PIN_FLD_BYTES_OUT	Volume of data received.
PIN_FLD_CALLED_TO	Phone number being called.
PIN_FLD_CALLING_FROM	Phone number the call originated from.
PIN_FLD_CALL_DURATION	Call time duration.
PIN FLD PRIMARY MSID	Primary MSID.
PIN FLD SERVICE TYPE	Basic service type.
PIN_FLD_START_TIME	Time the call was initiated. **Note**: The start time is not stored in UTC format.
PIN_FLD_USAGE_TYPE	Describes the call scenario, for example, customer-to-customer call, birthday call, closed-user-group call, or friends & family call.
PIN_FLD_CELL_ID	Network cell ID where the call originated.
PIN_FLD_DESTINATION_SID	Destination MSC or switch ID.
PIN_FLD_DIALED_NUMBER	Number that was called.
PIN_FLD_ORIGIN_SID	Origin MSC or switch ID.
PIN_FLD_SECONDARY_MSID	IMSI field.

All of the fields in subclasses of **/suspended_usage** are queryable.

Adding /suspended_usage Subclasses with Queryable Fields

Suspense Manager uses the **/suspended_usage** storable class to store failed call records in the BRM database. You must extend this class for each type of suspended call record that your business requires. Suspense Manager provides the **/suspended_usage/telco**, **/suspended_usage/telco/gsm**, and **/suspended_usage/telco/gprs** default subclasses to store data for suspended calls.

To add new types of call records to Suspense Manager:

1. Determine how to subclass **/suspended_usage** objects.
2. Create custom **/suspended_usage** objects.

   Use the Storable Class Editor to add a subclass to **/suspended_usage** for each type of call record your business uses.

   For more information, see "Creating Custom Fields And Storable Classes" in *BRM Developer's Guide*.

Creating a List of Editable Fields Based on Your /suspended_usage Subclasses

You use Suspense Management Center to correct these fields in failed EDRs, and then recycle and rate the calls.

Review these objects and create the list of fields you need to change to correct a failed call. All of the fields you added to **/suspended_usage** subclasses are eligible for editing. You can change this list any time by following the steps in "Loading Editable Fields into the Database".

Loading Editable Fields into the Database

To load your list of editable fields into the database for use by Suspense Management Center:

1. Review the list of fields you assembled in "Creating a List of Editable Fields Based on Your /suspended_usage Subclasses".
2. Add these fields to the *BRM_Home***/sys/data/config/pin_suspense_editable_flds** file. *BRM_Home* is the directory where you installed BRM components.
3. Run the **load_pin_suspense_editable_flds** utility (located in *BRM_Home***/bin)** to load the editable fields into the database:

   ```
   %load_pin_suspense_editable_flds pin_suspense_editable_flds
   ```

   For details, see "load_pin_suspense_editable_flds".

Changing the List of Suspense Reasons and Subreasons

Suspense Manager adds the specific reasons for call failures to the EDRs it stores. These reasons, called *suspense reasons,* can be divided into more specific *suspense subreasons*. These suspense reasons and subreasons are stored in the call record along with the rest of the call record data. Because they are stored in the call records, you can search for them by using Suspense Management Center. For example, you can search for all the calls that could not be associated with a subscriber.

The Pipeline Manager error messages that actually cause call failures are mapped to these suspense reasons and subreasons. An extensive default error code-to-suspense reason mapping is provided in Suspense Manager. If your business requires different suspense reasons or subreasons, you can change them and their mappings. You can make these changes at any time, but because you may have to upgrade existing data, it is best to have this mapping in place before you go into production.

Deciding whether to Change the Suspense Reason and Subreason Lists

Each suspense reason covers a group of Pipeline Manager error codes. The strings that define these error messages are listed in the **suspense_reason_code.en_US** file (in the *BRM_Home*/**sys/msgs/suspense_reason_code** directory). These error code strings are mapped to Pipeline Manager error codes in the **pin_suspense_reason_code** file (in the *BRM_Home*/**sys/data/config** directory).

If you need to change the default mapping or add additional reasons or subreasons, continue with the instructions in "Changing the Suspense Reason and Subreason Lists" that describe the process. If the default suspense reasons and subreasons are appropriate for your business, then skip the rest of this section.

Changing the Suspense Reason and Subreason Lists

If the default error messages or error message mappings do not meet your business needs, follow the steps in this section to change them. You first edit the text file with your new suspense reasons and subreasons, and then load the mapping into the database.

1. Determine your new suspense reasons.

   This is a list of the most common problems that cause calls to fail. It can be as extensive as you like. You can also look at these as *categories* of suspense reasons, because this is the first of two levels. For example, on this level you might use "Validation check failed," then use the next step to add more specific subreasons, such as "Call exceeds maximum time" or "Suspiciously long call."

   Note: Any Pipeline Manager error message without a suspense reason will use the default Pipeline Manager error message.

2. Determine any new suspense subreasons.

3. Edit the *BRM_Home*/**sys/msgs/suspense_reason_code.en_US** file, adding suspense reasons as strings and mapping them to integers.

   Sample entry for a suspense reason with the ID of 1:

```
STR
      ID = 1 ;
      VERSION = 1 ;
      STRING - "Unable to identify customer information" ;
END
```

   Sample entry for a suspense subreason for a suspense reason with the ID of 2:

```
DOMAIN = "suspense_subreason_1" ;
STR
      ID = 2;
      VERSION = 1 ;
      STRING = "B number missing" ;
END
```

   Note: The default string has an ID of 0. This string appears by default in the case of an error that is not mapped to a suspense reason.

Important: The reason code numbers 65535 and 65534 are reserved for use by BRM.

The format of the **suspense_reason_code.***locale* file is similar to that of the reasons.locale file.

4. Map your suspense reasons to Pipeline Manager error messages by editing the **pin_suspense_reason_code** (for CDRs) or **pin_batch_suspense_reason_code** (for CDR files) file in the *BRM_Home***/sys/data/config** directory.

   Excerpt from the default version of **pin_suspense_reason_code**:

```
#ErrorCodes         SuspenseReason       SuspenseSubReason

# Default error (error that is not specified in this file)
00000                    0                    0

# Framework errors
00464                    1                    1
00479                    1                    1
00496                    1                    1
00497                    1                    1
00480                    1                    1
00481                    1                    1
00482                    1                    1

00208                    9                    0
00209                    7                    0
```

5. Load your localized strings into the database by using the **load_localized_strings** utility.

   Important: The **load_localized_strings** utility requires a configuration file to function correctly. For details, see "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

   Example syntax:

```
%load_localized_strings suspense_reason_code.en_US
```

   Note: If you're loading a localized version of this file, use the correct file extension for your locale. For a list of file extensions, see "Locale Names" in *BRM Developer's Guide*.

6. Load your suspense reason code mapping into the database by using the **load_pin_suspense_reason_code** or **load_pin_batch_suspense_reason_code** utility (in the *BRM_Home***/bin** directory). For details, see "load_pin_suspense_reason_code" or "load_pin_batch_suspense_reason_code".

Important: The **load_pin_suspense_reason_code** and **load_pin_batch_suspense_reason_code** utilities require configuration files to function correctly. For details, see "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

Example syntax for CDRs:

```
%load_pin_suspense_reason_code pin_suspense_reason_code
```

Example syntax for CDR files:

```
%load_pin_batch_suspense_reason_code pin_batch_suspense_reason_code
```

7. Verify that the strings were loaded by displaying the **/strings** objects using the Object Browser or the **robj** command with the **testnap** utility. See "Reading objects by using Object Browser" in *BRM Developer's Guide* for information on how to use Object Browser. See "Using Testnap" in *BRM Developer's Guide* for general instructions on using **testnap**.
8. Stop and restart the Connection Manager (CM). See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.
9. Stop and restart Suspense Management Center.

Your suspense reason and subreason strings are now loaded into the BRM database to be displayed and used by Suspense Management Center.

Configuring Pipeline Manager for Suspense Manager

The following Pipeline Manager configuration steps are required for Suspense Manager.

Configuring a Standard Recycling Pipeline

Follow the instructions in "Configuring Standard Recycling" for your initial Pipeline Manager setup. The Suspense Manager Pipeline Manager configuration builds on the steps used to create a standard recycling pipeline.

Configuring a Rating Pipeline

The following configuration steps are required to configure a rating pipeline.

Configuring FCT_PreSuspense

You added FCT_PreSuspense as the first function module of the pipeline during standard recycling configuration. For details, see "Configuring Standard Recycling".

The FCT_PreSuspense registry requires additional configuration for Suspense Manager. This module requires information from the object class definition you created in "Adding /suspended_usage Subclasses with Queryable Fields".

The example FCT_PreSuspense registry below shows queryable fields for the **/suspended_usage/telco**, **/suspended_usage/telco/gsm**, and **/suspended_usage/telco/gprs** objects.

```
PreSuspense
                {
                  ModuleName                = FCT_PreSuspense
```

```
Module
{
   Active                  = True
   QueryableFields
   {
     # table name. If more than one table, use a separate block
     SUSP_USAGE_TELCO_INFO_T
     {
        # format : <database_column_name> = <edr_conatiner_field_name>
        BYTES_IN = DETAIL.VOLUME_RECEIVED
        BYTES_OUT = DETAIL.VOLUME_SENT
        CALLED_TO = DETAIL.B_NUMBER
        #CALLING_FROM = DETAIL.B_NUMBER
        CALL_DURATION = DETAIL.DURATION
        PRIMARY_MSID = DETAIL.A_NUMBER
        SERVICE_TYPE = DETAIL.BASIC_SERVICE
        START_TIME = DETAIL.CHARGING_START_TIMESTAMP
        USAGE_TYPE = DETAIL.USAGE_TYPE
     }
     SUSP_USAGE_TELCO_GPRS_INFO_T
     {
         # format : <database_column_name> = <edr_conatiner_field_name>
         APN = DETAIL.ASS_GPRS_EXT.APN_ADDRESS
         GGSN_ADDRESS = DETAIL.ASS_GPRS_EXT.GGSN_ADDRESS
         NODE_ID = DETAIL.ASS_GPRS_EXT.NODE_ID
         SECONDARY_MSID = DETAIL.ASS_GPRS_EXT.PORT_NUMBER
         SGSN_ADDRESS = DETAIL.ASS_GPRS_EXT.SGSN_ADDRESS
     }
     SUSP_USAGE_TELCO_GSM_INFO_T
     {
        APN = DETAIL.ASS_GSMW_EXT.APN_ADDRESS
        CELL_ID = DETAIL.ASS_GSMW_EXT.CELL_ID
         DESTINATION_SID = DETAIL.ASS_GSMW_EXT.TERMINATING_SWITCH_IDENTIFICATION
        DIALED_NUMBER = DETAIL.ASS_GSMW_EXT.DIALED_DIGITS
        ORIGIN_SID = DETAIL.ASS_GSMW_EXT.ORIGINATING_SWITCH_IDENTIFICATION
                     SECONDARY_MSID = DETAIL.ASS_GSMW_EXT.PORT_NUMBER
     }
   }
  }
}
```

For details on the syntax, see "FCT_PreSuspense".

Configuring SuspenseCreateOutput

Configure the suspense create output module as one of the output modules for this pipeline.

You can use the sample output module in "Configuring Standard Recycling" in a rating pipeline with one change. You need to change the **EventType** entry from **/suspended_usage** to **/suspended_usage/***type*, where *type* is the subclass you created in "Adding /suspended_usage Subclasses with Queryable Fields".

This example shows the **/suspended_usage/telco** being used:

```
SuspenseCreateOutput

{
   ModuleName                  = OUT_GenericStream

   EventType = /suspended_usage/telco
```

. . .

Configuring a Pre-Recycling Pipeline

You configured a pre-recycling pipeline as part of standard recycling configuration.

The pre-recycling pipeline uses the INP_Recycle module. This module reads suspended usage records from the BRM database, restores original EDRs, applies edits to them, and pushes EDRs into the pre-recycling pipeline.

For Suspense Manager, the INP_Recycle module does the following:

- Sets the process status to either recycling or test recycling, depending on the processing mode selected in Suspense Management Center.
- Sets override reasons in the DETAIL.ASS_SUSPENSE_EDT.OVERRIDE_REASONS field, and the batch ID to DETAIL.ORIGINAL_BATCH_ID.
- Provides feedback to DAT_Recycle about the status of recycling (commit, cancel, or rollback).
- Takes the original batch ID (from a routing switch, mediation system, or other source) from the /suspended_usage object and copies it to DETAIL.ORIGINAL_BATCH_ID. This module also creates a new batch ID for each batch of recycled records, and set it in the HEADER.BATCH_ID and DETAIL.BATCH_ID fields of those records. FCT_PreSuspense appends DETAIL.BATCH_ID with more information to ensure that it remains unique.

To configure a pre-recycling pipeline, see "Configuring a Pre-Recycling Pipeline". These additional steps are also required for Suspense Manager:

1. In the INP_Recycle module, change this EXT_InEasyDB entry:

   ```
   SqlDetail    = StdRecycleDetail.sql
   ```

   to this:

   ```
   SqlDetail    = RecycleDetail.sql
   ```

   This file is used by EXT_InEasyDB for reading queryable fields in **/suspended_usage** objects.

2. Edit the *Pipeline_Home***/database/Oracle/Scripts/Suspense/RecycleDetail.sql** script. *Pipeline_Home* is the directory where you installed Pipeline Manager.
   - Add any custom queryable fields that you added in "Adding /suspended_usage Subclasses with Queryable Fields".
   - Remove any non-editable fields from the SELECT statement to improve performance.

Setting Up Suspended Event (SE) Loader for Suspense Manager

This section explains the configuration steps necessary to load suspended CDRs into the BRM database. You used the steps in "Configuring SE Loader for Standard Recycling" to configure the SE Loader for standard recycling. Follow these steps to configure your **Infranet.properties** file and store your Suspense Manager customizations in **/suspended_usage** objects:

1. Append the contents of **suspense_Infranet.properties** to your **Infranet.properties** file:

```
% cd BRM_Home/apps/pin_rel
% cat suspense_Infranet.properties Infranet.properties
```

2. Edit the *BRM_Home*/**apps/pin_rel/Infranet.properties** file to create new event types for each of your **/suspended_usage** or **/suspended_batch** subclasses and for temporary objects. Use the **/suspended_usage/telco** section of *BRM_Home*/**apps/pin_rel/Infranet.properties** as a guide.

   You use the queryable fields you set up in "Selecting a List of Queryable EDR Fields" in this step.

Setting Up Suspended Batch (SB) Loader for Suspense Manager

This section explains the SB Loader configuration steps necessary to load CDR files into the BRM database.

Follow these steps to configure SB Loader:

1. Add FCT_BatchSuspense as the first functional plugin of the pre-processing pipeline. See "FCT_BatchSuspense".

   Note: This module can be added to any pipeline doing file-level validation, but this is most likely the pre-processing pipeline.

   Specify the object you will use to store suspended CDR file information using the **StorableClass** registry entry in FCT_BatchSuspense. The default object is **/suspended_batch/cdr**. For details, see "FCT_BatchSuspense".

2. Add these entries to your CM's **pin.conf** file to provide connection to the database to store your suspended CDR files:

```
- nap cm_ptr ip_host port_no
- nap login_type 1
- - userid 0.0.0.1 /service/pcm_client 1
- nap login_name root.0.0.0.1
- nap login_pw password
```

Creating Indexes for Search Templates

By default, Suspense Manager does not include any database indexes for searches other than indexes based on POID IDs. You can improve database performance by creating indexes for your most common searches. The example below guides you through the process.

Tip: If there are many indexes on the tables for **/suspended_usage** objects, you run the risk of degrading SE Loader performance during bulk loading of **/suspended_usage** objects. Experiment to find the right balance of indexes for your system.

Example search template:

```
#Suspense Management Template
#Fri Nov 14 09:16:53 PST 2003
PIN_FLD_CALL_DURATION.max=
PIN_FLD_SUSPENSE_REASON.value=<All>
PIN_FLD_CALL_DURATION.selected=false
```

```
PIN_FLD_EDITED.value=<All suspended>
PIN_FLD_TEST_SUSPENSE_SUBREASON.value=<All>
PIN_FLD_RECORD_TYPE.selected=true
PIN_FLD_FILENAME.selected=true
PIN_FLD_TEST_ERROR_CODE.min=
PIN_FLD_STATUS.value=Suspended
PIN_FLD_RECORD_TYPE.text=
PIN_FLD_PIPELINE_NAME.text= datadictionary.class=/suspended_usage/telco
PIN_FLD_PIPELINE_ERROR_CODE.max= PIN_FLD_TEST_SUSPENSE_SUBREASON.selected=false
PIN_FLD_SUSPENSE_SUBREASON.selected=false
PIN_FLD_SERVICE_CODE.text=
PIN_FLD_NUM_RECYCLES.max=0
PIN_FLD_PIPELINE_NAME.selected=false
PIN_FLD_SUSPENSE_REASON.selected=true
PIN_FLD_TEST_SUSPENSE_REASON.value=<All>
PIN_FLD_START_TIME.selected=false
PIN_FLD_CALLING_FROM.text=
PIN_FLD_CALL_DURATION.min=
PIN_FLD_NUM_RECYCLES.selected=true
PIN_FLD_FILENAME.text=
PIN_FLD_EDITED.enabled=true
PIN_FLD_CALLED_TO.selected=true
PIN_FLD_STATUS.selected=false
PIN_FLD_TEST_SUSPENSE_REASON.selected=false
PIN_FLD_RECYCLE_T.selected=false
PIN_FLD_TEST_ERROR_CODE.selected=false
PIN_FLD_CALLING_FROM.selected=true
PIN_FLD_CALLED_TO.text=
PIN_FLD_TEST_ERROR_CODE.max=
PIN_FLD_BATCH_ID.selected=false
PIN_FLD_BATCH_ID.text=
PIN_FLD_PIPELINE_ERROR_CODE.min=
PIN_FLD_SERVICE_CODE.selected=false
PIN_FLD_EDITED.selected=false
PIN_FLD_SUSPENSE_SUBREASON.value=<All>
PIN_FLD_NUM_RECYCLES.min=0
PIN_FLD_PIPELINE_ERROR_CODE.selected=true
```

The example search template translates into this SQL statement:

```
SQL> select st.called_to, st.calling_from, s.filename, s.error_code,
SQL> s.suspense_reason, s.num_recycles from suspended_usage_t s,
SQL> susp_usage_telco_info_t st where s.status = 0 and s.num_recycles
SQL> >= 0 and s.num_recycles <= 0 and s.poid_id0 = st.obj_id0;
```

For Oracle databases, use the statements below to determine which indexes would improve performance.

To evaluate this SQL statement, turn on autotrace and run this statement.

This is the output:

```
SQL> set autotrace on;
SQL> select st.called_to, st.calling_from, s.filename, s.error_code,
SQL> s.suspense_reason, s.num_recycles from suspended_usage_t s,
SQL> susp_usage_telco_info_t st where s.status = 0 and s.num_recycles
SQL> >= 0 and s.num_recycles <= 0 and s.poid_id0 = st.obj_id0;
...
13 rows selected.

Execution Plan
----------------------------------------------------------
```

```
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   NESTED LOOPS
   2    1     TABLE ACCESS (FULL) OF 'SUSP_USAGE_TELCO_INFO_T'
   3    1    TABLE ACCESS (BY INDEX ROWID) OF 'SUSPENDED_USAGE_T'
   4    3    INDEX (UNIQUE SCAN) OF 'I_SUSPENDED_USAGE__ID' (UNIQUE)

Statistics
----------------------------------------------------------
         176  recursive calls
           0  db block gets
          38  consistent gets
           0  physical reads
           0  redo size
        1218  bytes sent via SQL*Net to client
         430  bytes received via SQL*Net from client
           2  SQL*Net roundtrips to/from client
           4  sorts (memory)
           0  sorts (disk)
          13  rows processed
```

The Execution Plan shows a listing of TABLE ACCESS (FULL), indicating that search performance would be better if you had created the indexes. Based on the select statement, add appropriate indexes. In this example add them to both **num_recycles** and the status in **suspended_usage_t**. This sample statement creates those indexes:

```
SQL> create index i_susp_usage_test on suspended_usage_t (status,
SQL> num_recycles);
```

After creating the indexes, rerunning the select statement results in a more efficient Execution Plan:

```
Execution Plan
----------------------------------------------------------
   0        SELECT STATEMENT Optimizer=CHOOSE
   1    0   NESTED LOOPS
   2    1     TABLE ACCESS (BY INDEX ROWID) OF 'SUSPENDED_USAGE_T'
   3    2       INDEX (RANGE SCAN) OF 'I_SUSP_USAGE_TEST' (NON-UNIQUE)
   4    1     TABLE ACCESS (BY INDEX ROWID) OF 'SUSP_USAGE_TELCO_INFO_T'
   5    4       INDEX (UNIQUE SCAN) OF 'I_SUSP_USAGE_TELCO__ID' (UNIQUE)

Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
          19  consistent gets
          0  physical reads
          0  redo size
          1218  bytes sent via SQL*Net to client
          430  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          13  rows processed
```

The table scan does not read FULL this time, and there are no **recursive calls** and fewer **consistent gets**. The result is a more efficient call.

Configuring and Customizing Suspense Management Center

Follow these procedures to configure and customize the Suspense Management Center application. See Suspense Management Center Help for instructions about using the application.

Setting Up Permissions for Using Suspense Management Center

Before you can use Suspense Management Center, you must first set up its user permissions in Permissioning Center.

- For general information on setting up permissions, see "Setting Up Permissions In BRM Applications" in *BRM System Administrator's Guide*.
- For details on setting up Suspense Management Center permissions, see "Permission Types" in *BRM System Administrator's Guide*.

Adding Custom Fields to Suspense Management Center

Edit **custom.properties** in the *BRM_Home*/**Program Files/Portal Software/SuspenseManagementCenter/lib** directory, adding any custom fields from your **/suspended_usage** subclasses. Suspense Management Center displays the fields defined in the **custom.properties** files.

This example defines a new field called **Record Type:** to display in Suspense Management Center:

```
#1. Specify the display name:
#
#field.<dd_field_name>.name = <display name>

field.PIN_FLD_RECORD_TYPE.name = Record Type:
```

Adding Custom Fields to Suspense Management Center Web Start

Edit the **SuspenseManager_en.jnlp** file (in the *Web_Start_Home* directory), adding any custom fields from your **/suspended_usage** subclasses. The Web version of Suspense Management Center displays the fields that are defined in **SuspenseManager_en.jnlp**.

This example defines a new field called **Record Type:** to display in Suspense Management Center:

```
<resources>
      <j2se version="1.4*"/>
      ...
      ...
      <jar href="lib/Suspense_Management_Help_en.jar"/>
      <extension name="JavaHelp" href="3plibs/jsoft.jnlp"/>
      <property name="suspensemanagement.home" value="f:/apache/apache2/htdocs" />
      <property name="field.PIN_FLD_RECORD_TYPE.name" value="Record Type:" />
      <property name="field.PIN_FLD_RECORD_TYPE.column" value="Record Type" />
</resources>
```

Configuring Event Notification for Suspense Manager

When suspended EDRs are recycled or written off, Suspense Manager uses event notification to call opcodes that perform the appropriate follow-up operations.

Although any subclass of the /event class can be used to trigger event notification (see "About Notification Events" in *BRM Developer's Guide*), Suspense Manager generates the following nonpersistent events specifically to use for event notification:

- **/event/notification/suspense/recycle**: By default, when this event occurs, the EAI framework publishing opcode is called.
- **/event/notification/suspense/writeoff**: By default, when this event occurs, PCM_OP_PROCESS_AUDIT_CREATE_WRITEOFF_SUMMARY is called.
- **/event/notification/suspense/delete**: By default, when this event occurs, *no* opcode is called. To enable this event to trigger an opcode call, see "Editing The Event Notification List" in *BRM Developer's Guide*.
- **/event/notification/suspense/edit**: By default, when this event occurs, *no* opcode is called. To enable this event to trigger an opcode call, see "Editing The Event Notification List" in *BRM Developer's Guide*.

Before you can use Suspense Manager, you must configure the event notification feature as follows:

1. If your system has multiple configuration files for event notification, merge them. See "Merging Event Notification Lists" in *BRM Developer's Guide*.
2. Ensure that the merged file includes entries for these events:
   - (For Revenue Assurance Manager only) From *BRM_Home*/**sys/data/config/pin_notify_ra**:

     /event/notification/suspense/writeoff
   - From *BRM_Home*/**sys/data/config/pin_notify_ifw_sync**:

     /event/notification/suspense/recycle
3. (Optional) Add entries for these events to your final event notification list:
   - **/event/notification/suspense/edit**
   - /event/notification/suspense/delete

   Note: These events are *not* in a default event notification configuration file. You must manually add them to your final event notification list. See "Editing The Event Notification List" in *BRM Developer's Guide*.
4. (Optional) If necessary to accommodate your business needs, add, modify, or delete entries in your final event notification list. See "Editing The Event Notification List" in *BRM Developer's Guide*.
5. (Optional) If necessary to accommodate your business needs, create custom code for event notification to trigger. See "Triggering Custom Operations" in *BRM Developer's Guide*.
6. Load your final event notification list into the BRM database. See "Loading The Event Notification List" in *BRM Developer's Guide*.

For more information, see "Using Event Notification" in *BRM Developer's Guide*.

Configuring Debugging (Optional)

This section explains how to set up the Suspense Management Center debugging information logs and the Suspense Management Center console. This task is optional.

About Logging Debugging Information

Suspense Management Center provides the following ways for capturing and displaying debugging information:

- The **SuspenseManagementCenter_opcodes.log** log file captures all opcode input and output flists used by Suspense Management Center.
- The **javapcm.log** file contains detailed debugging information. By default, the logging level is set to **0**, the lowest level. The highest level is **3**. You must set the error buffer to **true** to enable **javapcm** logging.
- The **Dloglevel** entry creates a console window for the Suspense Management Center that displays error messages and debugging information.

Setting Up Logging of Debugging Information

1. Set up Java PCM Logging by adding these entries to the **Infranet.properties** file (in the *BRM_Home*/**Program Files/Portal Software/SuspenseManagementCenter/lib** directory):

   ```
   infranet.log.level=3
   infranet.log.logallebuf=true
   infranet.log.opcodes.enabled=true
   infranet.log.opcodes.file=SuspenseManagementCenter_opcodes.log
   ```

2. Set up Suspense Management Center console logging by opening the **RunSM.bat** file (in the *BRM_Home***Program Files\Portal Software\SuspenseManagementCenter\lib** directory), and changing the **javaw** entry to **java,** and add this parameter:

   ```
   java -Dloglevel="ALL".
   ```

Logging information is now:

- Displayed in the Suspense Management Center console window.
- Available in these files:
  - *BRM_Home***Program Files\Portal Software\SuspenseManagementCenter\lib\javapcm.log**
  - *BRM_Home***Program Files\Portal Software\SuspenseManagementCenter\lib\SuspenseManagementCenter_opcodes.log**

Configuring the Number of Suspended Records to Process in a Transaction

To configure the number of suspended records you want the suspense management operations to process in a transaction, perform the following tasks:

1. Edit the **pin_suspense_params** file in the *BRM_Home*/**sys/data/config** directory to specify the maximum number of records to process in a transaction. The file includes examples and instructions.

2. Load the contents of the file into the **/config/suspense_params** object in the BRM database by using **load_pin_suspense_params**.

   See "load_pin_suspense_params".

Suspense Management Center and the **pin_recycle** utility read the **/config/suspense_params** file to get the number of records to process in each opcode call and determine the number of times to call the opcodes. For more information, see "Processing Suspended Records in Multiple Steps".

What's Next

Your Suspense Management Center is now ready to accept call records.

23

Using Suspense Manager

This chapter provides general information and advice on how to use Oracle Communications Billing and Revenue Management (BRM) Suspense Manager to process your suspended call records.

- For an overview of Suspense Manager and it's capabilities, see "About Suspense Manager".
- For instructions on how to set up and configure Suspense Manager, see "Configuring Suspense Manager".
- For instructions about using Suspense Management Center, see Suspense Management Center Help.

Processing a Large Number of Suspended Records

You can define search criteria to edit, delete, recycle, and write off a large number of suspended records. You define the search criteria for a specific action, such as edit, recycle, write off, or delete, in Suspense Management Center. Suspense Manager opcodes then perform the specified action on all the records that meet the search criteria and are in a valid state for the action.

For instructions on performing actions on a large set of data, see Suspense Management Center Help.

To avoid a large database transaction during bulk operations, you can specify the number of records to process in each transaction in a bulk operation. Based on the number you specify, Suspense Management Center and the **pin_recycle** utility perform several transactions to process the records in the search result. See "Configuring the Number of Suspended Records to Process in a Transaction".

CSRs who perform operations on large numbers of records require specific permissions that allow these operations. For a list of permission types for Suspense Management Center, see "Suspense Management Center Permission Types" in *BRM System Administrator's Guide*.

Overriding Pipeline Suspense Handling Rules

During recycling, Suspense Management Center lets you process call records and batched call records that do not pass your pipeline validation rules.

This override feature allows you to capture and temporarily hold suspicious calls in a suspended state until you can inspect them. If they pass inspection, you can override your validation rules and recycle the calls to capture the revenue they represent. The

reasons for suspended CDR file are separate from those or individual CDRs and must be handled separately.

You select the override reasons from the Suspense Management Center Recycle screen. The suspense reasons are then overridden for all of the calls in that recycle CDR files. This directs Suspense Manager to successfully process the individual CDRs or CDR files, even though they do not pass your pipeline validation rules.

Changing the List of Override Reasons

The list of override reasons offered to Customer Service Representatives (CSRs) during recycling is configurable. You can change the list at any time by editing the *BRM_Home*/**sys/data/config/pin_suspense_override_reason** file, and then loading it into your database by using the **load_pin_suspense_override_reason** utility in the *BRM_Home*/**bin** directory. *BRM_Home* is the directory where you installed BRM components.

> **Important:** The **load_pin_suspense_override_reason** utility requires a configuration file to function correctly. For details, see "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

For example:

```
%load_pin_suspense_override_reason pin_suspense_override_reason
```

For details on this utility, see "load_pin_suspense_override_reason".

Changing the List of CDR File Override Reasons

The list of CDR file override reasons offered to Customer Service Representatives (CSRs) while resubmitting them is configurable and is separate from the override reasons for individual call records. You can change the list at any time by editing the *BRM_Home*/**sys/data/config/pin_batch_suspense_override_reason** file, and then loading it into your database by using the **load_pin_batch_suspense_override_reason** utility in the *BRM_Home*/**bin** directory.

> **Important:** The **load_pin_batch_suspense_override_reason** utility requires a configuration file to function correctly. For details, see "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

For example:

```
%load_pin_batch_suspense_override_reason pin_batch_suspense_override_reason
```

Using Suspense Management Center with Standard Recycling Call Records

If you upgraded from standard recycling to Suspense Manager, you will have two types of call records in your database. Call records created using standard recycling use the default **/suspended_usage** fields, and have a suspense reason of **Other**. Records created under Suspense Manager will have a **type** that corresponds to your custom subclasses of **/suspended_usage**, and have the suspense reasons you created during the Suspense Manager installation and configuration process.

The records you created using standard recycling can be recycled, written off, and deleted using Suspense Management Center. To search for *all* records created under standard recycling, search for call records with a:

- **type** of **/suspended_usage**
- Suspense reason of **Other**.

To limit the search further, enter the values for any of the **/suspended_usage** fields used by standard recycling.

Standard Recycling does not produce CDR file suspense records.

Troubleshooting Suspense Manager

This section contains fixes to common Suspense Manager problems.

Increasing Heap Size to Avoid Performance Problems

If the searches you run in Suspense Management Center return particularly large results, your performance may slow noticeably, or you may get "Out of memory" error messages. The solution is to increase your maximum heap size. The exact amount varies greatly with your needs and system resources. If performance is very bad or you get "Out of memory" messages frequently, start by doubling the maximum heap size to 128 MB. Remember, however, that making the heap size too large will degrade the performance of other processes.

There are two ways to increase the maximum heap size, depending on whether you have standalone or WebStart BRM implementations.

Increasing Heap Size for Standalone Implementations

1. Edit the *BRM_Home_dir***/lib/runSMC.bat** file to increase the heap size (memory allocation pool) to solve "Out of memory" messages.

   By default, Suspense Management Center has a maximum heap size of 64 MB. This variable is controlled by the **-Xmx** *size* entry in the Suspense Manager Center startup line in **runSMC.bat**. No **-Xmx** *size* entry is present in the startup line by default. To increase the heap size, add this entry and a number (in megabytes) to the Suspense Management Center startup line.

   This example adds a 128 MB maximum heap size to Suspense Management Center:

   ```
   @start C:\PROGRA~1\COMMON~1\PORTAL~1\JRE\bin\javaw.exe -Xmx128m -cp
   ".;%SMCDIR%;%SMCDIR%\lib;%SMCDIR%\lib\suspensemgtmain.jar;%SMCDIR%\lib\pfc.jar;
   %SMCDIR%\3plibs\jh.jar;%SMCDIR%\lib\pcmext.jar;%SMCDIR%\lib\pcm.jar;%SMCDIR%\li
   b\Suspense_Management_Help_en.jar;%SMCDIR%\lib\Application_Center_Help_en.jar;"
   com.portal.appcenter.AppCenterMain suspensemgtsuite
   ```

   Important: Be sure to precede and follow the **-Xmx** *size* entry with a space.

2. Stop and restart Suspense Management Center to make the change take effect.

Increasing Heap Size for Web Start Implementations

1. Open your **SuspenseManagement**_*locale*.**jnlp** file.

2. Change the **j2se** element to include a max-heap-size attribute.

   The default entry looks like this:

   ```
   <j2se version="1.4*"/>
   ```

   For example, this entry changes the maximum heap size to 128 megabytes:

   ```
   <j2se version="1.4*" max-heap-size="128m"/>
   ```

   Note: The max heap size specified in the JNLP file is used for all associated Suspense Management Center clients.

3. Stop and restart Suspense Management Center to make the change take effect.

Unexpected Log Messaged Caused by Missing MaxErrorRates Entry

Your pipeline output section must contain a MaxErrorRates module containing at least one entry. If this entry is missing, your log files will contain a misleading message like this one:

```
"16.11.2004 21:00:37 All checks are successful. File can be recycled."
```

Suspense Manager Performance

The more call records you attempt to edit, recycle, delete, archive, or archive then delete, the longer it takes. It is impossible to say exactly how long because every implementation is different, but 30,000 records will take a few minutes, and recycling 300,000 records will take many minutes.

24

Suspense Reasons

Table 24–1 shows the default mapping between various Oracle Communications Billing and Revenue Management (BRM) error messages and Suspense Manager reasons for suspending event data records (EDRs). The information in this table is derived from several different source files and is much easier to understand in this format.

You may want to change this mapping or add your own error messages as appropriate for your BRM implementation. For information on how to add to or change these messages, see "Changing the List of Suspense Reasons and Subreasons."

Table 24–1 Suspense Reasons

BRM Error Messages	Suspense Reason String	Suspense Subreason String
Pipeline Framework Errors	NA	NA
ERR_INPUT_MAPPING_FAILED	Batch rating engine processing error	Record mapping error
ERR_TRANSFER_CUTOFF_VIOLATED	Batch rating engine processing error	Record mapping error
ERR_INCORRECT_FILLER_LENGTH	Batch rating engine processing error	Record mapping error
ERR_CANNOT_JOIN_EVENT_ HANDLER_PROC	Batch rating engine processing error	Record mapping error
ERR_INVALID_RECORD_NUMBER	Batch rating engine processing error	Record mapping error
ERR_INVALID_FIRST_CALL_TIMESTAMP	Batch rating engine processing error	Record mapping error
ERR_INVALID_LAST_CALL_TIMESTAMP	Batch rating engine processing error	Record mapping error
ERR_A_CUSTOMER_NOT_FOUND	Customer data error	Customer error
ERR_NO_SPLITTING_PERFORMED	Splitting error	Splitting error
ERR_ADD_DATABLOCK	Batch rating engine processing error	Batch rating engine processing error
ERR_DATABASE	Batch rating engine processing error	Database error
ERR_RATEPLAN_NOT_FOUND	Rating error	Invalid rate plan
ERR_INSUFFICIENT_MEMORY	Batch rating engine processing error	Insufficient memory
DAT_Account Errors	NA	NA

Table 24–1 (Cont.) Suspense Reasons

BRM Error Messages	Suspense Reason String	Suspense Subreason String
ERR_BALANCE_GROUP_UPDATE	Customer data error	Internal error
ERR_BILL_INFO_UPDATE	Customer data error	Internal error
ERR_MAPPING_TABLE_UPDATE	Customer data error	Internal error
ERR_CUSTOMER_LOGIN_NOT_FOUND	Customer data error	Customer error
ERR_CUSTOMER_LOGIN_SERVICE_ NOT_FOUND	Customer data error	Customer error
ERR_CUSTOMER_LOGIN_ACCOUNT_ NOT_FOUND	Customer data error	Customer error
ERR_SERVICE_NOT_CONFIGURED	Customer data error	Customer error
ERR_CUSTOMER_SERVICE_NOT_ FOUND	Customer data error	Customer error
ERR_CUSTOMER_EDR_PARSING	Customer data error	Customer error
ERR_CUSTOMER_LOGIN_NOT_VALID_ FOR_TIME	Customer data error	Customer error
ERR_CUSTOMER_NO_VALID_PRODUCT	Customer data error	Customer error
ERR_CUSTOMER_NO_VALID_ PRODUCT_RATING	Customer data error	Customer error
ERR_CUSTOMER_INVALID_ITEM_POID	Customer data error	Customer error
ERR_INVALID_OUTPUT_STREAM	Customer data error	Customer error
ERR_CUSTOMER_LOGIN_INTERNAL_ ERROR	Customer data error	Internal error
ERR_ACCRT_MESSAGE	Customer data error	Internal error
ERR_NOT_IMPLEMENTED	Customer data error	Internal error
FCT_AggreGate Errors	NA	NA
ERR_NO_DEPENDENT_CLASS_DEFINED	Record aggregation error	Aggregation error
ERR_EDR_ITERATOR_FAILURE	Record aggregation error	Aggregation error
FCT_BalancePlugIn Errors	NA	NA
ERR_BALANCE_NOT_FOUND	Billing record error	Failed to add discount balance info
FCT_CallAssembling Errors	NA	NA
ERR_CHAIN_REFERENCE_MISSING	Call assembling error	Call assembling error
ERR_INVALID_STATE_INDICATOR	Call assembling error	Call assembling error
ERR_REJECTED_EDR_NOT_IN_ WORKFILE	Call assembling error	Call assembling error
ERR_EDR_ALREADY_CLOSED	Call assembling error	Call assembling error
FCT_CustomerRating Errors	NA	NA
ERR_CUSTOMER_NOT_FOUND	Rating error	Customer rating error
ERR_RATEPLAN_NOT_DEFINED	Rating error	Customer rating error
FCT_Discount Errors	NA	NA

Table 24–1 (Cont.) Suspense Reasons

BRM Error Messages	Suspense Reason String	Suspense Subreason String
ERR_INVALID_GRANT_TYPE	Discount processing error	Discount processing error
ERR_PLUGIN_INVALID_STATE	Discount processing error	Discount processing error
ERR_EDR_ITERATOR	Discount processing error	Discount processing error
ERR_EDRPACK_NOT_READY_DSC	Discount processing error	Discount processing error
ERR_BEGIN_EDR	Discount processing error	Discount processing error
ERR_COMMIT_EDR	Discount processing error	Discount processing error
ERR_CANCEL_EDR	Discount processing error	Discount processing error
ERR_ROLLBACK_EDR	Discount processing error	Discount processing error
ERR_CANCEL_DEMANDED_EDR	Discount processing error	Discount processing error
ERR_BEGIN_DSC_TRANSACTION	Discount processing error	Discount processing error
ERR_END_DSC_TRANSACTION	Discount processing error	Discount processing error
ERR_DSC_CONF_NOT_FOUND	Discount processing error	Discount processing error
ERR_INVALID_THRESHOLD_TYPE	Discount processing error	Discount processing error
ERR_INVALID_DISCOUNT_TYPE	Discount processing error	Discount processing error
ERR_DISCOUNT_DETACH_MODULE	Discount processing error	Discount processing error
ERR_ACCOUNT_COMMIT_RESTART	Discount processing error	Discount processing error
ERR_ACCOUNT_PREPARECOMMIT	Discount processing error	Discount processing error
ERR_ACCOUNT_COMMIT	Discount processing error	Discount processing error
ERR_ACCOUNT_CANCEL	Discount processing error	Discount processing error
ERR_ACCOUNT_ROLLBACK	Discount processing error	Discount processing error
ERR_NO_ACCOUNT_BALANCE_ERROR	Discount processing error	Discount processing error
ERR_CANNOT_COMPILE_SCRIPT	Discount processing error	Discount processing error
ERR_CURRENCY_RESID_NOT_FOUND	Discount processing error	Discount processing error
ERR_INVALID_BASE_AMOUNT	Discount processing error	Discount processing error
ERR_INVALID_BASE_EXPR	Discount processing error	Discount processing error
ERR_INVALID_COND_AMOUNT	Discount processing error	Discount processing error
WRN_INVALID_DRUM_AMOUNT	Discount processing error	Discount processing error
ERR_INVALID_THRESHOLD_AMOUNT	Discount processing error	Discount processing error
ERR_REJECT_EDR	Discount processing error	Discount processing error
ERR_EXPR_REF_CP	Discount processing error	Discount processing error
ERR_GETTING_BG_ID	Discount processing error	Discount processing error
FCT_EnhancedSplitting Errors	NA	NA
ERR_NO_SPLITTING_ENTRY	Splitting error	Splitting error
FCT_ExchangeRate Errors	NA	NA
ERR_EXCHANGERATE_BRK_ HEADERDATE	Rating error	Exchange rate error
ERR_EXCHANGERATE_FILEDATE_NOT_ EXIST	Rating error	Exchange rate error

Table 24–1 (Cont.) Suspense Reasons

BRM Error Messages	Suspense Reason String	Suspense Subreason String
ERR_EXCHANGERATE_NOT_FOUND	Rating error	Exchange rate error
FCT_MainRating Errors	NA	NA
ERR_RATEPLAN_VERSION_ID_NOT_FOUND	Rating error	Invalid rate plan
ERR_RATEPLAN_VERSION_DATE_NOT_FOUND	Rating error	Invalid rate plan
ERR_TIMEMODEL_NOT_FOUND	Rating error	Invalid time model or time zone
ERR_RATE_PRICEMODEL_NOT_FOUND	Rating error	Invalid price model
ERR_TIMEZONE_NOT_FOUND	Rating error	Invalid time model or time zone
ERR_PRICEMODEL_NOT_FOUND	Rating error	Invalid price model
ERR_PRICEMODEL_RUM_NOT_FOUND	Rating error	Invalid price model
ERR_PRICEMODEL_STEP_NOT_FOUND	Rating error	Invalid price model
ERR_PRICEMODEL_CONFIG_NOT_FOUND	Rating error	Invalid price model
ERR_INVALID_ADDON_TYPE	Rating error	Other main rating error
ERR_RUM_GROUP_NOT_FOUND	Rating error	Other main rating error
FCT_PreRating Errors	NA	NA
ERR_RATEPLAN_NOT_A_NUMBER	Rating error	Invalid rate plan
ERR_RATEPLAN_TYPE_INV	Rating error	Invalid rate plan
ERR_RATEPLAN_VERSION_NOT_FOUND	Rating error	Invalid rate plan
ERR_NO_RATEPLAN	Rating error	Invalid rate plan
FCT_Tadig2PlmnPlugIn Errors	NA	NA
ERR_TADIG_NOT_FOUND	Mapping problem	TADIG to PLMN map error
Ciber Errors	NA	NA
ERR_CIBER_RET	Roaming records error	CIBER record error
Tap Errors	NA	NA
ERR_TAP3_RET	Roaming records error	TAP record error
FCT_AccountRouter Errors	NA	NA
ERR_JOB_AMT_MIGRATION	Internal suspension	Account being migrated
FCT_TriggerBill Errors	NA	NA
ERR_TRIGGER_BILLING	Internal suspension	Awaiting billing of account
FCT_Account Errors	NA	NA
ERR_JOB_RERATING_ACCOUNT	Internal Suspension	Account usage being re-rated
FCT_Filter_Set Errors	NA	NA
ERR_INVALID_DISCOUNT_ID	Filter set error	Invalid Discount ID
ERR_INVALID_PRODUCT_ID	Filter set error	Invalid Product ID

Table 24–1 (Cont.) Suspense Reasons

BRM Error Messages	Suspense Reason String	Suspense Subreason String
Network Mediation Errors	NA	NA
CANCEL_NOT_ALLOWED	Network mediation ECE DC cartridge error	Request processing cannot be cancelled.
CREDIT_CEILING_BREACH	Network mediation ECE DC cartridge error	Request processing resulted in ceiling floor breached of an involved BalanceItem.
CREDIT_FLOOR_BREACH	Network mediation ECE DC cartridge error	Request processing resulted in credit floor breached of an involved BalanceItem.
CUSTOM_EXTENSION_ERROR	Network mediation ECE DC cartridge error	Error has occurred while executing operator specific custom extension code.
CUSTOMER_IN_TRANSACTION	Network mediation ECE DC cartridge error	Customer is in another transaction
DUPLICATE_REQUEST	Network mediation ECE DC cartridge error	It is a duplicate request
FINAL_UNIT_INDICATOR	Network mediation ECE DC cartridge error	Request processing raised a final unit indicator flag.
INCORRECT_IMPACTS_FOR_DEBIT	Network mediation ECE DC cartridge error	The no. of debit impacts does not match with the actual debit impacts.
INSUFFICIENT_RATED_QUANTITY	Network mediation ECE DC cartridge error	Rated quantity was less than the minimum quantity for reservation.
LIFECYCLE_VALIDATION_FAILED	Network mediation ECE DC cartridge error	After life cycle validation call was disallowed.
MISSING_SYSTEM_ALTERATION_FOR_DEBIT	Network mediation ECE DC cartridge error	There are no system alteration offers for the debit request.
NO_QUALIFIED_CHARGE_OFFERS	Network mediation ECE DC cartridge error	There are no qualified charge offers for this request.
NO_RATED_QUANTITY	Network mediation ECE DC cartridge error	No quantity rated.
REFUND_NOT_ALLOWED	Network mediation ECE DC cartridge error	Refund request is not allowed.
SYSTEM_ERR	Network mediation ECE DC cartridge error	Undefined system error.
TX_FAILED	Network mediation ECE DC cartridge error	Transaction commit failed.
ZERO_RUM_QUANTITY	Network mediation ECE DC cartridge error	The RUM is zero.

25

About Suspense Manager Opcodes

This chapter describes the Oracle Communications Billing and Revenue Management (BRM) Suspense Manager standard opcodes. These opcodes manage suspended EDRs stored in the BRM database as **/suspended_usage** objects. These opcodes also manage the Batch Suspense Record of suspended CDR file, stored in the BRM database as **/suspended_batch** objects.

For information about Suspense Manager, see "About Suspense Manager".

Recycling Suspended EDRs

PCM_OP_SUSPENSE_SEARCH_RECYCLE searches for and queues suspended call records for recycling. There are many search criteria that you may use to search the records, such as by CDR file or by recycle key.

This opcode is usually called by the **pin_recycle** utility, which searches for and deletes call records with a specific recycle key and a status of **Succeeded** (or **Written-off**).

Searching for EDRs in a CDR File

The BRM standard recycling feature collects and manipulates all calls contained in a CDR file at the same time. PCM_OP_SUSPENSE_SEARCH_RECYCLE searches for a specified CDR file and queues its suspended EDRs for recycling. If successful, the EDRs are assigned a status of **Succeeded**.

Searching for EDRs with a Recycle Key

PCM_OP_SUSPENSE_SEARCH_RECYCLE is used by features that need to temporarily delay rating of EDRs. These features include pipeline modules to add a recycle key and error to those EDRs during pipeline rating. This opcode searches for all **Suspended** call records containing the recycle key passed in on the input flist. It then queues those call records to be rated at the next opportunity. If successful, the EDRs are assigned a status of **Succeeded**.

Initiating Suspense Recycling

PCM_OP_SUSPENSE_RECYCLE_USAGE initiates EDR recycling. During recycling, suspended EDRs are sent back through their original rating pipelines. The Suspense Management Center calls this opcode when the user chooses to recycle suspended EDRs.

PCM_OP_SUSPENSE_RECYCLE_USAGE can operate in test mode. In test mode, the EDR is sent through the rating pipeline normally. The rating results, however, are not output from the pipeline if the EDR is processed successfully. In test mode, the

pipeline does not make any state changes, such as updating aggregation counters in discounting.

PCM_OP_SUSPENSE_RECYCLE_USAGE takes as input an array of **/suspended_usage** object POIDs, a list of suspense override reasons, and a value that indicates whether the EDRs should be recycled in test mode. This opcode then creates an **/admin_action/suspended_usage/recycle** object with that information.

For each **/suspended_usage** object specified in the input flist, PCM_OP_SUSPENSE_RECYCLE_USAGE performs these operations:

- Confirms that the suspended EDR has a status of **Suspended**. The PIN_FLD_STATUS value in the object must be **0**.
- Changes the EDR status to **Recycling**. The value of the PIN_FLD_STATUS field in the **/suspended_usage** object is set to **1**.
- Creates an **/admin_action/suspended_usage/recycle** object containing the recycle mode and override reasons (passed in the input flist).
- Adds the POID of the newly created **/admin_action/suspended_usage/recycle** object to the actions array of each **/suspended_usage** object.
- Sets the PIN_FLD_RECYCLE_OBJ field of each **/suspended_usage** object to the **/admin_action/suspended_usage/recycle** object POID. Pipeline Manager uses this field to find all EDRs associated with a recycle request.
- Creates a notification event that triggers the Account Synchronization Data Manager (**dm_ifw_sync**) to send a message to the DAT_Listener module. This initiates the process of recycling the EDR through the rating pipeline.

After EDRs are recycled, their **/suspended_usage** objects are updated by using the Suspended Event (SE) Loader:

- If an EDR is successfully recycled, the status is changed to **Succeeded**, and the PIN_FLD_STATUS value is changed to **2**.
- If an EDR is not successfully recycled, the status is changed back to **Suspended**, and the PIN_FLD_STATUS value is changed to **0**. In addition, the following fields are updated because their original values could have changed:
  - The suspense reason code, PIN_FLD_SUSPENSE_REASON.
  - The suspense subreason code, PIN_FLD_SUSPENSE_SUBREASON.
  - The Pipeline Manager error code, PIN_FLD_PIPELINE_ERROR_CODE.
- If PCM_OP_SUSPENSE_RECYCLE_USAGE was run in test mode, the status is changed back to **Suspended**, and the PIN_FLD_STATUS value is changed to **0**. The suspense reason Pipeline Manager error code are not updated, but the corresponding test values (PIN_FLD_TEST_SUSPENSE_REASON, PIN_FLD_TEST_SUSPENSE_SUBREASON, and PIN_FLD_TEST_SUSPENSE_ERROR_CODE) are updated to include information about the results of the test recycling.
- The number of times the record has been recycled (in PIN_FLD_NUM_RECYCLES) is increased by 1.

PCM_OP_SUSPENSE_RECYCLE_USAGE returns the routing POID specified in the input flist.

Resubmitting Suspended Batches

PCM_OP_BATCH_SUSPENSE_RESUBMIT_BATCHES resubmits suspended CDR file. When resubmitted, suspended CDR file are sent back through their original pipeline.

Suspense Management Center calls this opcode when the user chooses to resubmit suspended batches.

PCM_OP_BATCH_SUSPENSE_RESUBMIT_BATCHES takes an array of **/suspended_batch** object POIDs and a list of suspense override reasons as input. This opcode then creates an **/admin_action/suspended_batch/resubmit** object with that information.

For the whole set of **/suspended_batch** objects specified in the input flist, PCM_OP_BATCH_SUSPENSE_RESUBMIT_BATCHES performs these operations:

- Creates a transaction if it is not already opened.
- Creates an ADMIN_ACTION object, **/admin_action/suspended_batch/resubmit**, with the override reason, for each **/suspended_batch** object.
- Validates the status of each **/suspended_batch** object (Batch Suspense Record) and updates the status with the result of the resubmission.
- Creates an event Flist (with **/event/notification/suspense/batch_resubmit**) and calls PCM_OP_ACT_USAGE in CALC_ONLY mode.
- Closes the transaction

After CDR file are resubmitted, their **/suspended_batch** objects (Batch Suspense Records) are updated by using the SE Loader:

- The PIN_FLD_NUM_RESUBMISSIONS field in each **/suspended_batch** object is incriminated.
- If a batch is successfully resubmitted, the status is changed to **Succeeded**, and the PIN_FLD_STATUS value is changed to **2**.
- If a batch is not recycled successfully, the status is changed back to **Suspended**, and the PIN_FLD_STATUS value is changed to **0**. This will cause all the batches in the resubmission task to be rolled back as well. In addition, these fields are updated because their original values could have changed:
  - The suspense reason code, PIN_FLD_BATCH_SUSPENSE_REASON.
  - The suspense sub-reason code, PIN_FLD_BATCH_SUSPENSE_SUBREASON.
  - The Pipeline Manager error code, PIN_FLD_PIPELINE_ERROR_CODE.

PCM_OP_BATCH_SUSPENSE_RESUBMIT_BATCHES returns the routing POID specified in the input flist.

Changing the Contents of Fields in Suspended EDRs

Use PCM_OP_SUSPENSE_EDIT_USAGE to change the contents of fields in suspended EDRs. Suspense Management Center calls this opcode to edit a suspended call record.

Important: This opcode is available to Suspense Manager customers only.

Note: You cannot edit the CDR or EDR fields of records in a CDR file that has been suspended by batch suspense and has a Batch Suspense Record.

PCM_OP_SUSPENSE_EDIT_USAGE performs these operations:

- Searches for the **/suspended_usage_edits** object that correspond to the appropriate brand account (determined by the login session).
  - If a **/suspended_usage_edits** object exists for the appropriate brand account, this opcode reads and locks it.
  - If this opcode does not find a **/suspended_usage_edits** object does not exist for the appropriate brand account, this opcode creates it.
- Takes as input an array of **/suspended_usage** object POIDs and an array of the EDR fields to be edited, including the old and new values.
- For each EDR field to be edited, this opcode:
  - Creates an **/admin_action/suspended_usage/edit** object, which includes both the old and new values.
  - Adds the **/admin_action/suspended_usage/edit** object POID to the top of the **/suspended_usage_edits** object stack. If the stack is full, it removes the oldest POID.
- For each **/suspended_usage** object, this opcode:
  - Confirms that the suspended EDR has a status of **Suspended**. The PIN_FLD_STATUS value in the object must be **0**.
  - Adds the **/admin_action/suspended_usage/edit** object POID and the old value to each **/suspended_usage** object's action array.
  - Updates the modified flag (PIN_FLD_EDITED), which indicates that the field has been edited.

PCM_OP_SUSPENSE_EDIT_USAGE returns an array of the POID of the **/admin_action/suspended_usage/edit** objects it creates.

Undoing Edits to Suspended EDRs

Use the PCM_OP_SUSPENSE_UNDO_EDIT_USAGE opcode to undo edits to suspended EDRs.

This opcode is called by Suspense Management Center to the undo edits. It replaces the value of a field in a suspended call record with the value in that field before the last edit was made.

> **Important:** This opcode is available to Suspense Manager customers only.

PCM_OP_SUSPENSE_UNDO_EDIT_USAGE performs these operations:

- Searches for the **/suspended_usage_edits** object that corresponds to the appropriate brand account (determined by the login session). If a **/suspended_usage_edits** object exists for that brand account, this opcode reads and locks it.
- Confirms that the POID of the **/admin_action/suspended_usage/edit** object on the input flist matches that of the top POID of the **/suspended_usage_edits** object stack. If the two don't match, returns failure status (PIN_FLD_RESULT is set to **1**) and returns the POID at the top of the **/suspended_usage_edits** object stack.
- Confirms that each suspended call record affected by the edit has a status of **Suspended**.

- Undoes the edit by replacing the existing field values with the values before the last edit was saved (the values referenced by the POID at the top of the **/suspended_usage_edits** object stack).
- Adds the session, service, and date and time details of the undo edit operation in the PIN_FLD_UNDO_DATA array of the **/admin_action/suspended_usage/edit** object.
- Removes the POID of the **/admin_action/suspended_usage/edit** object from the top of the **/suspended_usage_edits** object stack.

PCM_OP_SUSPENSE_UNDO_EDIT_USAGE returns a **PIN_FLD_RESULT** value of:

- **0** if the undo edit was successful. Also:
  - The PIN_FLD_POID field contains the POID of the **/admin_action/suspended_usage/edit** object.
  - The PIN_FLD_COUNT field contains the number of records that were processed.
- **1** if the **/admin_action/suspended_usage/edit** POID in the input flist does not match the POID at the top of the **/suspended_usage_edits** object stack.

Deleting Records for Suspended EDRs

Use PCM_OP_SUSPENSE_DELETE_USAGE to delete records for suspended EDRs.

Important: This opcode is available to Suspense Manager customers only.

The Suspense Management Center calls this opcode to delete EDRs. EDRs can be deleted only if their status is **Written-off** or **Succeeded**.

PCM_OP_SUSPENSE_DELETE_USAGE takes as input an array of **/suspended_usage** object POIDs to be deleted.

For each **/suspended_usage** object specified in the input flist, PCM_OP_SUSPENSE_DELETE_USAGE performs these operations:

- Confirms that the suspended EDR has a status of **Succeeded** or **Written off.** The value of the PIN_FLD_STATUS field in the **/suspended_usage** object must be **2** or **3**.
- Deletes the **/suspended_usage** object.
- Generates a **/event/notification/suspense/delete** object that records the deletion.

PCM_OP_SUSPENSE_DELETE_USAGE returns the routing POID specified in the input flist.

Deleting Records for Suspended Batches

Use PCM_OP_BATCH_SUSPENSE_DELETE_BATCHES to delete **/suspended_batch** objects (Batch Suspense Records).

Note: This opcode deletes records, not the files associated with them.

Suspense Management Center calls this opcode to delete Batch Suspense Records (**/suspended_batch** objects). Batch Suspense Records can be deleted only if their status is **Written-off** or **Succeeded**.

PCM_OP_BATCH_SUSPENSE_DELETE_BATCHES takes as input an array of **/suspended_batch** object POIDs to be deleted.

For each **/suspended_batch** object specified in the input flist, PCM_OP_BATCH_SUSPENSE_DELETE_BATCHES performs these operations:

- Creates a transaction if it is not already opened.
- Confirms that the suspended batch file has a status of **Succeeded** or **Written-off**. The value of the PIN_FLD_STATUS field in the **/suspended_batch** object must be **2** or **3**. Otherwise, an error code is generated and the transaction ends.
- Deletes the **/suspended_batch** object.
- Generates an **/event/notification/suspense/batch_delete** event.
- Closes the transaction.

PCM_OP_BATCH_SUSPENSE_DELETE_BATCHES returns the routing POID specified in the input flist.

Deleting Call Records with a Specific Recycle Key and a Status of Succeeded or Written-Off

Use PCM_OP_SUSPENSE_SEARCH_DELETE to delete call records with a specific recycle key and a status of **Succeeded** or **Written-off**.

Set the **PIN_FLD_FLAGS** field to either of these values:

- **0**: Directs this opcode to delete EDRs with a status of:
  - **Succeeded** (successfully processed)
  - Written-off
- **1**: Directs this opcode to delete EDRs with a status of:
  - **Succeeded** (successfully processed)
  - Written-off
  - **Suspended**. This opcode first writes off, then deletes, these EDRs

Deleting EDRs in a CDR File

PCM_OP_SUSPENSE_SEARCH_DELETE is used with the BRM standard recycling feature, which acts on all the calls contained in a single CDR file simultaneously. Using **pin_recycle** with the **-d** parameter deletes all calls in a CDR file that have a status of **Succeeded**. Using **pin_recycle** with the **-D** parameter deletes all calls in a CDR file with a status of **Succeeded** or **Written off**.

Deleting Calls with a Recycle Key

PCM_OP_SUSPENSE_SEARCH_DELETE is also used by features that need to temporarily delay rating of EDRs. These features include pipeline modules to add a recycle key and error to those EDRs during pipeline rating. The error prevents EDRs from being rated by assigning them a status of **Suspended**. Those features then use PCM_OP_SUSPENSE_EDIT_USAGE to find and recycle the call records. If successful,

the EDRs are assigned a status of **Succeeded**. This opcode then deletes those successfully recycled call records.

Writing Off Suspended EDRs

Use PCM_OP_SUSPENSE_WRITTEN_OFF_USAGE to write off suspended EDRs.

When a suspended EDR is written off, it can no longer be edited or recycled.

> **Important:** This opcode is available to Suspense Manager customers only.

PCM_OP_SUSPENSE_WRITTEN_OFF_USAGE takes as input an array of **/suspended_usage** object POIDs.

- PCM_OP_SUSPENSE_WRITTEN_OFF_USAGE creates an **/admin_action/suspended_usage/writeoff** object that records the write-off.

For each **/suspended_usage** object specified in the input flist, PCM_OP_SUSPENSE_WRITTEN_OFF_USAGE performs these operations:

- Confirms that the suspended EDR has a status of **Suspended**; the PIN_FLD_STATUS value must be **0**.
- Adds the POID of the newly created **/admin_action/suspended_usage/writeoff** object to the array of actions in the **/suspended_usage** object.
- Changes the status to **Written-off**. The value of the PIN_FLD_STATUS field in the **/suspended_usage** object is changed to **3**.
- Generates an **/event/notification/suspense/batch_writeoff** event.

PCM_OP_SUSPENSE_WRITTEN_OFF_USAGE returns the POID of the **/admin_action/suspended_usage/writeoff** object created.

Writing Off Suspended Batches

Use PCM_OP_BATCH_SUSPENSE_WRITE_OFF_BATCHES to write off suspended CDR files.

When you write off a suspended CDR file, you can no longer resubmit it, but you can delete it.

The opcode creates an **/admin_action/suspended_batch/writeoff** object that records the write-off and sets the status of the Batch Suspense Record (**/suspended_batch**) to **Written-off**.

PCM_OP_BATCH_SUSPENSE_WRITE_OFF_BATCHES performs these operations:

- Create a transaction if it is not already opened.
- Confirms that the suspended EDR has a status of **Suspended**; the PIN_FLD_STATUS value must be **0**.
- PCM_OP_BATCH_SUSPENSE_WRITE_OFF_BATCHES takes as input an array of **/suspended_batch** object POIDs.
- Adds the POID of the newly created **/admin_action/suspended_batch/writeoff** object to the array of actions in the **/suspended_batch** object.
- Changes the status of the Batch Suspense Record (**/suspended_batch**) to **Written off**; The value of the PIN_FLD_STATUS field in **/suspended_batch** is changed to **3**.

- Generates an **/event/notification/suspense/batch_writeoff** event.
- Closes the transaction.

PCM_OP_BATCH_SUSPENSE_WRITE_OFF_BATCHES returns the POID of the **/admin_action/suspended_batch/writeoff** object created.

Processing Suspended Records in Bulk

You can use the Suspense Manager opcodes to edit, delete, recycle, and write off a large number of suspended records. For more information, see:

- Processing Suspended Records in Multiple Steps
- Editing Suspended Records in Bulk
- Writing Off Suspended Records in Bulk
- Deleting Suspended Records in Bulk

Processing Suspended Records in Multiple Steps

You can process the suspended records in multiple steps by calling the opcodes multiple times, to avoid a large database transaction:

1. Specify the number of records to process in each opcode call in a configuration file and load the file into the **/config/pin_suspense_system_params** storable class.

   For more information, see "Configuring the Number of Suspended Records to Process in a Transaction".
2. Call the opcode in calc-only mode to retrieve the count and POID range of the records that match your search criteria.
3. Use a simple logic to determine the number of times to call the opcodes depending on the number of records you want each opcode call to process.
4. Call the opcode several times with a set of records each time and consolidate the results returned by each opcode call.

Note: For each opcode call, you must provide the POID range and the corresponding arguments in the input flist.

Because each operation is performed in multiple steps, if the operation is successful in any of the steps, you get the number of records processed. You also get an error message for the unsuccessful records, which you can display to the user. If any one of the steps fails, the entire step and the following steps are canceled and an error message is returned.

Note: Suspense Management Center and the **pin_recycle** utility perform suspense management operations in multiple steps by calling the opcodes multiple times.

Editing Suspended Records in Bulk

Use the PCM_OP_SUSPENSE_SEARCH_EDIT opcode to perform the same set of edits on a large number of suspended records that meet the criteria you specify.

This opcode makes changes to the records in one database operation instead of accessing the database for each record. It calls the following opcodes:

- PCM_OP_BULK_WRITE_FLDS, to update the objects in the database.
- PCM_OP_ACT_USAGE, to generate the edit event notification.

> **Caution:** You cannot undo edits performed on a large number of records or any edits made before the bulk edit operation.

PCM_OP_SUSPENSE_SEARCH_EDIT follows these steps to edit suspended records:

1. Takes as input the following information:
   - The POID type of the suspended usage class.
   - The search criteria template.
   - The fields and values that need to be edited in the object.
   - An array of the EDR fields to be edited, including the old and new values.
2. Does one of the following:
   a. If the PCM_OPFLG_CALC_ONLY flag is set, returns the count and the POID range of records that meet the search criteria.
   b. If the PCM_OPFLG_CALC_ONLY flag is not set, searches for the **/suspended_usage_edits** objects that correspond to the appropriate brand account, which is determined by the login session.
3. Does one of the following:
   a. If it finds **/suspended_usage_edits** objects for the specified brand account, clears all the POIDs of the edit actions stored in the objects.

   > **Note:** After the objects are changed, the current changes or previous changes cannot be undone.

   b. If it does not find a **/suspended_usage_edits** object for the appropriate brand account, creates a new **/suspended_usage_edits** object.
4. For each EDR field to be edited, creates an **/admin_action/suspended_usage/edit** object, which includes both the old and new values.
5. For each **/suspended_usage** object, performs the following operations:
   - Verifies that the suspended EDR has a status of **Suspended**. The PIN_FLD_STATUS value in the object must be **0**.
   - Adds the **/admin_action/suspended_usage/edit** object POID and the old value to each **/suspended_usage** object's action array.
   - Updates the PIN_FLD_EDITED field to indicate that the field has been edited.

If successful, PCM_OP_SUSPENSE_SEARCH_EDIT generates an edit notification event that includes the administrative action POID of the edit action and returns success along with the count of the objects edited. If the operation fails in any record, it cancels the entire operation and returns failure with the appropriate error code, leaving the state of the record as it was before the operation.

Writing Off Suspended Records in Bulk

Use the PCM_OP_SUSPENSE_SEARCH_WRITE_OFF opcode to write off all suspended records that meet the criteria you define.

> **Note:** You cannot edit or recycle suspended records that are written off.

This opcode writes off a large set of suspended records in one database operation instead of accessing the database for each record. It calls the following opcodes:

- PCM_OP_BULK_WRITE_FLDS to mark a large number of objects in the database as written off.
- PCM_OP_ACT_USAGE to generate the write-off event notification.

PCM_OP_SUSPENSE_SEARCH_WRITE_OFF follows these steps to write off suspended records:

1. Takes as input the POID type of the suspended usage class and the search criteria template for the objects to be written off.
2. If the PCM_OPFLG_CALC_ONLY opcode flag is set, returns the count of records that match the search criteria and the POID range of the records that satisfy the criteria.
3. If the PCM_OPFLG_CALC_ONLY flag is *not* set, creates the **/event/notification/suspense/batch_writeoff** object that records the write off and returns that object with the count of records written off.
4. For each **/suspended_usage** object that meets the criteria specified in the template, performs these operations:
   - Verifies that the suspended EDR has a status of **Suspended**. The PIN_FLD_STATUS value in the object must be **0**.
   - Adds the POID of the newly created **/admin_action/suspended_usage/writeoff** object to the array of actions in the **/suspended_usage** object.
   - Changes the status to **Written-off**. The value of the PIN_FLD_STATUS field in the **/suspended_usage** object is changed to **3**.

If successful, PCM_OP_SUSPENSE_SEARCH_WRITE_OFF generates a write-off notification event that includes the administrative action POID of the write-off action and returns success along with the number of records written off. If the operation fails in any record, it cancels the entire operation and returns failure with the appropriate error code, leaving the state of the record as it was before the operation.

Deleting Suspended Records in Bulk

Use the PCM_OP_SUSPENSE_SEARCH_DELETE opcode to delete all suspended records that meet the criteria you define.

> **Note:** You can only delete records that are succeeded or written off.

This opcode deletes a large set of suspended records in one database operation instead of accessing the database for each record. It calls PCM_OP_ACT_USAGE to generate the delete event notification.

PCM_OP_SUSPENSE_SEARCH_DELETE follows these steps to delete suspended records:

1. Takes as input the POID type of the suspended usage class and the search criteria template for the objects to be deleted.
2. If the PCM_OPFLG_CALC_ONLY opcode flag is set, returns the count of records that match the search criteria and the POID range of the records that satisfy the criteria.

   If the PCM_OPFLG_CALC_ONLY flag is *not* set, creates the **/event/notification/suspense/batch_delete** object that records the deletion.
3. For each **/suspended_usage** object that meets the criteria specified in the template, performs these operations:
   - Verifies that the suspended EDR has a status of **Succeeded** or **Written off**. The PIN_FLD_STATUS value in the object must be **2** or **3**.
   - EDRs with a status of **Succeeded** and those with a status of **Written off** that do not have a deferred duration are deleted immediately.
   - If the status of the suspended record is **Written off**, the opcode checks if there is a deferred duration and if the deferred duration is greater than 0. The deferred duration is a parameter defined in the **/config/suspense_params** object.

     Note: The load utility is provided to load the deferred duration parameter.

     If there is a deferred duration, then the PCM_OP_SUSPENSE_SEARCH_DELETE opcode will not remove the suspended record from the database but rather change the status of the suspended records from written off to delete pending. The PCM_OP_SUSPENSE_SEARCH_DELETE opcode will create a schedule object to execute the PCM_OP_SUSPENSE_DEFERRED_DELETE opcode.
   - The schedule object will call the PCM_OP_SUSPENSE_DEFERRED_DELETE opcode to be executed at the deferred duration time.
   - The PCM_OP_SUSPENSE_SEARCH_DELETE or PCM_OP_SUSPENSE_DEFERRED_DELETE opcode deletes the object from the suspense db.
4. Generates a **/event/notification/suspense/delete** object that records the deletion for each suspended record that was deleted.

If successful, PCM_OP_SUSPENSE_SEARCH_DELETE generates a delete notification event that includes the administrative action POID of the delete action and returns success along with the number of records deleted. If the operation fails in any record, it cancels the entire operation and returns failure with the appropriate error code, leaving the state of the record as it was before the operation.

26

Suspense Management Utilities

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Suspense Management utilities.

load_edr_field_mapping

Use this utility to load EDR field mapping into the **/edr_field_mapping** object in the BRM database.

Location

*BRM_Home***/bin**

Syntax

```
load_edr_field_mapping [-d] [-v] [-t] [-h] XML_file
```

Parameters

-d

Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors but the data has not been loaded into the database.

-v

Displays detailed information as the utility runs.

-t

Validates the XML file. Your EDR field mapping XML file must conform to the XML schema rules in the *BRM_Home***/xsd/edr_field_mapping.xsd** file.

-h

Shows the help on the utility.

XML_file

The name and location of the EDR field mapping configuration file, which maps the EDR field name to an ID number. The default **edr_field_mapping.xml** file is in *BRM_Home***/sys/data/config** directory.

If you copy the **edr_field_mapping.xml** file to the same directory from which you run the **load_edr_field_mapping** utility, you do not have to specify either the path or the file name.

If you run the command in a different directory from where the **edr_field_mapping.xml** file is located, you must include the entire path for the file.

load_pin_suspense_editable_flds

Use this utility to load editable fields into the **/config/suspense_editable_flds** object in the BRM database. You define editable fields in the **pin_suspense_editable_flds** file in *BRM_Home*/**sys/data/config**.

For more information, see "Loading Editable Fields into the Database" and "Mapping EDR Fields to Brand Information".

Note: You cannot load separate **/config/suspense_editable_flds** objects for each brand. All brands use the same object.

Caution: The **load_pin_suspense_editable_flds** utility overwrites existing **/config/suspense_editable_flds** objects. If you are updating editable fields, you cannot load new editable fields only. You must load complete sets of editable fields each time you run the utility.

Important: To connect to the BRM database, the **load_pin_suspense_editable_flds** utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

Location

BRM_Home/**bin**

Syntax

```
load_pin_suspense_editable_flds [-d] [-v] [pin_suspense_editable_flds_file]
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors but the data has not been loaded into the database.

-v
Displays detailed information as the utility runs.

pin_suspense_editable_flds_file
The name and location of the file that defines the list of editable fields used by Suspense Management Center. The default **pin_suspense_editable_flds** file is in *BRM_Home*/**sys/data/config**.

If you copy the **pin_suspense_editable_flds** file to the same directory from which you run the **load_pin_suspense_editable_flds** utility, you do not have to specify either the path or the file name.

If you run the command in a different directory from where the **pin_suspense_editable_flds** file is located, you must include the entire path for the file.

Results

The **load_pin_suspense_editable_flds** utility notifies you when it successfully creates the **/config/suspense_editable_flds** object. Otherwise, look in the **default.pinlog** file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

To verify that the network elements were loaded, display the **/config/suspense_editable_flds** object by using Object Browser or the **robj** command with the **testnap** utility. See "Reading Objects by Using Object Browser" and "Using testnap" in *BRM Developer's Guide*.

Important: You must restart Suspense Management Center to make new editable fields available. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

load_pin_suspense_edr_fld_map

Use this utility to load brand-to-brand-ID mapping into the **/config/suspense_edr_fld_map** object in the BRM database. You define the brand ID mapping in the **pin_suspense_edr_fld_map** file in *BRM_Home***/sys/data/config**.

For more information, see "Mapping EDR Fields to Brand Information".

Note: You cannot load separate **/config/suspense_edr_fld_map** objects for each brand. All brands use the same object.

Caution: The **load_pin_suspense_edr_fld_map** utility overwrites existing brand ID mapping. If you are updating brand ID mapping, you cannot load new brand IDs only. You must load complete sets of brand mapping relationships each time you run the utility.

Important: To connect to the BRM database, the **load_pin_suspense_edr_fld_map** utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

Location

*BRM_Home***/bin**

Syntax

```
load_pin_suspense_edr_fld_map [-d] [-v] pin_suspense_edr_fld_map_file
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors but the data has not been loaded into the database.

-v
Displays detailed information as the utility runs.

pin_suspense_edr_fld_map_file
The name and location of the file that defines the brand-to-brand ID mapping. The default **pin_suspense_edr_fld_map** file is in *BRM_Home***/sys/data/config**.

If you copy the **pin_suspense_edr_fld_map** file to the same directory from which you run the **load_pin_suspense_edr_fld_map** utility, you do not have to specify either the path or the file name.

If you run the command in a different directory from where the **pin_suspense_edr_fld_map** file is located, you must include the entire path for the file.

Results

The **load_pin_suspense_edr_fld_map** utility notifies you when it successfully creates the **/config/edr_field_map** object. Otherwise, look in the **default.pinlog** file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

To verify that the network elements were loaded, display the **/config/edr_field_map** object by using Object Browser or the **robj** command with the **testnap** utility. See "Reading Objects by Using Object Browser" and "Using testnap" in *BRM Developer's Guide*.

Important: You must restart the rating pipeline to make the new brand mapping available. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

load_pin_suspense_override_reason

Use this utility to load Pipeline Manager override reasons into the **/config/suspense_override_codes** object in the BRM database. You define Pipeline Manager override reasons in the **pin_suspense_override_reason** file in *BRM_Home***/sys/data/config**.

For more information, see "Overriding Pipeline Suspense Handling Rules".

Caution: The **load_pin_suspense_override_reason** utility overwrites existing suspense override reasons. You must load complete sets of override reasons each time you run the utility.

Important: To connect to the BRM database, the **load_pin_suspense_override_reason** utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

Location

*BRM_Home***/bin**

Syntax

```
load_pin_suspense_override_reason [-d] [-v] pin_suspense_override_reason_file
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors but the data has not been loaded into the database.

-v
Displays detailed information as the utility runs.

pin_suspense_override_reason_file
The name and location of the file that defines the override reasons. The default **pin_suspense_override_reason** file is in *BRM_Home***/sys/data/config**.

If you copy the **pin_suspense_override_reason** file to the same directory from which you run the **load_pin_suspense_override_reason** utility, you do not have to specify either the path or the file name.

If you run the command in a different directory from where the **pin_suspense_override_reason** file is located, you must include the entire path for the file.

Results

The **load_pin_suspense_override_reason** utility notifies you when it successfully creates the **/config/suspense_override_codes** object. Otherwise, look in the **default.pinlog** file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

To verify that the network elements were loaded, display the **/config/suspense_override_codes** object by using Object Browser or the **robj** command with the **testnap** utility. See "Reading Objects by Using Object Browser" and "Using testnap" in *BRM Developer's Guide*.

Important: You must restart Suspense Management Center to enable it to use the new suspense override reasons. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

load_pin_suspense_params

Use this utility to load system-level configuration information for Suspense Manager into the **/config/suspense_params** object in the BRM database. You define the system parameters for Suspense Manager, such as the number of records to process in each opcode call, in the **pin_suspense_params** file in the *BRM_Home*/**sys/data/config** directory.

For more information, see "Configuring the Number of Suspended Records to Process in a Transaction" and "Processing Suspended Records in Multiple Steps".

Note: You cannot load separate **/config/suspense_params** objects for each brand. All brands use the same object.

Caution: The **load_pin_suspense_params** utility overwrites existing Suspense Manager system parameters. You must load complete sets of parameters each time you run the utility.

Important: To connect to the BRM database, the **load_pin_suspense_params** utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

Location

BRM_Home/**bin**

Syntax

```
load_pin_suspense_params [-d] [-v] filename
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors but the data has not been loaded into the database.

-v
Displays detailed information as the utility runs.

filename
The name of the text file containing the configuration parameters for suspense management. The default file name is **pin_suspense_params**.

Results

The **load_pin_suspense_params** utility notifies you when it successfully creates the **/config/suspense_params** object. Otherwise, look in the **default.pinlog** file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

To verify that the data was loaded, display the **/config/suspense_params** object by using Object Browser or the **robj** command with the **testnap** utility. See "Reading Objects by Using Object Browser" and "Using testnap" in *BRM Developer's Guide*.

load_pin_suspense_reason_code

Use this utility to load suspense reasons and subreasons into the **/config/suspense_reason_code** object in the BRM database. You define suspense reasons and subreasons in the **pin_suspense_reason_code** file in *BRM_Home*/**sys/data/config/suspense_reason_code**.

For more information, see "Changing the List of Suspense Reasons and Subreasons".

Note: You cannot load separate **/config/suspense_reason_code** objects for each brand. All brands use the same object.

Caution: The **load_pin_suspense_reason_code** utility overwrites existing suspense reason and subreason codes. If you are updating suspense reason and subreason codes, you cannot load new codes only. You must load complete sets of codes each time you run the utility.

Important: To connect to the BRM database, the **load_pin_suspense_reason_code** utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

Location

BRM_Home/**bin**

Syntax

```
load_pin_suspense_reason_code [-d] [-v] pin_suspense_reason_code_file
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors but the data has not been loaded into the database.

-v
Displays detailed information as the utility runs.

pin_suspense_reason_code_file
The name and location of the file that defines suspense reasons and subreasons. The default **pin_suspense_reason_code** file is in *BRM_Home*/**sys/data/config**.

If you copy the **pin_suspense_reason_code** file to the same directory from which you run the **load_pin_suspense_reason_code** utility, you do not have to specify either the path or the file name.

If you run the command in a different directory from where the **pin_suspense_reason_code** file is located, you must include the entire path for the file.

Results

The **load_pin_suspense_reason_code** utility notifies you when it successfully creates the **/config/suspense_reason_code** object. Otherwise, look in the **default.pinlog** file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

To verify that the network elements were loaded, display the **/config/suspense_reason_code** object by using Object Browser or the **robj** command with the **testnap** utility. See "Reading Objects by Using Object Browser" and "Using testnap" in *BRM Developer's Guide*.

Important: You must restart Pipeline Manager to make the new suspense reason codes available. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Important: If you are changing the suspense reason or subreason codes, you must also modify the **suspense_reason_code.en_US** file and run the **load_localized_strings** utility. See "Configuring Suspense Manager".

load_pin_batch_suspense_override_reason

Use the **load_pin_batch_suspense_override_reason** utility to load batch suspense override-able reason codes into the **/config/batch_suspense_override_reason** object in the BRM database. You define batch suspense override reason codes in the **pin_batch_suspense_override_reason** file in *BRM_Home***/sys/data/config**. By default, no reason can be overridden, so the file is a placeholder.

Note: You cannot load separate **/config/batch_suspense_override_reason** objects for each brand. All brands use the same object.

Caution: The **load_pin_batch_suspense_override_reason** utility overwrites the existing **/config/batch_suspense_override_reason** object in the BRM database. If you are updating the **/config/batch_suspense_override_reason** object, you must load complete sets of batch suspense override-able reasons each time.

Important: To connect to the BRM database, the utility needs the Connection Manager (CM) to be up and running.

Location

*BRM_Home***/bin**

Syntax

```
load_pin_batch_suspense_override_reason [-d] [-v] pin_batch_suspense_override_reason_file
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors but the data has not been loaded into the database.

-v
Displays information about successful or failed processing as the utility runs.

Note: This parameter is always used in conjunction with other parameters and commands. It is not position dependent. For example, you can enter **-v** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace *filename***.log** with the name of the log file:

load_pin_batch_suspense_override_reason *any_other_parameter* **-v >** *filename***.log**

pin_batch_suspense_override_reason_file
The name and location of the file that defines batch suspense override-able reason codes. The default **pin_batch_suspense_override_reason** file is in *BRM_Home*/**sys/data/config**.

If you do not run the utility from the directory in which the file is located, you must include the complete path to the file.

> **Tip:** If you copy the **pin_batch_suspense_override_reason** file to the directory from which you run the **load_pin_batch_suspense_override_reason** utility, you don't have to specify the path or file name. The file must be named **pin_batch_suspense_override_reason**.

Results

If the utility does not notify you that it was successful, look in the **default.pinlog** file to find any errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

To verify that the override reason codes were loaded, display the **/config/batch_suspense_override_reason** object by using Object Browser or the **robj** command with the **testnap** utility. See "Reading Objects by Using Object Browser" and "Using testnap" in *BRM Developer's Guide*.

The following is an example of a **pin_batch_suspense_override_reason** file, which would be an input for this utility and could be compared to the output:

```
# Override Suspense Reason
00001
00002
00003
```

load_pin_batch_suspense_reason_code

Use the **load_pin_batch_suspense_reason_code** utility to load batch suspense reason codes into the **/config/batch_suspense_reason_code** object in the BRM database. You define batch suspense reasons in the **pin_batch_suspense_reason_code** file in *BRM_Home***/sys/data/config**. BRM uses suspense reason codes to load suspense reasons into a batch suspense record when a call details record (CDR) file is suspended.

Note: You cannot load separate **/config/batch_suspense_reason_code** objects for each brand. All brands use the same object.

Caution: The **load_pin_batch_suspense_reason_code** utility overwrites the existing **/config/batch_suspense_reason_code** object in the BRM database. If you are updating load batch suspense reason codes, you cannot load new batch suspense reason codes only. Therefore, you must load a complete set of load batch suspense reason codes each time you run the utility.

Important: The **load_pin_batch_suspense_reason_code** utility must be connected to a running CM to load batch suspense reason codes into the Infranet database.

Location

*BRM_Home***/bin**

Syntax

```
load_pin_batch_suspense_reason_code [-d] [-v] pin_batch_suspense_reason_code_file
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors but the data has not been loaded into the database.

-v
Displays information about successful or failed processing as the utility runs.

Note: This parameter is always used in conjunction with other parameters and commands. It is not position dependent. For example, you can enter **-v** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace *filename***.log** with the name of the log file:

load_pin_batch_suspense_reason_code *any_other_parameter* **-v >** *filename***.log**

pin_batch_suspense_reason_code_file

The name and location of the file that defines the batch suspense reason codes. The default **pin_batch_suspense_reason_code** file is in *BRM_Home*/**sys/data/config**.

If you do not run the utility from the directory in which the file is located, you must include the complete path to the file.

> **Tip:** If you copy the **pin_batch_suspense_reason_code** file to the directory from which you run the **load_pin_batch_suspense_reason_code** utility, you do not have to specify the path or file name. The file must be named **pin_batch_suspense_reason_code**.

Results

The **load_pin_batch_suspense_reason_code** utility notifies you when it successfully creates the **/config/batch_suspense_reason_code** object. Otherwise, look in the **default.pinlog** file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

To verify that the elements were loaded, display the **/config/batch_suspense_reason_code** object by using Object Browser or the **robj** command with the **testnap** utility. See "Reading Objects by Using Object Browser" and "Using testnap" in *BRM Developer's Guide*.

The following example shows sample entries from the **/config/batch_suspense_reason_code** object:

```
PIN_FLD_POID                     POID [0] 0.0.0.1 /config/batch_suspense_reason_code 93712 0
0 PIN_FLD_CREATED_T              TSTAMP [0] (1153474255) 21/07/2006 15:00:55:000 PM
0 PIN_FLD_MOD_T                  TSTAMP [0] (1153474255) 21/07/2006 15:00:55:000 PM
0 PIN_FLD_READ_ACCESS               STR [0] "G"
0 PIN_FLD_WRITE_ACCESS              STR [0] "S"
0 PIN_FLD_ACCOUNT_OBJ              POID [0] 0.0.0.1 /account 1 0
0 PIN_FLD_DESCR                     STR [0] ""
0 PIN_FLD_HOSTNAME                  STR [0] "-"
0 PIN_FLD_NAME                      STR [0] "batch_suspense_reason_code"
0 PIN_FLD_OP_CORRELATION_ID         STR [0]
"2:CT1255:UnknownProgramName:0:AWT-EventQueue-0:3:1153836497:0:root.0.0.0.1::user1:123456789"
0 PIN_FLD_PROGRAM_NAME              STR [0] "load_pin_batch_suspense_reason_code"
0 PIN_FLD_VALUE                     STR [0] ""
0 PIN_FLD_VERSION                   STR [0] "1"
0 PIN_FLD_SUSPENSE_REASONS        ARRAY [0] allocated 1, used 1
1     PIN_FLD_SUSPENSE_REASON      ENUM [0] 0
0 PIN_FLD_SUSPENSE_REASONS        ARRAY [471] allocated 1, used 1
1     PIN_FLD_SUSPENSE_REASON      ENUM [0] 1
0 PIN_FLD_SUSPENSE_REASONS        ARRAY [119] allocated 1, used 1
1     PIN_FLD_SUSPENSE_REASON      ENUM [0] 2
0 PIN_FLD_SUSPENSE_REASONS        ARRAY [120] allocated 1, used 1
1     PIN_FLD_SUSPENSE_REASON      ENUM [0] 2
0 PIN_FLD_SUSPENSE_REASONS        ARRAY [126] allocated 1, used 1
1     PIN_FLD_SUSPENSE_REASON      ENUM [0] 2
0 PIN_FLD_SUSPENSE_REASONS        ARRAY [127] allocated 1, used 1
1     PIN_FLD_SUSPENSE_REASON      ENUM [0] 2
0 PIN_FLD_SUSPENSE_REASONS        ARRAY [147] allocated 1, used 1
1     PIN_FLD_SUSPENSE_REASON      ENUM [0] 2
0 PIN_FLD_SUSPENSE_REASONS        ARRAY [148] allocated 1, used 1
1     PIN_FLD_SUSPENSE_REASON      ENUM [0] 2
```

27

Recycling EDRs in Pipeline-Only Systems

This chapter describes how to configure and use EDR recycling. EDR recycling is the Oracle Communications Billing and Revenue Management (BRM) feature used by systems that use Pipeline Manager, but do not store suspended EDRs in the BRM database. The pipeline-only recycling feature uses the FCT_PreRecycle module to mark EDRs for recycling, and the FCT_Recycle module to send the rejected EDRs to a file for processing by hand.

Systems using BRM with Pipeline Manager use either the standard recycling tools or the Suspense Manager service integration component for recycling and deleting EDRs.

For details, see "About the EDR Recycling Features".

For details on how to use FCT_Reject to reject EDRs, see "FCT_Reject".

Before reading this document, you should be familiar with how Pipeline Manager works and how to configure it. See "Configuring Pipeline Manager" in *BRM System Administrator's Guide*.

About Recycling EDRs

When processing a CDR file, there might be non-valid EDRs in the file, or your pipelines might not be set up correctly to handle certain EDRs. You use EDR recycling to fix configuration problems and re-process EDRs.

The recycling process uses these pipeline modules:

- FCT_Reject
- FCT_PreRecycle
- FCT_Recycle

Overview of EDR recycling:

1. You start Pipeline Manager with the FCT_PreRecycle, FCT_Recycle, and FCT_Reject modules active. (The FCT_PreRecycle and FCT_Recycle modules do nothing until you start the recycle process by using a semaphore.)
2. When an EDR is processed, a module may find an error in the EDR. The error is appended to the EDR, and a flag is set to indicate that the EDR has an error. The EDR is sent to the next module. Each module adds errors, if any more are found.
3. The FCT_Reject module analyzes the errors in the EDR. If necessary, the EDR is moved to a reject file.
4. You examine the errors and determine how to reconfigure Pipeline Manager to prevent the errors.

5. You use a semaphore file entry to start the pre-recycling process. This sends the rejected EDRs through the pipeline again. The FCT_PreRecycle module adds a flag to the EDR to let the other modules know that the EDR is being recycled.

   You can pre-recycle and recycle EDRs in test mode or real mode. Typically, you run the pre-recycle and recycling processes in test mode first, to see if the errors have been fixed. When there are no longer any errors, you pre-recycle and recycle in real mode.

6. The FCT_Recycle module runs at the end of the pipeline It does one of the following:

   - In test mode, the module creates a report about the processing, but does not send the EDRs to an output file.
   - In recycle mode, the module sends the results to an output file, and attaches a sequence number to the output file.

   Note: You can configure the output module to send an entire file to the error directory if it includes a lot of errors. You can configure the threshold for the number of errors allowed per file. See "Specifying the Maximum Errors Allowed in an Input File".

How the FCT_Reject Module Works

The FCT_Reject module must be run after all rating and enrichment modules. It should be run as the second-to-last function module in the pipeline (the last function module is the FCT_Suspense module). This is because all potential errors must be found before the FCT_Reject module processes the EDRs.

You can run the FCT_Reject module from the registry or by using a semaphore file entry.

The FCT_Reject module does the following:

1. The FCT_Reject module checks the error status of the EDR. If the EDR contains an error status with a *warning* or *critical* severity, the EDR is rejected. The FCT_Reject module changes the value of the DETAIL.DISCARDING field from 0 to 1.

   Note: If the DETAIL.DISCARDING field is already set to 1, the EDR was rejected in a previous pass through the pipeline, and is rejected again.

   If the error type in the EDR is not identified in the registry **StreamMap** entry, the EDR is sent to default reject stream.

2. By default, the EDR is moved to the reject stream, as identified in the **RejectStream** registry entry. The EDR is stored in a file that is used by the recycling modules. EDRs can also be rejected in the recycle process.

   If the reject stream is not specified, the EDR is moved to the normal output stream, but the discard field is set to **1**, indicating that the EDR has been rejected.

Using a Reject Output Stream

Use the **UseRejectStream** entry to specify how to handle rejected EDRs. You can do the following:

Important: If you use **UseRejectStream**, you must use the **StreamMap** entry.

Specifies whether to use the reject output stream:

- **True**. Rejected EDRs are sent to the reject stream.
- **False**. Rejected EDRs are sent to the normal output stream, but flagged as discarded.

Specifying Multiple Reject Streams

By default, rejected EDRs are sent to a single reject stream. However, you can use the **StreamMap** registry entry to specify separate reject streams for different types of errors.

Important: If you use **StreamMap**, you must use the **UseRejectStream** entry.

For example, this entry sends errored TAP records to the output stream named **Rap0101Output**.

```
ERR_TAP3_RET = Rap0101Output
```

The output stream must be configured.

Recycling Assembled EDRs

If you use both the FCT_CallAssembling module and the FCT_Reject module in a pipeline, use the optional FCT_Reject module **CallAssemblingModule** registry entry to ensure that the complete EDRs are recycled. Otherwise, only part of the EDR is recycled.

The FCT_Reject **CallAssemblingModule** registry entry is a pointer to the FCT_CallAssembling module, for example:

```
CallAssemblingModule = ifw.Pipelines.Pipe.Functions.Standard.FunctionPool.CallAssembling
```

Processing EDRs with Errors

Use the FCT_Reject **MinErrorSeverity** registry entry to reject EDRs that have a specified severity. This allows the EDR to be processed with warning or normal error messages without being rejected.

You can specify one of the following:

- **-1** = undef
- **0** = debug
- **1** = normal
- **2** = warning
- **3** = minor
- **4** = major
- **5** = critical

To allow warning and normal messages without rejecting the EDR, set this entry to **3**. Valid values for **MinErrorSeverity** are 3, 4, and 5.

By default, this entry is not used.

How the FCT_PreRecycle Module Works

The FCT_PreRecycle module is always the first module in the pipeline.

Although you can *activate* the FCT_PreRecycle module from the startup registry, you cannot run the FCT_PreRecycle module from the startup registry; you must *run* it by using a semaphore file.

The FCT_PreRecycle module does the following:

1. The module gets the file of rejected EDRs from the reject stream output directory.
2. The module puts the reject EDR file into the input directory for recycling. It uses the same input directory as the incoming CDR files. It adds a recycle suffix to the file and a sequence number, so the original input file in the output directory cannot be overwritten.

   You can recycle all EDRs in the reject directory, or list specific files to recycle. See "Recycling EDRs".
3. For each EDR to recycle, the module sets a value in the INTERN_PROCESS_STATUS field to indicate that the EDR is being recycled. This tells the FCT_Recycle module which EDRs to process, and allows the discounting modules to recalculate discount amounts correctly.
   - The value is set to **1** if the EDR is being recycled.
   - The value is set to **2** if the recycling is in test mode.

You can recycle all EDRs in the reject directory, or list specific files to recycle. See "Sample Semaphore File Entries".

How the FCT_Recycle Module Works

The FCT_Recycle module is the last function module in the pipeline, before the output.

You activate the FCT_Recycle module from the startup registry, but it does nothing until the FCT_PreRecycle module starts the recycling process.

The FCT_Recycle module reads the INTERN_PROCESS_STATUS field for each EDR.

- If the value is **2**, recycling is in test mode. The FCT_Recycle module doesn't send the EDRs to an output directory. Instead, the FCT_Recycle module creates a report with the following data:
  - Stream name.
  - Total number of processed EDRs.
  - Number of EDRs that can be recycled without an error.
  - Number of EDRs that still generate an error.
  - List of all errors.
  - The wholesale charge amount from all successfully recycled EDRs. (This data is taken from the WHOLESALE_CHARGE field.)
  - The wholesale charge amount from all EDRs that still have errors. (This data is taken from the WHOLESALE_CHARGE field.)

  - The total duration for all successfully recycled EDRs. (This data is taken from the DURATION_MINUTES field.)
  - The total duration from all EDRs that still have errors. (This data is taken from the DURATION_MINUTES field.)

  You can use this data to determine if the EDRs are worth further configuration and processing.

- If the value is **1**, recycling occurs. All EDRs are processed as usual, with the following differences in comparison to normal input file processing:
  - A sequence number is generated.
  - The sequence offset value is generated.
  - The sequence check is inactivated.

  For more information, see "Checking And Generating Sequence Numbers" in *BRM System Administrator's Guide*.

Testing Recycling EDRs

Once you have determined that EDRs have been rejected, the first step is to correct any pipeline problems that caused the problem. After that you usually test recycle the CDR file to ensure that the changes have had the desired affect.

Follow these steps to test recycle EDRs:

1. Configure the FCT_Reject module. See "FCT_Reject".

   Typically, rejected EDRs are sent to the reject stream. You configure the reject stream in the registry in the following places:

   - In the FCT_Reject module pipeline configuration
   - In the Output stream

   For a sample Output stream configuration see "Sample Output Configuration".

   Important: When you test recycling, first inactivate the FCT_Reject module.

2. Configure the FCT_PreRecycle module. See "FCT_PreRecycle".

   You configure the reject stream in the registry in the following places:

   - In the FCT_PreRecycle module pipeline configuration.
   - In the input stream.

   The module uses the same input configuration as the incoming CDR files, so you don't need to configure a separate input stream.

3. Configure the FCT_Recycle module. See "FCT_Recycle".

   Configure the FCT_Recycle module RecycleLog registry entry to specify the message file parameters. These settings are specified in the **ProcessLog** registry section. For more information, see "LOG".

4. Use a semaphore to inactivate the FCT_Reject module:

```
Module
Reject.Module.Active = False
```

5. Use a semaphore to run the FCT_PreRecycle module in test mode:

```
Recycle.Module.RecycleTest {}
```

6. Review the log files that you configured in FCT_Recycle for errors, and repeat these steps as necessary.

Recycling EDRs

When you finish test recycling EDRs, follow these steps to do the actual recycling:

1. Configure the FCT_Reject module. See "FCT_Reject".

   Typically, rejected EDRs are sent to the reject stream. You configure the reject stream in the registry in the following places:

   - In the FCT_Reject module pipeline configuration
   - In the Output stream

   For a sample Output stream configuration see "Sample Output Configuration".

   Important: When you test recycling, first inactivate the FCT_Reject module.

2. Configure the FCT_PreRecycle module. See "FCT_PreRecycle".

   You configure the reject stream in the registry in the following places:

   - In the FCT_PreRecycle module pipeline configuration.
   - In the input stream.

   The module uses the same input configuration as the incoming CDR files, so you don't need to configure a separate input stream.

3. Configure the FCT_Recycle module. See "FCT_Recycle".

   Configure the FCT_Recycle module **RecycleLog** registry entry to specify the message file parameters. These settings are specified in the **ProcessLog** registry section. For more information, see "LOG".

4. Use a semaphore to run the FCT_PreRecycle module.

   You can recycle all EDRs in the reject directory, or list specific files to recycle. See "Sample Semaphore File Entries".

5. Review the log files that you configured in FCT_Reject for errors.

Part IV

Twin Talk Enabler

Part IV describes how to configure Twin Talk Enabler in an Oracle Communications Billing and Revenue Management (BRM) system. It contains the following chapters:

- About Twin Talk Enabler
- Sample Twin Talk Enabler Configuration Procedures

28

About Twin Talk Enabler

This chapter describes how to install, configure and use Oracle Communications Billing and Revenue Management (BRM) Twin Talk Enabler to offer twin talk services to your customers.

Before you read this document, you should read "About managing prepaid services and extended rating attributes" in *BRM Telco Integration*.

About Twin Talk Enabler

Twin Talk Enabler allows you to use BRM to rate and bill your customers' twin talk service usage.

> **Note:** Twin Talk Enabler uses batch pipeline rating. Real-time rating is not supported for rating twin talk service usage.

A twin talk service allows subscribers to direct usage charges for outbound calls to a primary or secondary account, such as a business or personal account. For example, if a company issues an employee a mobile phone with twin talk service enabled, the employee can use the phone to make business and personal calls.

You set up the service so that usage charges for each type of call are directed to the correct account. For example, you can configure the twin talk service so that if the subscriber dials two pound signs (##) as a suffix to the outbound number, the usage charges for the call are directed to the customer's personal (secondary) account. If no suffix is dialed, BRM assumes it is a business call, and the business (primary) account is charged.

> **Note:** Only usage fees can be charged to a secondary account. One-time, monthly fees and other non-usage fees are always charged to the primary account.

You can configure other criteria that determine the account to charge, such as time of day or day of week. For example, calls made during business hours could be charged to the subscriber's business account, and calls made outside these hours could be charged to the subscriber's personal account.

About Configuring Services for Primary and Secondary Accounts

This document describes how to set up twin talk secondary accounts that use:

- The same service that the primary account uses.
- A different (secondary) service.

> **Note:** Oracle recommends that you configure secondary accounts to use the same service that the primary account uses, since it involves far fewer configuration steps.

Data Flow Overview

A system configured for twin talk service includes a custom iScript placed just before the **FCT_Account** module in the pipeline. This custom iScript does the following:

> **Note:** The name and location of the iScript file are configurable in the registry file. This document uses the sample iScript file **ISC_TwinTalkEnabler.isc** in the *Pipeline_Home*/i**ScriptLib/iScriptLib_Standard** directory. *Pipeline_Home* is the directory where you installed Pipeline Manager.

1. Checks each incoming EDR for twin talk attributes.

> **Note:** Twin talk attributes are typically associated with the A or B number, but you can associate the attributes with any EDR field depending on your requirements. This document assumes that the twin talk attributes are associated with the B number unless otherwise noted.

2. If a twin talk attribute exists, uses an EDR field, such as the A number, together with the enabler functions to retrieve twin talk account or service profiles.
3. If a twin talk account or service profile is found, returns ERAs such as time of day and the affected account.

> **Note:** The **ISC_TwinTalkEnabler.isc** iScript defines the selection criteria to select the secondary account.

4. If a secondary account is determined:
   - Replaces the INTERN_A_NUMBER_ZONE value in the EDR with the login ID of the secondary account.

> **Note:**
>
> - By default, the **service/telco/gsm/telephony**, **/sms**, and **/data** services are configured to use the A_NUMBER value to retrieve customer details. In this case, your iScript can replace the A_NUMBER value with a new secondary account login ID. There is no need to change the INTERN_A_NUMBER_ZONE value.
> - The sample iScript (**ISC_TwinTalkEnabler.isc**) changes the INTERN_SERVICE_CODE and INTERN_A_NUMBER_ZONE values.

- If the primary and secondary accounts use different services, replaces the INTERN_SERVICE_CODE value in the EDR with the service code of the secondary account.

5. Passes the EDR to the next module in the pipeline.

   The replaced account values in the EDR cause the secondary account to be charged.

If no twin talk accounts or service profiles are found, or no secondary account is selected by the **ISC_TwinTalkEnabler.isc** iScript, the EDR is passed to the next module without any changes and the primary account is charged.

For more information on custom iScript behavior, see "Creating an iScript to Support Twin Talk".

Twin Talk Enabler Components

Twin Talk Enabler includes the following components:

- **Two new Twin Talk Enabler functions (getServExtRating and getAcctExtRating).** You use these functions in the custom iScript.
- A sample wireless application that includes:
  - A sample custom iScript (*Pipeline_Home*/**iScriptLib/iScriptLib_Standard/ISC_TwinTalkEnabler.isc**). This iScript determines which account should be charged and updates the ERA fields accordingly by using account- or service-level twin talk profiles returned by Twin Talk Enabler functions.
  - An updated sample wireless registry file (*Pipeline_Home***/conf/wireless.reg**).

Configuring a Twin Talk Service

To configure a twin talk service, follow the steps in these sections:

1. Configuring BRM for Twin Talk
2. Configuring Pipeline Manager for Twin Talk
3. Configuring Twin Talk Pricing
4. Creating an iScript to Support Twin Talk

Configuring BRM for Twin Talk

To configure BRM for twin talk, follow the steps in these sections:

> **Important:** Perform steps 1, 2, and 3 only if secondary accounts use a secondary service.

1. (For secondary service only) Defining a Twin Talk Service Type
2. (For secondary service only) Configuring and Loading Twin Talk Event Mappings
3. (For secondary service only) Configuring and Loading Twin Talk Billing Items
4. Creating and Loading Twin Talk Extended Rating Attributes Names
5. Configuring and Loading Twin Talk Provisioning Tags

6. Create Aliases to be Used by Secondary Accounts

Defining a Twin Talk Service Type

Define a service to be used only for twin talk service, for example:

- **/service/twin**
- **/service/telco/gsm/voicetwintalk**
- **/service/telco/gsm/smstwintalk**

See "Adding Support For A New Service" in *BRM Developer's Guide*.

Configuring and Loading Twin Talk Event Mappings

To configure twin talk event map information:

1. Open the event map configuration file (*event_map_config_file*) that you used to configure your telco service.
2. Add the following lines to map the twin talk service to events. Use this format:

```
/service/twintalk_service
 : /event/session/telco/gsm                    : Event description
 : /event/delayed/session/telco/gsm            : Event description
```

For example:

```
/service/telco/gsm/voicetwintalk
 : /event/session/telco/gsm                    : Real Time Telco GSM Session
 : /event/delayed/session/telco/gsm            : Delayed Telco GSM Session
```

Important: Be sure to add the twin talk event mapping text to the event map file you used to configure your telco services.

3. Save the file.
4. Load the event map by using the load_event_map utility:

```
load_event_map event_map_config_file
```

For more information, see "load_event_map" in *BRM Setting Up Pricing and Rating*.

Configuring and Loading Twin Talk Billing Items

To configure twin talk custom bill items:

1. Open the custom bill item tags configuration file (*BRM_Home*/**sys/data/pricing/example/config_item_tags.xml**) and define a tag for the new service. *BRM_Home* is the directory where you installed BRM components. For example:

```
<ItemTagElement>
   <ItemTag>twintalk</ItemTag>
   <EventType>/event/*</EventType>
   <ServiceType>/service/telco/gsm/twintalk</ServiceType>
</ItemTagElement>
```

2. Save the file.

3. Load the configuration item tags by using the load_config_item_tags utility.

   ```
   load_config_item_tags config_item_tags.xml
   ```

4. Open the custom bill item types configuration file (*BRM_Home*/**sys/data/pricing/example/config_item_types.xml**) and define a type for the item. For example:

   ```
   <ItemTypeElement>
      <ItemTag>twintalk</ItemTag>
      <ItemDescription>twintalk</ItemDescription>
      <ItemType precreate="true" type="cumulative">/item/misc</ItemType>
   </ItemTypeElement>
   ```

5. Save the file.

6. Load the configuration item types by using the load_config_item_types utility:

   ```
   load_config_item_types config_item_types.xml
   ```

Creating and Loading Twin Talk Extended Rating Attributes Names

When you create IDs, names and descriptions for twin talk extended rating attributes (ERAs), you edit the **era_descr.en_US** sample file in the *BRM_Home*/**sys/msgs/eradescr** directory. The following sample entry defines a twin talk account ERA in the provisioning tags section of the file:

```
[]  STR
        ID = 30 ;
        VERSION = 1 ;
        STRING = "TWINTALK_ACCOUNT" ;
END
STR
        ID = 31 ;
        VERSION = 1 ;
        STRING = "To enable Twin Talk Account provisioning tags." ;
END
```

For more information, see "About managing prepaid services and extended rating attributes" in *BRM Telco Integration*

After you customize the file, you use the load_localized_strings utility to load the contents of the **era_descr.***locale* file into the **/strings** object.

> **Note:** Default ERA names and descriptions are loaded when you install GSM Manager. You need to load them again only if you customize them.

When you run the **load_localized_strings** utility, use this command:

```
load_localized_strings era_descr.locale
```

> **Note:** If you're loading a localized version of this file, use the correct file extension for your locale. For a list of file extensions, see "Locale Names" in *BRM Developer's Guide*.

For information on loading the **era_descr.***locale* file, see "Loading Localized or Customized Strings" in *BRM Developer's Guide*. For information on creating new strings for this file, see "Creating New Strings and Customizing Existing Strings" in *BRM Developer's Guide*.

Configuring and Loading Twin Talk Provisioning Tags

To configure twin talk provisioning tags:

1. Open the provisioning tags configuration file that you used to configure your telco service.

   Caution: If you don't use the configuration file you used to configure your telco service, you might lose configuration information when you load the file with the load_pin_telco_tags utility. For more information, see "load_pin_telco_tags" in *BRM Telco Integration*.

2. Add the appropriate provisioning tags. The following sample entry specifies the provisioning tag TWINTALK_ACCOUNT:

   ```
   account_era "TWINTALK_ACCOUNT" "30" "31"
   ```

   Note:

   - If you are using the same service for both the primary and secondary accounts, add the new twin talk provisioning tags to the configuration for your existing primary service.
   - If you are creating a new service for twin talk, add the required provisioning tags for the new services (in addition to the primary service).

3. Save the file.
4. Load the provisioning tags by using the load_pin_telco_tags utility:

   ```
   load_pin_telco_tags prov_tags_config_file
   ```

   If you are using the same service for primary and secondary twin talk accounts, this utility updates the service configuration object, for example, **/config/telco/gsm/telephony**.

   If you are creating a new service for twin talk, this utility creates a service configuration object, for example **/config/telco/gsm/voicetwintalk**, for the new service.

For more information, see "About managing prepaid services and extended rating attributes" and "load_pin_telco_tags" in *BRM Telco Integration*.

Create Aliases to be Used by Secondary Accounts

Use the PCM_OP_CUST_SET_LOGIN opcode to create aliases.

For more information, see "Creating Secondary Accounts with a Different (Twin Talk) Service".

Configuring Pipeline Manager for Twin Talk

> **Note:** This section only applies if secondary accounts use a secondary service.

To configure Pipeline Manager to support twin talk services, start Pricing Center and follow the steps in these sections:

1. Activating Twin Talk in the Pipeline Manager Registry
2. Defining the Twin Talk Service
3. Mapping the Twin Talk Service to a Usage Event
4. Defining the EDR Container
5. Setting Up a Rate Plan for the Twin Talk Service

Activating Twin Talk in the Pipeline Manager Registry

1. Open the registry file (*Pipeline_Home***/conf/wireless.reg)** with a text editor such as **vi**.
2. If this is a new installation, or you aren't using a pre-existing registry file, go to the **TwinTalkPlugIn** section and change the **Active** parameter to **True**.

> **Note:** Use the sample registry (*Pipeline_Home***/conf/wireless.reg)** when you start the pipeline.

3. If you are using a pre-existing registry file, copy the **TwinTalkPlugIn** section from the *Pipeline_Home***/conf/wireless.reg** file to your pre-existing registry file. Put the text just before the **FCT_Account** section. See "Sample Registry Entry for Twin Talk".

Defining the Twin Talk Service

To define the twin talk service:

1. In Pricing Center, choose **View - Pipeline Setup Toolbox - Product and Service - Service**.
2. In the Service window, add a record for the twin talk service. For example:
   - In the **Service Code** column, specify **TTK**.
   - In the and **PIN Service Type** column, specify **/service/***twintalk_service***.**

Mapping the Twin Talk Service to a Usage Event

To map the twin talk service to a usage event:

1. In Pricing Center, choose **View - Pipeline Setup Toolbox - Product and Service - Reference mapping**.
2. In the Reference Mapping window, add a reference map record. For example:
   - In the **Reference Object** column, specify **/service/***twintalk_service.*

- In the **Reference Parameter** column, specify **/event/delayed/session/telco/gsm.**

Defining the EDR Container

To define the EDR container:

1. In Pricing Center, choose **Pipeline Setup Tools - EDR - EDR Container Description**.
2. Select the **ALL_RATE** EDR container description record and click **Edit**.

   Note: **ALL_RATE** is the default pipeline name configured in the **wireless.reg** file during installation.

3. Go to the **Alias Mapping** tab.
4. Add a record that describes the EDR container. For example:
   - In the **Reference** column, specify **UniData_CustA.**
   - In the **Key** column, specify **TTK**.

     Note: This value must match the one you specified for **Service Code** in step 2 of "Defining the Twin Talk Service".

   - In the **Field ID** column, specify **DETAIL.INTERN_A_NUMBER_ZONE.**

   For more information, see "Adding Customer Balance Impact Data to EDRs" in *BRM Setting Up Pricing and Rating*.

Setting Up a Rate Plan for the Twin Talk Service

To set up a rate plan for the twin talk service:

1. In Pricing Center, choose **View - Pipeline Toolbox - Rate Plan**.
2. Add a new rate plan. For the code field, specify an appropriate code, such as **TwinTal**k.

For more information on configuring pricing information, see Pricing Center Help.

Configuring Twin Talk Pricing

Important: This section only applies if you are creating secondary accounts with secondary services.

To configure a price list to include twin talk service:

1. Start Pricing Center, create a product that includes the twin talk service:
   a. In the Event Map section of the **General Product Info** tab, add a new row.
   b. Specify **Delayed Telco GSM Session** in the **Event** column.
   c. Specify the rate plan you created in "Setting Up a Rate Plan for the Twin Talk Service".

2. Create a deal and a plan for the twin talk product.
3. Add the plan to your plan list.

   For more information on adding services to a price list, see Pricing Center Help.

Creating an iScript to Support Twin Talk

To create a customized iScript to support your twin talk service.

1. Add the following line to your iScript to include the ERA extension interface:

   ```
   use IXT_Era;
   ```

2. Configure the registry name of the **DAT_AccountBatch** module by using the **setDAT_AccountModule()** function in the **BEGIN** function in the iScript.

For general twin talk iScript requirements, see "Data Flow Overview".

For a sample Twin Talk Enabler iScript file, see *Pipeline_Home*/**iScriptLib/iScriptLib_Standard/ISC_TwinTalkEnabler.isc**.

For general information on creating custom iScripts, see "Creating iScripts and iRules" in *BRM Developer's Guide*.

About Using the onDetailEdr Function to Implement Your Twin Talk Logic

Use the **onDetailEdr** function in your iScript to implement your twin talk logic. This function:

1. Uses the **getServExtRating()** or **getAcctExtRating()** functions to retrieve the twin talk ERAs.
2. If the string "##" is in the A number, this function determines the secondary account by searching for required ERAs, for example, OVERRIDE_ACCT.
   - If the required twin talk ERAs are found, this function uses them to determine a twin talk (secondary) account login and service and substitutes these values for those in the corresponding EDR Container fields.
   - If twin talk ERAs are not found or no secondary account is selected in the twin talk iScript, the EDR is passed to the next module and usage is charged to the to primary account.
3. If the string "##" is not in the A number, this function searches for time-of-day ERAs, such as AM_ACCT,000 and PM_ACCT,001). Depending on the time of usage and the ERAs found, this function determines a twin talk (secondary) account login and service and substitutes these values for those in the corresponding EDR container fields.

Any secondary account identified by the **onDetailEdr** function is subsequently charged for this usage.

Sample Registry Entry for Twin Talk

This example registry text shows how to specify an iScript called **ISC_TwinTalkEnabler.isc**:

```
TwinTalkPlugIn
            {
              ModuleName = FCT_IScript
              Module
              {
```

```
  Active = True
  Source = FILE
  Scripts
  {
    TwinTalkIScript
    {
     FileName = ./iScriptLib/iScriptLib_Standard/ISC_TwinTalkEnabler.isc
    }
  }
}
```

Creating Twin Talk Accounts

To create primary and secondary twin talk accounts:

1. Start Customer Center.
2. Create a primary account with a product, such as Standard GSM, that includes a primary service such as **/service/gsm/telephony**.
3. Associate a SIM Card and Number, such as 004912345678, with the service.
4. Create the secondary account by using one of the following methods, depending on your iScript logic and ERAs for the TWINTALK profile:
   - Creating Secondary Accounts with a Different (Twin Talk) Service
   - Creating Secondary Accounts that Use the Same Service as the Primary Account

Creating Secondary Accounts with a Different (Twin Talk) Service

To create secondary accounts that use a different (twin talk) service:

1. In Customer Center, create a secondary account with the secondary (twin talk) service such as **/service/telco/gsm/twintelephony**.

   Note: You don't need to associate Sim Card and Number to the secondary account by using Customer Center.

2. Use the PCM_OP_CUST_SET_LOGIN opcode to associate the number with the secondary account:

```
0 PIN_FLD_POID                        POID [0] 0.0.0.1 /account XXXXX 0
0 PIN_FLD_END_
T                     TSTAMP [0] (1100093374) 10/11/2004 18:59:34:000 PM
0 PIN_FLD_OP_CORRELATION_
ID            STR [0] "2:CT1255:UnknownProgramName:0:AWT-
  EventQueue-0:59:1092287278:0:root.0.0.0.1::user1:123456789"
0 PIN_FLD_PROGRAM_NAME                 STR [0] "fm_num_pol_util.c"
0 PIN_FLD_SERVICE_OBJ                 POID [0] 0.0.0.1
  /service/telco/gsm/voicetwintalk YYYYYY 0
0 PIN_FLD_START_
T                   TSTAMP [0] (1100093374) 10/11/2004 18:59:34:000 PM
0 PIN_FLD_LOGINS                     ARRAY [0] allocated 1, used 1
1     PIN_FLD_ALIAS_LIST             ARRAY [1] allocated 1, used 1
2         PIN_FLD_NAME                 STR [0] "ZZZZZZ"
```

Where:

- XXXXX is the account POID of the secondary account.
- YYYYY is the service POID of the service associated with the secondary account.
- ZZZZZ is the alias (also called the login or number), such as **004912345678001, 004912345678002**, and so forth.

3. Repeat steps 1 and 2 until all secondary accounts are created.

Creating Secondary Accounts that Use the Same Service as the Primary Account

To create a secondary account that uses the same service as the primary account:

1. Create a number, for example, 004912345678001, with service TEL, by using Number Administration Center.
2. Create the secondary account with the service used by the primary account by using Customer Center.

Note:

- Use the number created in Step 1.
- You don't need to associate a SIM with the secondary account.
- The login ID of the secondary account is the A number plus a suffix. The A number is the A number of the primary account, and the suffix is one of the ERA values configured for the twin talk profile in the primary account. For example, if **0049100053** is the A number for the primary account and 001 is the value configured for OVERRIDE_ACCT ERA in the TWINTALK profile, then the login ID for the secondary account would be **0049100053001**.
- The secondary account can be any account depending on your iScript logic and the ERAs for the TWINTALK profile. If a customer has more than one secondary account, your iScript logic should determines which secondary account to select depending on the configured ERAs.

3. Repeat Steps 1 and 2 until all secondary accounts are created.

Twin Talk iScript Functions

This section describes the new Twin Talk Enabler iScript functions.

getServExtRating

Retrieves service-level ERAs.

This function returns the attribute-value pair array for a service ERA based on the identifying service, such as the mobile telephone number.

Syntax

```
BAS::String& getServExtRating( const BAS::String& key,
  const ::String& svcCode,
  const BAS::String& era,
  const BAS::DateTime& date);
```

Parameters

- *key* - The account login for which ERA is required, such as IMSI, MSISDN, ip address, login name, and so forth
- *svcCode* - The service code
- *era* - The ERA name
- *date* - The usage datestamp

Return Value

This function returns a string containing all the attributes of the requested ERA in attribute-value CSV format.

Example

The following list shows some examples of returned TWINTALK ERA name-value pairs:

- AM_ACCT,000
- PM_ACCT,001
- OVERRIDE_ACCT,002
- TOD_AM_BEGIN,12:00AM
- TOD_AM_END,8:00AM
- TOD_PM_BEGIN,5:00PM
- TOD_PM_END,5:00PM
- TOD_BUS_BEGIN,8:00AM
- TOD_BUS_END,5:00PM

getAcctExtRating

Retrieves account-level ERAs.

This function returns the attribute-value pair array for an account ERA based on the identifying service, such as the mobile telephone number.

Syntax

```
BAS::String& getAcctExtRating( const BAS::String& key,
  const BAS::String& svcCode,
  const BAS::String& era,
  const BAS::DateTime& date);
```

Parameters

- *key* - The account login for which ERA is required, such as IMSI, MSISDN, ip address, login name, and so forth
- *svcCode* - The service code
- *era* - The ERA name
- *date* - The usage datestamp

Return Value

This function returns a string containing all the attributes of the requested ERA in attribute-value CSV format.

Example

The following list shows some examples of returned TWINTALK ERA name-value pairs:

- AM_ACCT,000
- PM_ACCT,001
- OVERRIDE_ACCT,002
- TOD_AM_BEGIN,12:00AM
- TOD_AM_END,8:00AM
- TOD_PM_BEGIN,5:00PM
- TOD_PM_END,5:00PM
- TOD_BUS_BEGIN,8:00AM
- TOD_BUS_END,5:00PM

29

Sample Twin Talk Enabler Configuration Procedures

This chapter provides sample scenarios for setting up Oracle Communications Billing and Revenue Management (BRM) Twin Talk Enabler.

Before you read this document, you should read "About Twin Talk Enabler".

Overview of Sample Procedures

The sections below describe how to set up Twin Talk Enabler so that:

- Secondary accounts use the same service as the primary account. See "Sample 1: Configuring Twin Talk so that the Primary and Secondary Accounts Use the Same Service".
- Secondary accounts use a different service than the primary account. See:
  - Sample 2: Configuring Twin Talk so that Secondary Accounts Use Secondary Service /service/twintalk
  - Sample 3: Configuring Twin Talk so that Secondary Accounts Use Secondary Service /service/telco/gsm/twintelephony

Sample 1: Configuring Twin Talk so that the Primary and Secondary Accounts Use the Same Service

This section describes a sample procedure for setting up Twin Talk Enabler so that secondary accounts use the *same* service as their primary accounts.

Follow these procedures in the order given:

1. Configuring BRM
2. Creating Accounts
3. Activating Twin Talk in the Registry
4. Configuring the Twin Talk iScript
5. Testing Usage Rating

This sample uses the following accounts listed in Table 29–1:

Table 29–1 Sample 1 Accounts

Account	POID	Associated number	Description
Primary	**1000**	**004912345678**	The primary customer account. If no secondary account is selected from the ERAs, this account is charged.
AM	**10001**	**004912345678001**	The account to be charged for usage between 1:00 am and 11:00 am.
PM	**10002**	**004912345678002**	The account to be charged for usage between 1:00 pm and 11:00 pm.
OVERRIDE	**10003**	**004912345678003**	If the customer specifies an override (such as by typing two pound signs), this account will be charged regardless of the time of day settings defined in the AM and PM account ERAs.

Configuring BRM

Note: When you use the same service for primary and secondary accounts, you don't need to configure pipelines except for adding the twin talk iScript.

To configure BRM for twin talk when using the same service for primary and secondary accounts:

1. Open the ERA description file (*BRM_Home*/**sys/msgs/eradescr/era_descr.***locale*) and add the ERA names and descriptions. *BRM_Home* is the directory where you installed BRM components.

   The following example entry shows twin talk account- and service-level profiles in the **era_descr.en_US** file:

   - For account-level profiles:

```
STR
        ID = 30 ;
        VERSION = 1 ;
        STRING = "TWINTALK_ACCOUNT" ;
END
STR
        ID = 31 ;
        VERSION = 1 ;
        STRING = "To enable Twin Talk Account level provisioning tags." ;
END
```

   - For service-level profiles:

```
STR
        ID = 28 ;
        VERSION = 1 ;
        STRING = "TWINTALK_SERVICE" ;
END
STR
        ID = 29 ;
        VERSION = 1 ;
        STRING = "To enable Twin Talk Service level provisioning tags." ;
END
```

2. Save the file.
3. Load the ERA names and descriptions into the database by using the **load_localized_strings** utility.
4. Load the twin talk *service-level* ERA into the database by doing the following:
   a. Start Pricing Center and click **Launch - Provisioning Tags**.
   b. Add the TWINTALK_SERVICE era to a new or existing provisioning tag. For information, see Provisioning Tags Help.
   c. Select the description you entered in the **era_descr.en_US** file.
   d. Do not select the check box that indicates that provisioning is required.
5. Load the twin talk *account-level* ERA into the database by doing the following:
   a. Add the following line to the *BRM_Home***/sys/data/config/pin_telco_tags_gsm** file:

      ```
      account_era  "TWINTALK_ACCOUNT" 30 31
      ```

   b. Run the **load_pin_telco_tags** utility.

Creating Accounts

To create accounts when using the same service for primary and secondary accounts:

1. Create a primary account with the **/service/telco/gsm/telephony** service:
   a. Start Customer Center and choose **File - New - Consumer**.
   b. In the **Contact** tab, configure the contact information and click **Next**.
   c. In the **General** tab, choose **Euro** as the primary currency and click **Next**.
   d. In the **Plan** tab, select the **Standard GSM** plan and click **Next**.
   e. In the **Customize Services** tab, associate a SIM Card and Number with the service and click **Next**.
   f. Complete setting up the account in the rest of the tabs.
2. Start Number Administration Center and create the numbers **004912345678001**, **004912345678002**, and **004912345678003** with the TEL service.
3. Create the secondary account AM.
   a. Start Customer Center and choose **File - New - Consumer**.
   b. In the **Contact** tab, configure the contact information and click **Next**.
   c. In the **General** tab, choose **Euro** as the primary currency, and click **Next**.
   d. In the **Plan** tab, select the **Standard GSM** plan and click **Next**.
   e. In **Customize Services** tab, associate number **004912345678001** with the service and click **Next**.

      Note: SIM card association is not required.

   f. Complete setting up the account in the rest of the tabs.
4. Repeat step 3 for the PM and OVERRIDE accounts using the numbers **004912345678002** and **004912345678003**, respectively.

5. Add account- and service-level TWIN TALK ERAs for the primary account:
   a. In Customer Center, go to the **Promotions** tab of the primary account.
   b. In the Account-level Promotions section, add the following ERAs for TWINTALK_ACCOUNT:

```
- AM_ACCT            001
- TOD_AM_BEGIN       1:00 AM
- TOD_AM_END         11:00 AM
- PM_ACCT            002
- TOD_PM_BEGIN       1:00 PM
- TOD_PM_END         11:00 PM
- OVERRIDE_ACCT      003
```

   c. In the Service-level Promotions section, select ServiceID **GSM/Telephony**.
   d. Add the following ERA's for TWINTALK_SERVICE:

```
- AM_ACCT            001
- TOD_AM_BEGIN       1:00 AM
- TOD_AM_END         11:00 AM
- PM_ACCT            002
- TOD_PM_BEGIN       1:00 PM
- TOD_PM_END         11:00 PM
- OVERRIDE_ACCT      003
```

Activating Twin Talk in the Registry

Activate twin talk in the registry. See "Activating Twin Talk in the Pipeline Manager Registry".

Configuring the Twin Talk iScript

To configure the twin talk iScript when using the same service for primary and secondary accounts:

1. Open the Twin Talk Enabler iScript (*Pipeline_Home*/**iScriptLib/iScriptLib_Standard/ISC_TwinTalkEnabler.isc)** with a text editor such as vi. *Pipeline_Home* is the directory where you installed Pipeline Manager.
2. Change this line:

```
edrString(DETAIL.INTERN_SERVICE_CODE) = "TTK"
```

   to this:

```
edrString(DETAIL.INTERN_SERVICE_CODE) = "TEL"
```

3. Check and modify the following lines as required.

> **Note:** By default, the TEL service configuration uses DETAIL.A_NUMBER to retrieve profile information. For more information, see step 4 in "Defining the EDR Container".

   - Change this line:

```
edrString(DETAIL.INTERN_A_NUMBER_ZONE) = edrString( DETAIL.A_NUMBER) +
pmAccount;
```

to this:

```
edrString(DETAIL.A_NUMBER) = edrString( DETAIL.A_NUMBER) + pmAccount;
```

- Change this line:

```
edrString(DETAIL.INTERN_A_NUMBER_ZONE) = edrString( DETAIL.A_NUMBER) +
amAccount;
```

to this:

```
edrString(DETAIL.A_NUMBER) = edrString( DETAIL.A_NUMBER) + amAccount;
```

- Change this line:

```
edrString(DETAIL.INTERN_A_NUMBER_ZONE) = edrString( DETAIL.A_NUMBER) +
nameValue[1];
```

to this:

```
edrString(DETAIL.A_NUMBER) = edrString( DETAIL.A_NUMBER) + nameValue[1];
```

4. To use account-level profiles to select the secondary account, be sure that:

   - The following lines **are** commented out:

```
# String result = getServExtRating(edrString(DETAIL.A_NUMBER),
#                                  edrString(DETAIL.INTERN_SERVICE_CODE),
#                                  "TWINTALK",
#                                  edrDate(DETAIL.CHARGING_START_
TIMESTAMP));
```

   - The following lines are **not** commented out:

```
String result = getAcctExtRating(edrString(DETAIL.A_NUMBER),
                                  edrString(DETAIL.INTERN_SERVICE_CODE),
                                 "TWINTALK",
                                  edrDate(DETAIL.CHARGING_START_
TIMESTAMP));
```

5. To use service-level profiles to select the secondary account, be sure that:

   - The following lines **are** commented out:

```
# String result = getAcctExtRating(edrString(DETAIL.A_NUMBER),
#                                   edrString(DETAIL.INTERN_SERVICE_
CODE),
#                                   "TWINTALK",
#                                   edrDate(DETAIL.CHARGING_START_
TIMESTAMP));
```

   - The following lines are **not** commented out:

```
String result = getServExtRating(edrString(DETAIL.A_NUMBER),
                                 edrString(DETAIL.INTERN_SERVICE_CODE),
                                 "TWINTALK",
                                 edrDate(DETAIL.CHARGING_START_
TIMESTAMP));
```

Note: Both account- and service-level profiles can be accessed at the same time. You can customize your iScript logic to meet your requirements.

6. Stop and restart BRM and Pipeline Manager.

Testing Usage Rating

To test usage rating when using the same service for primary and secondary accounts:

1. Test usage rating for the primary account by sending this sample CDR to a pipeline:

   ```
   TEL; 004912345678;0049100090;20041119113000;3000;0;0;NORM;123456;
   ```

   Check that usage is charged to the primary account (account POID 10000) by using Customer Center.

2. Test usage rating for the AM account by sending this CDR to a pipeline:

   ```
   TEL; 004912345678;0049100090;20041119103000;3000;0;0;NORM;123456;
   ```

   Check that usage is charged to **AM_Account** (account POID 10001) by using Customer Center.

3. Test usage rating for the PM account by sending this CDR to a pipeline:

   ```
   TEL; 004912345678;0049100090;20041119133000;3000;0;0;NORM;123456;
   ```

   Check that usage is charged to **PM_Account** (account POID 10002) by using Customer Center.

4. Test usage rating for the OVERRIDE account by sending this CDR to a pipeline:

   ```
   TEL; 004912345678;0049100090##;20041119133000;3000;0;0;NORM;123456;
   ```

   Check that usage is charged to **OVERRIDE _Account** (account POID 10003) by using Customer Center.

Sample 2: Configuring Twin Talk so that Secondary Accounts Use Secondary Service /service/twintalk

This section describes a sample procedure for setting up Twin Talk Enabler so that secondary accounts use a *different* service (**/service/twintalk**) than that used by their primary accounts.

Follow these procedures in the order given:

1. Configuring BRM
2. Configuring Pipeline Manager
3. Creating Accounts
4. Activating Twin Talk in the Registry
5. Configuring the iScript
6. Testing Usage Rating

Notes:

- The service code for **/service/twintalk** is TTK.
- This sample uses the following accounts listed in Table 29–2:

Table 29–2 Sample 2 Accounts

Account	POID	Associated number	Description
Primary	**1000**	**004912345678**	The primary customer account. If no secondary account is selected from the ERAs, this account is charged.
AM	**10001**	**004912345678001**	The account to be charged for usage between 1:00 am and 11:00 am.
PM	**10002**	**004912345678002**	The account to be charged for usage between 1:00 pm and 11:00 pm.
OVERRIDE	**10003**	**004912345678003**	If the customer specifies an override (such as by typing two pound signs), this account will be charged regardless of the time of day settings defined in the AM and PM account ERAs.

Configuring BRM

To configure BRM for twin talk so that secondary accounts use the secondary service **/service/twintalk**:

1. Add the **/service/twintalk** storable class by using Developer Center. For information, see Developer Center Help.
2. Open the event map configuration file (the *BRM_Home*/**sys/data/pricing/example/pin_event_map**) and add the twin talk event map definition:

```
/service/twintalk
                : /event/session/telco/gsm            : Real Time Telco GSM S
ession
                : /event/delayed/session/telco/gsm    : Delayed Telco GSM Ses
sion
```

3. Save the file.
4. Load the event map into the database by using the **load_event_map** utility.
5. Open the custom bill item tags configuration file (*BRM_Home*/**sys/data/pricing/example/config_item_tags.xml**) and define a tag for the twin talk service:

```
<ItemTagElement>
      <ItemTag>twintalk</ItemTag>
      <EventType>/event/*</EventType>
      <ServiceType>/service/twintalk</ServiceType>
   </ItemTagElement>
```

6. Save the file.
7. Load the custom item tags into the database by using the **load_config_item_tags** utility.
8. Open the custom bill item types configuration file (*BRM_Home*/**sys/data/pricing/example/config_item_types.xml**) and define a bill item type for the twin talk item:

9. Save the file.
10. Load the custom bill item types into the database by using the **load_config_item_types** utility.
11. Open the ERA description file (*BRM_Home*/**sys/msgs/eradescr/era_descr.***locale*) and add the ERA names and descriptions:

    The following example entry shows twin talk profiles in the **era_descr.en_US** file.

    - For account-level profiles:

```
STR
        ID = 30 ;
        VERSION = 1 ;
        STRING = "TWINTALK_ACCOUNT" ;
END
STR
        ID = 31 ;
        VERSION = 1 ;
        STRING = "To enable Twin Talk Account provisioning tags." ;
END
```

    - For service-level profiles:

```
STR
        ID = 28 ;
        VERSION = 1 ;
        STRING = "TWINTALK_SERVICE" ;
END
STR
        ID = 29 ;
        VERSION = 1 ;
        STRING = "To enable Twin Talk Service level provisioning tags.";
END
```

12. Save the file.
13. Load the ERA names and descriptions into the database by using the **load_localized_strings** utility.
14. Load the twin talk *service-level* ERA into the database by doing the following:
    a. Start Pricing Center and click **Launch - Provisioning Tags**.
    b. Add the TWINTALK_SERVICE era to a new or existing provisioning tag. For information, see Provisioning Tags Help.
    c. Select the description you entered in the **era_descr.en_US** file.
    d. Do not select the check box that indicates that provisioning is required.
15. Load the twin talk account-level ERA into the database by doing the following:
    a. Add the following line to the *BRM_Home*/**sys/data/config/pin_telco_tags_gsm** file:

```
account_era   "TWINTALK_ACCOUNT" 30 31
```

    b. Run the **load_pin_telco_tags** utility.

Configuring Pipeline Manager

To configure Pipeline Manager when secondary accounts use the secondary service **/service/twintalk**:

1. Follow the steps in "Activating Twin Talk in the Pipeline Manager Registry".
2. To direct **/service/telco/gsm/twintalk** usage to a separate output file:
   a. Add the following text to the registry file (*Pipeline_Home***/conf/wireless.reg**):

```
          #------------------------------------------------------------------
----
          # The /service/gsm/twintalk output stream
          #------------------------------------------------------------------
----
          TTKOutput
          {
            ModuleName = OUT_GenericStream

            ProcessType = RATING_PIPELINE
            EventType = /event/delayed/session/telco/gsm

            Module
            {
              Grammar = ./formatDesc/Formats/Solution42/V670_EVENT_LOADER_
OutGrammar.dsc

              DeleteEmptyStream = True # defaults to TRUE

              OutputStream
              {
                ModuleName = EXT_OutFileManager
                Module
                {
                  OutputPath = ./data/out/gsm/TTK
                  OutputPrefix = test_TTK
                  OutputSuffix = .out
                  TempPrefix   = .

                  TempDataPath    = ./data/out/gsm/TTK
                  TempDataPrefix  = ttk.tmp.
                  TempDataSuffix  = .data

                  Replace = TRUE
                }
              }
            }
          } # end of TTKOutput
```

> **Important:** To ensure output file integrity, specify a unique combination of OutputPath, OutputSuffix, and OutputPrefix values for each output stream defined in the registry.

   b. Add the following text to the *Pipeline_Home*/**iScriptLib/iScriptLib_StandardIRL_EventTypeSplitting.dat**a file just before the entry **.*;TELOutput**:

```
TTK;TTKOutput
```

3. Save the file.
4. Define the twin talk service:

   a. In Pricing Center, choose **View - Pipeline Setup Toolbox - Product and Service.**

   b. Add a record with service code **TTK** and PIN Service Type **/service/twintalk**.

5. Map the twin talk service to a usage event:

   a. In Pricing Center, choose **View - Pipeline Setup Toolbox - Product and Service - Reference mapping**.

   b. Add a record with Reference Object **/service/twintalk** and Reference Parameter **/event/delayed/session/telco/gsm**.

6. Define the EDR container:

   a. In Pricing Center, choose **Pipeline Setup Tools - EDR - EDR Container Description - ALL_RATE - Alias Mapping**.

   b. Add a record with Reference **UniData_CustA,** Key **TTK**, FieldID **DETAIL.INTERN_A_NUMBER_ZONE**.

7. Set up a rate plan for the twin talk service:

   a. In Pricing Center, choose **View - Pipeline Toolbox - Rate Plan**.

   b. Create a rate plan with the code **TwinTalk**.

8. Configure the price list and create the plan **TTKPlan**. For more information, see "Configuring Twin Talk Pricing".

Creating Accounts

To create accounts when secondary accounts use the secondary service **/service/twintalk**:

1. Create a primary account with the **/service/telco/gsm/telephony** service.

   a. Start Customer Center and choose **File - New - Consumer**.

   b. In the **Contact** tab, configure the contact information and click **Next**.

   c. In the **General** tab, choose **Euro** as the primary currency and click **Next**.

   d. In the **Plan** tab, select the **Standard GSM** plan and click **Next**.

   e. In the **Customize Services** tab, associate a SIM Card and Number with the service and click **Next**.

   f. Complete setting up the account in the rest of the tabs.

2. Create the secondary AM account with the **/service/twintalk** service.

   a. Start Customer Center and choose **File - New - Consumer**.

   b. In the **Contact** tab, configure contact information and click **Next**.

   c. In the **Customize Services** tab, enter the login **004912345678001** and the password and click **Next**. (The login is the primary account number plus the suffix **001** for the AM account.)

   d. Complete setting up the account in the rest of the tabs.

3. Repeat step 2 for the PM and OVERRIDE accounts using login values **004912345678002** and **004912345678003**, respectively.

4. Add account- and service-level TWIN TALK ERAs for the primary account:

   a. In Customer Center, go to the **Promotions** tab of the primary account.

b. In the Account-level Promotions section, add the following ERAs for the TWINTALK_ACCOUNT account.

```
 - AM_ACCT            001
- TOD_AM_BEGIN        1:00 AM
- TOD_AM_END          11:00 AM
- PM_ACCT             002
- TOD_PM_BEGIN        1:00 PM
- TOD_PM_END          11:00 PM
- OVERRIDE_ACCT       003
```

c. In the Service-level Promotions section, select service **ID GSM/Telephony**.

d. Add the following ERAs for TWINTALK_SERVICE:

```
- AM_ACCT             001
- TOD_AM_BEGIN        1:00 AM
- TOD_AM_END          11:00 AM
- PM_ACCT             002
- TOD_PM_BEGIN        1:00 PM
- TOD_PM_END          11:00 PM
- OVERRIDE_ACCT       003
```

Activating Twin Talk in the Registry

Activate twin talk in the registry. See "Activating Twin Talk in the Pipeline Manager Registry".

Configuring the iScript

To configure the iScript when secondary accounts use the secondary service **/service/twintalk**:

1. Open the *Pipeline_Home*/**iScriptLib/iScriptLib_Standard/ISC_TwinTalkEnabler.isc** iScript file.
2. Be sure that **edrString(DETAIL.INTERN_SERVICE_CODE)** is assigned with the TTK service code.
3. To use account-level profiles to select the secondary account, be sure that:
   - The following lines are **not** commented out:

```
String result = getAcctExtRating(edrString(DETAIL.A_NUMBER),
                                  edrString(DETAIL.INTERN_SERVICE_CODE),
                                  "TWINTALK",
                                  edrDate(DETAIL.CHARGING_START_
TIMESTAMP));
```

   - The following lines **are** commented out:

```
#  String result = getServExtRating(edrString(DETAIL.A_NUMBER),
#                                   edrString(DETAIL.INTERN_SERVICE_CODE),
#                                   "TWINTALK",
#                                   edrDate(DETAIL.CHARGING_START_
TIMESTAMP));
```

4. To use service-level profiles to select the secondary account, be sure that:
   - The following lines **are** commented out:

```
# String result = getAcctExtRating(edrString(DETAIL.A_NUMBER),
```

```
#                                        edrString(DETAIL.INTERN_SERVICE_
CODE),
#                                       "TWINTALK",
#                                        edrDate(DETAIL.CHARGING_START_
TIMESTAMP));
```

- The following lines are **not** commented out:

```
String result = getServExtRating(edrString(DETAIL.A_NUMBER),
                                 edrString(DETAIL.INTERN_SERVICE_CODE),
                                 "TWINTALK",
                                 edrDate(DETAIL.CHARGING_START_
TIMESTAMP));
```

Note: Both account- and service-level profiles can be accessed at the same time. Customize your iScript logic according to your requirements.

5. Stop and restart BRM and Pipeline Manager.

Testing Usage Rating

To test usage rating when secondary accounts use the secondary service **/service/twintalk**:

1. Test usage rating for the primary account by sending this sample CDR to a pipeline:

   ```
   TEL; 004912345678;0049100090;20041119113000;3000;0;0;NORM;123456;
   ```

   Check that usage is charged to the primary account (account POID 10000) by using Customer Center.

2. Test usage rating for the AM account by sending this sample CDR to a pipeline:

   ```
   TEL; 004912345678;0049100090;20041119103000;3000;0;0;NORM;123456;
   ```

   Check that usage is charged to **AM_Account** (account POID 10001) by using Customer Center.

3. Test usage rating for the PM account by sending this sample CDR to a pipeline:

   ```
   TEL; 004912345678;0049100090;20041119133000;3000;0;0;NORM;123456;
   ```

   Check that usage is charged to **PM_Account** (account POID 10002) by using Customer Center.

4. Test usage rating for the OVERRIDE account by sending this sample CDR to a pipeline:

   ```
   TEL; 004912345678;0049100090##;20041119133000;3000;0;0;NORM;123456;
   ```

   Check that usage is charged to **OVERRIDE _Account** (account POID 10003) by using Customer Center.

Sample 3: Configuring Twin Talk so that Secondary Accounts Use Secondary Service /service/telco/gsm/twintelephony

This section contains example steps for setting up twin talk using the **/service/telco/gsm/twintelephony** service with secondary accounts. In this scenario, the CSR doesn't use the login screen in Customer Center. Instead, you use the PCM_OP_CUST_SET_LOGIN opcode to associate the login with the secondary account.

Follow these procedures in the order given:

1. Configuring BRM
2. Configuring Pipeline Manager
3. Creating Accounts
4. Activating Twin Talk in the Registry
5. Configuring the iScript
6. Testing Usage Rating

Notes:

- The service code for **/service/telco/gsm/twintelephony** is TTKTEL.
- This sample uses the following accounts listed in Table 29–3:

Table 29–3 Sample 3 Accounts

Account	POID	Associated number	Description
Primary	**1000**	**004912345678**	The primary customer account. If no secondary account is selected from the configured ERAs, this account is charged.
AM	**10001**	**004912345678001**	The account to be charged for usage between 1:00 am and 11:00 am.
PM	**10002**	**004912345678002**	The account to be charged for usage between 1:00 pm and 11:00 pm.
OVERRIDE	**10003**	**004912345678003**	If the customer specifies an override (such as by typing two pound signs), this account will be charged regardless of the time of day settings defined in the AM and PM account ERAs.

Configuring BRM

To configure BRM for twin talk so that secondary accounts use the secondary service **/service/telco/gsm/twintelephony**:

1. Add the classes **/service/telco/gsm/twintelephony** and **/config/telco/gsm/twintelephony** by using Developer Center. For information, see Developer Center Help.
2. Follow steps 2 through 14 of Configuring BRM in Sample 2: Configuring Twin Talk so that Secondary Accounts Use Secondary Service /service/twintalk, but replace all **/service/twintalk** specifications with **/service/telco/gsm/twintelephony** and replace all **TTK** specifications with **TTKTEL**.
3. Create the provisioning tag for the secondary service by doing the following:
   a. Start Pricing Center and click **Launch - Provisioning Tags**.

b. Click **New**, and enter a name for the provisioning tag.

c. In the **Service** field, select **/service/telco/gsm/twintelephony**.

d. Enter a description.

e. Select the **Deprovision when product is cancelled** check box.

f. On the **Features** tab, add a service extension with the code **PIN_FLD_BEARER_SERVICE** and the value **T11**.

g. Add these features to the **Features To Use** list: **CLIP**, **CW**, **VMBOX**.

h. On the **Extended Rating Attributes** tab, add a service ERA with the service code **TWINTALK_SERVICE**, and then select a name and a description for the ERA.

   Do not select the check box that indicates that provisioning is required

Configuring Pipeline Manager

To configure Pipeline Manager when secondary accounts use the **/service/telco/gsm/twintelephony** service, follow all steps in "Configuring Pipeline Manager" in "Sample 2: Configuring Twin Talk so that Secondary Accounts Use Secondary Service /service/twintalk", but replace all **/service/twintalk** specifications with **/service/telco/gsm/twintelephony** and replace all **TTK** specifications with **TTKTEL**.

Creating Accounts

To create accounts when secondary accounts use the secondary service **/service/telco/gsm/twintelephony**:

1. Create a primary account with the **/service/telco/gsm/telephony** service.

   a. Start Customer Center and choose **File - New - Consumer**.

   b. In the **Contact** tab, configure the contact information and click **Next**.

   c. In the **General** tab, choose **Euro** as the primary currency and click **Next**.

   d. In the **Plan** tab, select the **Standard GSM** plan and click Next.

   e. In the **Customize Services** tab, associate a SIM Card and Number with the service and click **Next**.

   f. Complete setting up the account in the rest of the tabs.

2. Create the secondary AM account with the **/service/telco/gsm/twintelephony** service.

   a. Start Customer Center and choose **File - New - Consumer**.

   b. In the **Contact** tab, configure the contact information and click **Next**.

   c. In the **Customize Services** tab, enter the login **004912345678001** and the password and click **Next**. (The login is the primary account number plus the suffix **001** for the AM account.)

   d. Complete setting up the account in the rest of the tabs.

   e. Start Developer Center and find the service POID of the account.

   f. Execute PCM_OP_CUST_SET_LOGIN to create the alias 004912345678001 for the secondary account. For more information, see "Creating Secondary Accounts with a Different (Twin Talk) Service".

3. Repeat step 2 for the PM and OVERRIDE accounts using the login values **004912345678002** and **004912345678003**, respectively.
4. Perform step 4 in "Creating Accounts".

Activating Twin Talk in the Registry

Activate twin talk in the registry. See "Activating Twin Talk in the Pipeline Manager Registry".

Configuring the iScript

Configure the iScript as described in "Configuring the iScript" in "Sample 2: Configuring Twin Talk so that Secondary Accounts Use Secondary Service /service/twintalk".

Testing Usage Rating

Test usage rating as described in "Testing Usage Rating" in "Sample 2: Configuring Twin Talk so that Secondary Accounts Use Secondary Service /service/twintalk".

Part V

Loading Rated Events

Part V describes how to load rated events in an Oracle Communications Billing and Revenue Management (BRM) system. It contains the following chapters:

- Understanding Rated Event Loader
- Installing Rated Event Loader
- Configuring Rated Event Loader
- Loading Prerated Events

30

Understanding Rated Event Loader

This chapter describes Oracle Communications Billing and Revenue Management (BRM) Rated Event (RE) Loader and how it imports pipeline-rated events into the BRM database.

You should be familiar with the following topics:

- BRM concepts and architecture. See "Introducing BRM" and "BRM System Architecture" in *BRM Concepts*.
- Using Pipeline Manager to rate events. See "About Pipeline Rating".
- Oracle In-Memory Database (IMDB) Cache. See "Using Oracle IMDB Cache Manager" in *BRM System Administrator's Guide*.
- Rerating concepts and using Pipeline Manager to rerate events. See "About Rerating Pipeline-Rated Events" in *BRM Setting Up Pricing and Rating*.

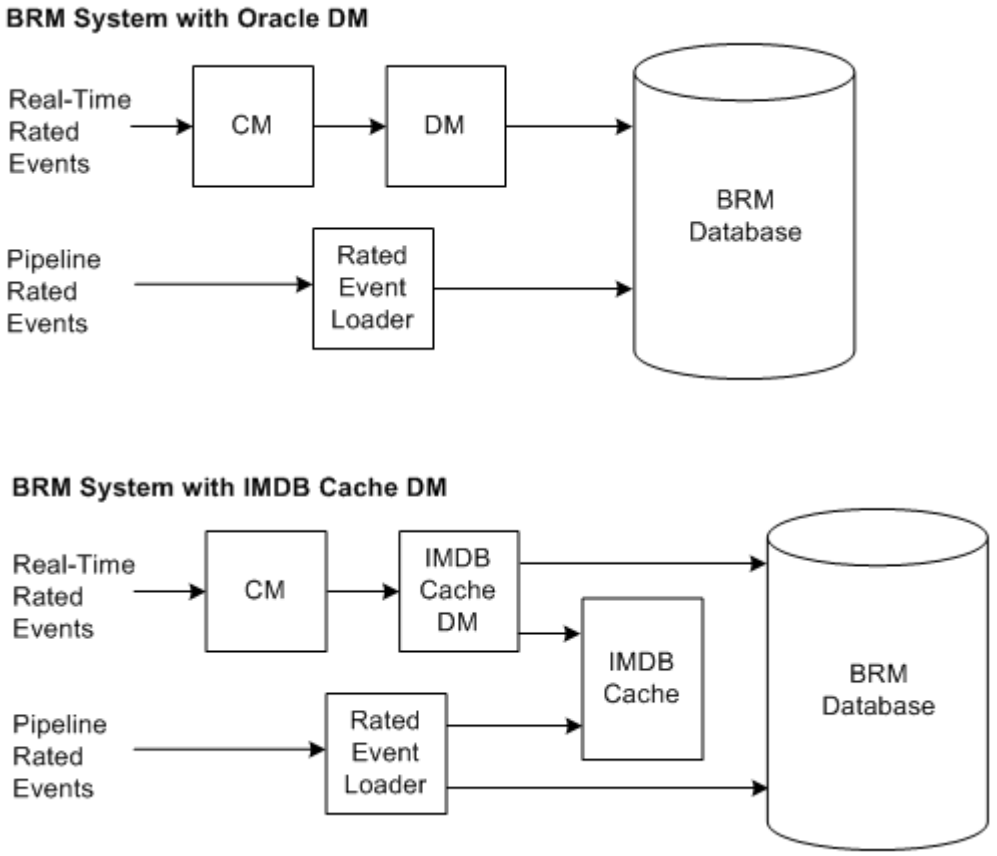
Important: RE Loader is an optional feature that requires a separate license.

About RE Loader

RE Loader is an optional BRM application that loads pipeline-rated events directly into the BRM database, bypassing the Connection Manager (CM) and Data Manager (DM). RE Loader then updates account balances, billing items, and journals in the BRM database (in Oracle DM systems) or in Oracle IMDB Cache (in IMDB Cache DM systems). After the events are loaded, you can run BRM applications such as billing and reports on the rated data.

Figure 30–1 compares the data flow of real-time events to the data flow of pipeline-rated events:

Figure 30–1 Real-Time vs Pipeline-Rated Event Data Flow



About the Database Schema

RE Loader uses a partitioned database and inserts prerated events into separate partitions.

Important: You *must* partition your database when loading prerated events. For more information, see "Partitioning Database Tables" in *BRM System Administrator's Guide*.

Because prerated events are loaded into their own partitions, there is minimal impact on real-time system performance.

About RE Loader Event Types

By default, RE Loader loads the wireless services and corresponding BRM event types shown in Table 30–1.

However, you can configure RE Loader to load custom events. For information, see "Configuring the RE Loader Infranet.properties File".

Table 30–1 Service and Event Types Loaded by Default

Service Type	Event Type
GPRS (General Packet Radio Service)	**/event/delayed/session/gprs**
GSM (Global System for Mobile Communication)	**/event/delayed/session/telco/gsm**

Note: Prerated event storable class types must start with **/event/delayed/** so that BRM can distinguish them from real-time events.

These events are loaded into separate BRM database partitions allocated for delayed events. The event types are called "delayed" because they are rated before they are loaded, and there is a delay between the two actions. This is unlike events loaded in real time or by Universal Event (UE) Loader. Because prerated events are loaded into their own tables, there is minimal impact on real-time system performance.

Important: You should not load the same event types by using RE Loader and another method such as an optional manager or UE Loader.

About Loading Prerated Events

Prerated events are events that have been rated by Pipeline Manager prior to being loaded into the BRM database. Basic steps of pipeline rating and event loading include:

1. Pipeline Manager rates events associated with call detail records (CDRs).

   For information on how Pipeline Manager prerates events, see "How Events Are Rated by Using Pipeline Manager".

2. In IMDB Cache DM systems, Pipeline Manager splits events based on their target logical partition.

   You configure Pipeline Manager to split files based on logical partitions by using the AccountLPRouter module. For information, see "Setting Up Pipeline Manager for IMDB Cache".

3. Pipeline Manager creates an output file for each service type and places them in one or more output directories.

   You configure the number and location of your pipeline output directories by using the pipeline EXT_OutFileManager module.

4. RE Loader loads the prerated events.

   For information, see "RE Loader Process Overview".

About Loading Rerated Events

It is possible to discover pricing or rating configuration errors after events have been rated by Pipeline Manager and loaded into the BRM database. When this occurs, you rerate any incorrectly rated events and reload them into the BRM database.

When you need to rerate and reload pipeline-rated events, you must:

1. Extract events that need rerating from the BRM database by using the Event Extraction Tool.

   Note: Event Extraction Manager is included in the RE Loader installation.

2. Rerate those events by using Pipeline Manager. Pipeline Manager backs out the previous rating changes and then rerates the events.
3. Reload the rerated events by using RE Loader.

   For information, see "RE Loader Process Overview".

For an overview of the rerating process, see "About Rerating Pipeline-Rated Events" in *BRM Setting Up Pricing and Rating*.

RE Loader Process Overview

RE Loader processes output files generated from Pipeline Manager. You send these files to RE Loader manually through a command-line utility or automatically by the Batch Controller.

After RE Loader receives a pipeline output file, it:

1. Checks the event header to determine the storable class type and whether the file contains prerated, rerated, or discount events.
2. Parses the event data record (EDR) data into multiple temporary files, one for each BRM database table to be loaded, and places the files in a temporary directory.
3. Loads events from each temporary file into the BRM database by using multiple Oracle SQL Loader utility **sqlldr** processes.
4. Calls stored procedures to update the account balances, bill items, and journals.
   - In BRM systems with Oracle DM, the stored procedures update account balances, billing items, and journals *in the BRM database*.
   - In BRM systems with IMDB Cache DM, the stored procedures update account balances, billing items, and journals *in Oracle IMDB Cache*.
5. Logs the session information in its log file (**rel.pinlog**).

About Running RE Loader

You can run RE Loader in one of the following ways:

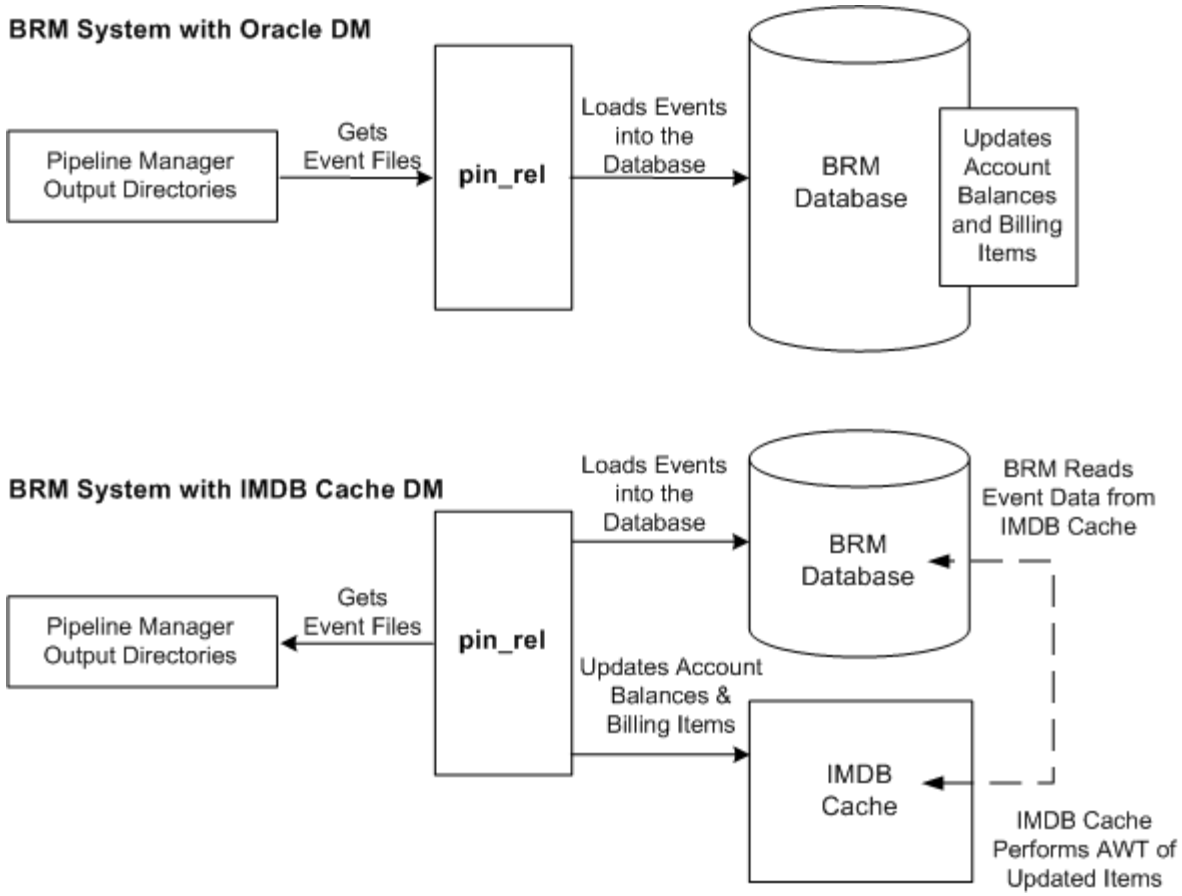
- Manually from a command line. See "About Running RE Loader Manually".
- Automatically by using Batch Controller and the RE Loader batch handler. See "About Running RE Loader Automatically".
- As a daemon. See "About Running the RE Loader Daemon".

About Running RE Loader Manually

When you run RE Loader manually from a command line, you specify the location of the pipeline output file in the command line.

Figure 30–2 shows the RE Loader work flow when you run it manually from a command line:

Figure 30–2 Work Flow of Manual Execution of RE Loader



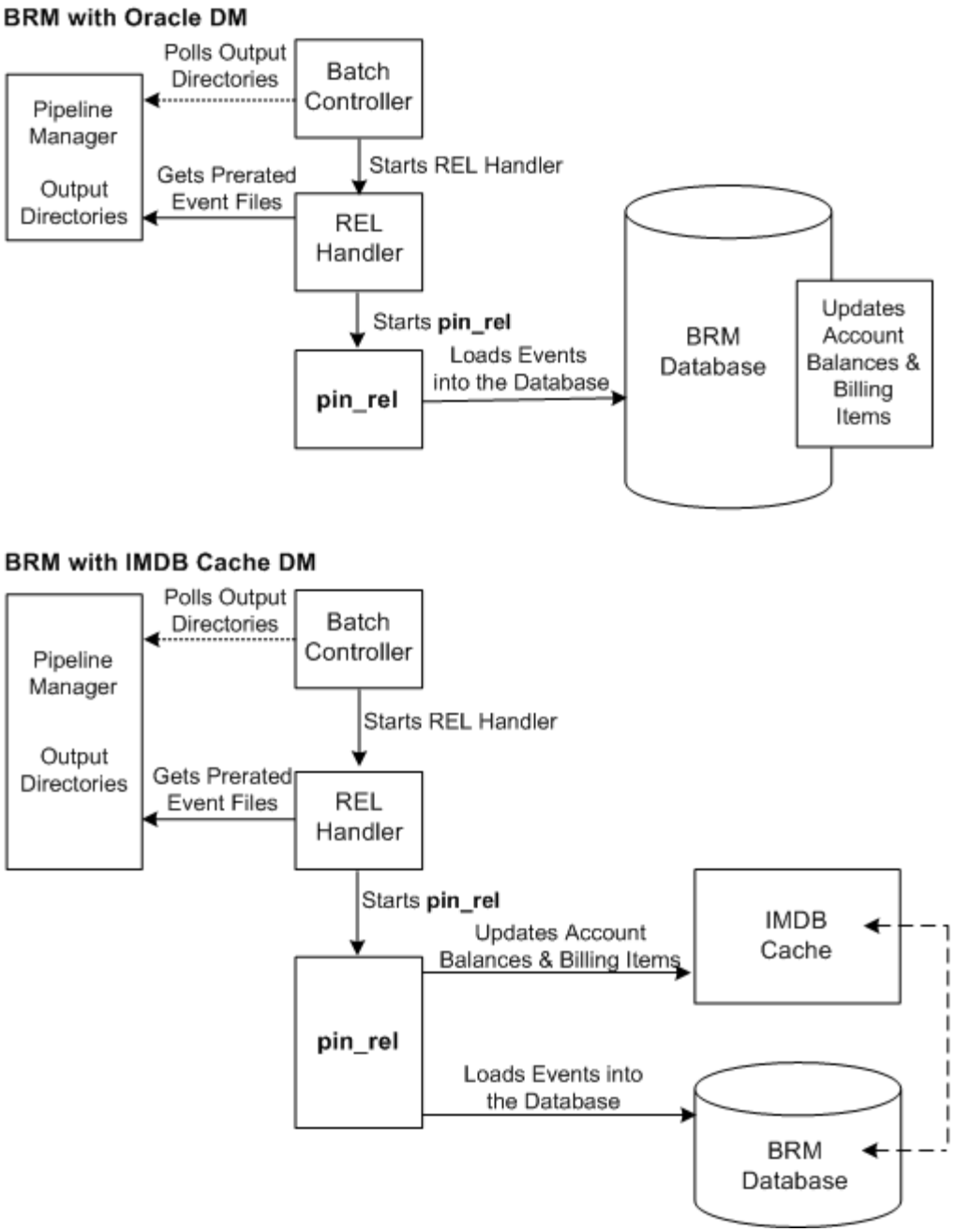
About Running RE Loader Automatically

To schedule RE Loader to run automatically, you must set up the following components:

- **Batch Controller**, which detects when an EDR file is present in the pipeline output directory and starts the RE Loader batch handler.
- **RE Loader batch handler**, which moves the EDR file from the pipeline output directory to the RE Loader processing directory and starts the RE Loader utility (**pin_rel**).

Figure 30–3 shows the RE Loader work flow when you schedule loading of events:

Figure 30–3 Work Flow of Scheduled Execution of RE Loader



The following actions are performed when RE Loader is scheduled to run automatically:

1. Batch Controller detects an EDR file in a pipeline output directory and starts the RE Loader batch handler.
2. The RE Loader batch handler moves the EDR file from the pipeline output directory to the RE Loader processing directory and starts the RE Loader utility (**pin_rel**).
3. RE Loader processes the file, loads the events into the BRM database, and updates the account balances, billing items, and journals. For more information, see "RE Loader Process Overview".

4. RE Loader batch handler moves the original file to an archive directory if the records are successfully loaded or to a reject directory if the records are not successfully loaded.

> **Tip:** The RE Loader batch handler loads one file at a time. You can load more than one file at a time by configuring Batch Controller to call several RE Loader batch handler processes. For more information, see "About Running Multiple RE Loader Processes".

About Running the RE Loader Daemon

When you run the RE Loader daemon, the daemon creates RE Loader threads to load EDR files. Each RE Loader thread loads one EDR file. The number of threads that the RE Loader daemon creates depends on the maximum number of threads that you configure to run simultaneously at a particular time. Because you can configure multiple threads that can run at a time, you can load more than one EDR file simultaneously. This helps in increasing loading performance.

For example, if you configure 10 threads to run simultaneously at peak time, the RE Loader daemon creates 10 threads, which means that 10 EDR files are loaded at the same time.

> **Important:** Running the RE Loader daemon is the preferable method if most of the EDR files that you want to load are of small size.

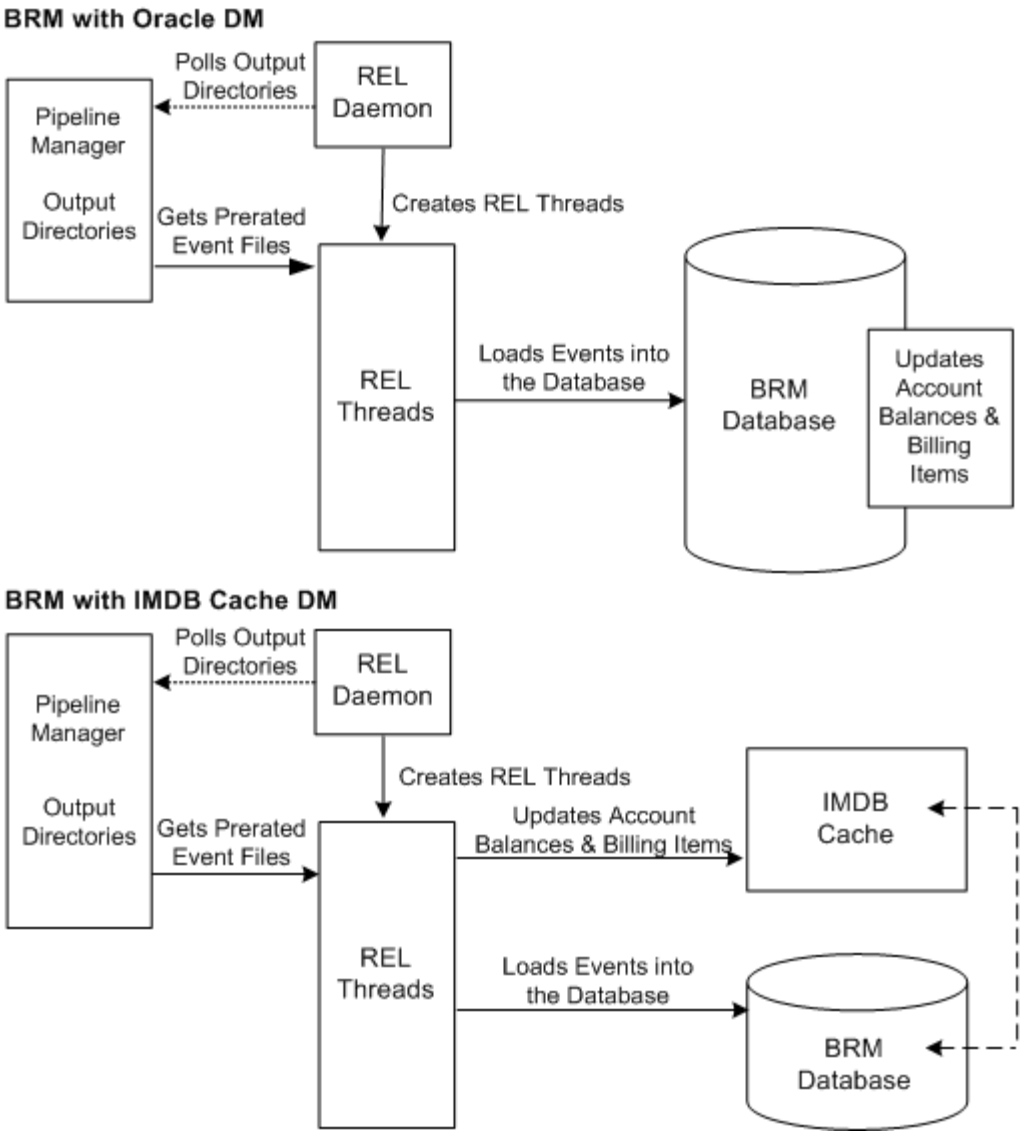
To run the RE Loader daemon, you must set up the following components:

- The RE Loader **Infranet.properties** file, which contains the entries for running the RE Loader daemon. See Table 32–5, " RE Loader Daemon Entries" for more information.
- The RE Loader daemon start and stop scripts (**start_rel_daemon** and **stop_rel_daemon**). See "Running the RE Loader Daemon" for more information.

You can also run the RE Loader daemon by using the **pin_ctl** utility. See "Starting a Component by Using the pin_ctl Utility" in *BRM System Administrator's Guide*.

Figure 30–4 shows the RE Loader work flow when you run the RE Loader daemon:

Figure 30–4 Work Flow of RE Loader Daemon



The following actions are performed when you run the RE Loader daemon:

1. The RE Loader daemon detects an EDR file in a pipeline output directory.
2. The RE Loader daemon creates RE Loader threads up to the maximum number of threads configured.
3. The RE Loader threads process the files, load the events into the BRM database, and update the account balances, billing items, and journals.
4. The RE Loader threads move the original file to an archive directory if the records are successfully loaded or to a reject directory if the records are not successfully loaded. These directories are configured in the RE Loader **Infranet.properties** file.

About Running Multiple RE Loader Processes

To achieve better loading performance, the sample Batch Controller configuration file (**SampleBatchControllerInfranet.properties**) is set to run three RE Loader processes in parallel. This means that when you schedule RE Loader to run automatically, Batch

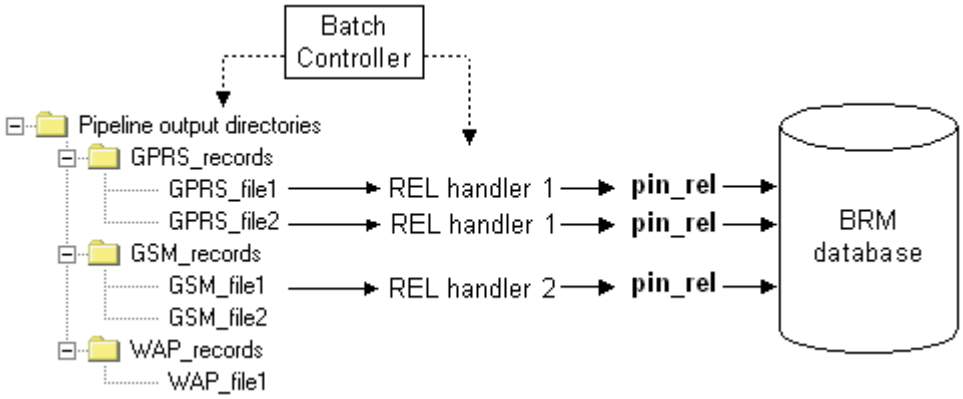
Controller starts up to three instances of the RE Loader batch handler. Each instance of the RE Loader batch handler starts an instance of the RE Loader utility.

If you configure RE Loader to run manually, you can start multiple processes from the command line.

Important: If you run multiple RE Loader processes in parallel, configure Pipeline Manager to delete empty output streams. For more information, see "Configuring Pipeline Manager to Delete Empty Output Streams".

Figure 30–5 shows an example of three RE Loader batch handler and **pin_rel** utility processes running to load EDR files. Each utility loads one file into the BRM database:

Figure 30–5 Example of Three Batch Handlers and pin_rel Utilities



The following actions are performed when multiple RE Loader processes are configured:

1. Batch Controller starts an RE Loader batch handler for each new file it detects in the pipeline output directory, up to the maximum number of RE Loader batch handler processes configured.
2. Each RE Loader batch handler process starts an RE Loader utility process.
3. The RE Loader utility processes the events in the file in three phases:
   - Preprocessing
   - Loading
   - Account balance, bill item, and journal updates

   During the loading phase, the database tables are locked so that only one RE Loader process can load events at one time.
4. Each RE Loader process polls the other processes to see whether they are currently in the loading phase and waits its turn to load events. When the first process is loading, the second process performs prepreprocessing tasks while waiting its turn. When the first process finishes loading, the second process loads while the first process performs account balance, bill item, and journal updates for the events it loaded:

Process 1	Preprocessing	Loading	Updating	Preprocessing	. . .
Process 2	Not started	Preprocessing	Loading	Updating	. . .
Process 3	Not started	Preprocessing	Waiting	Loading	. . .

5. When an RE Loader process has completed all three phases, it is free to process another file.

The files are loaded in sequence, one directory at a time. If the number of RE Loader processes is set to 3, only three handler processes can run at one time for all directories.

Setting the Optimal Number of RE Loader Processes

Because there are three phases to processing EDR files, the optimal number of RE Loader processes to configure is at least three. If you want to configure more than three processes, you should test your RE Loader performance. Because only one RE Loader process can load events at a time, having more than three means those that have finished the preprocessing phase wait their turn to load events. Depending on the size of your EDR files and the time it takes to load events, configuring four or five processes might save time.

You configure the number of RE Loader processes to run in parallel by setting the number of RE Loader batch handlers to start in the Batch Controller configuration file. For more information, see "Configuring Batch Controller".

Configuring Pipeline Manager to Delete Empty Output Streams

If you run multiple RE Loader processes, you should configure Pipeline Manager to delete empty output streams.

Pipeline Manager generates files based on the number of output streams that are running. If some of the output streams are empty, the pipeline can produce empty files, which causes an error when RE Loader attempts to load the empty files.

To produce only one output stream, set the **DeleteEmptyStream** pipeline registry entry to **True**. This is the default. For more information, see "Configuring Output for Rated Events and AAA Responses" and "OUT_GenericStream".

About Backing Up RE Loader Files

By default, RE Loader skips redo generation when loading files into the BRM database. This optimizes loading performance, but it can cause you to lose data if your system shuts down ungracefully.

To prevent data loss when your system shuts down:

- Make full backups of the BRM database on a regular basis.
- Archive all successfully loaded files until you make a full database backup.

You can re-enable redo generation, at the cost of loading performance, by modifying the RE Loader control files. For information, see "Configuring Whether to Perform Redo Generation".

About Handling Errors

If any errors occur during event loading, all events loaded in that session are deleted from the database. After events have been successfully loaded, if any errors occur

during the update procedure, you can correct the errors and then update the relevant events by rerunning the RE Loader utility. The utility detects that the events loaded correctly and performs only the update procedure. For more information, see "Troubleshooting Event Loading".

About Using RE Loader in a Multischema System

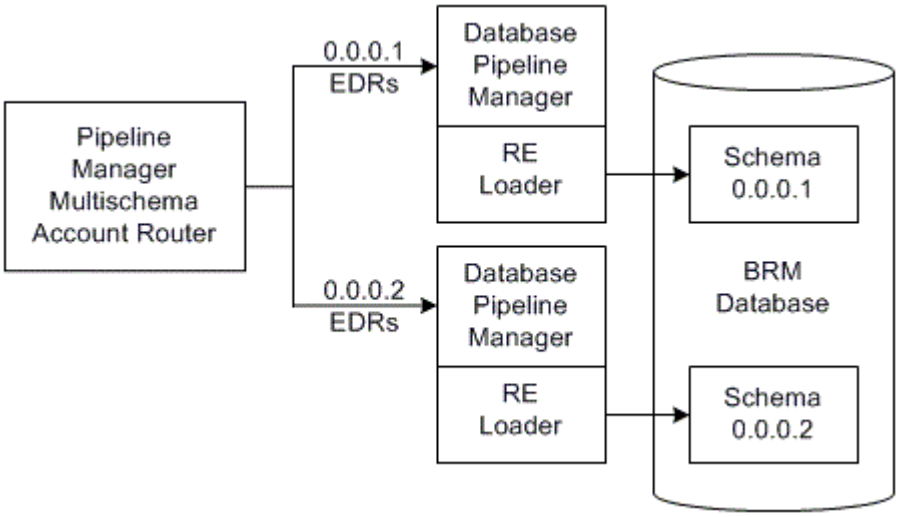
If you use a multischema system, you must set up the following for each BRM database schema in your system:

- Pipeline Manager instance. Each instance of Pipeline Manager must also have its own set of output files.
- RE Loader instance. Each instance of RE Loader must also have its own set of RE Loader processing directories.
- (Running RE Loader automatically only) An instance of Batch Controller and the RE Loader batch handler.

You must also install and configure a separate instance of Pipeline Manager. This Pipeline Manager multischema account router routes EDRs to the appropriate schema Pipeline Manager instance based on the account's database number.

Figure 30–6 shows the flow of data in a multischema system:

Figure 30–6 Multischema System Data Flow



About Using RE Loader in an Oracle IMDB Cache System

If you use a BRM system with IMDB Cache DM, you must set up the following for each logical partition in your system:

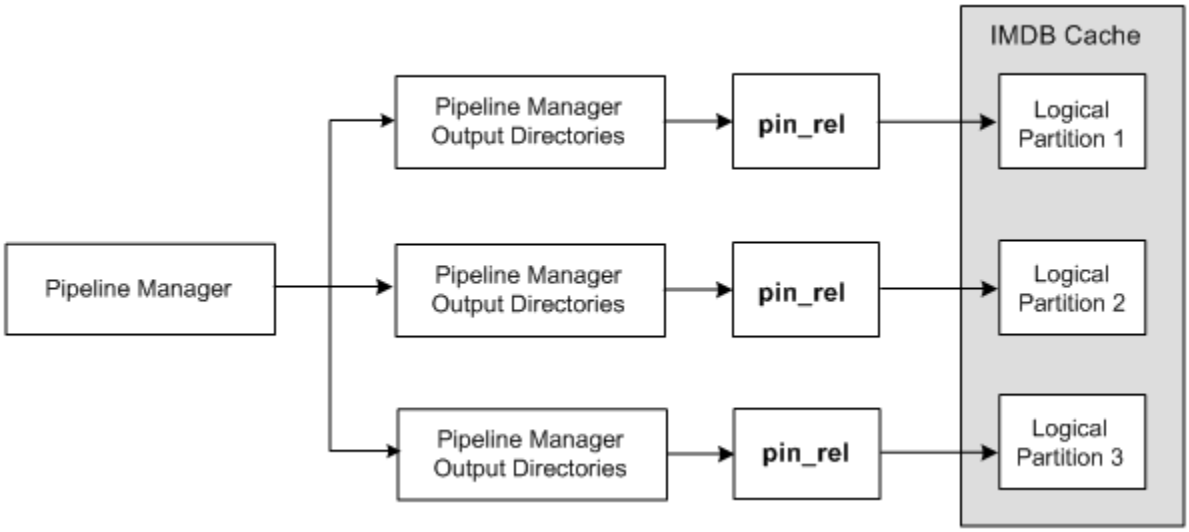
- Pipeline Manager output directories. You must have a separate set of output directories for each logical partition.
- RE Loader instance. Each instance of RE Loader must also have its own set of RE Loader processing directories.
- (Running RE Loader automatically only) An instance of Batch Controller and the RE Loader batch handler.

- (Running the RE Loader daemon only) The **start_rel_daemon** script and the *event*.**tt_node** entry in the RE Loader **Infranet.properties** file, where *event* specifies the event type.

You must also configure Pipeline Manager to split EDRs into separate output streams based on the account's logical partition number.

Figure 30–7 shows the flow of data in an Oracle IMDB Cache system:

Figure 30–7 Oracle IMDB Cache System Data Flow



31

Installing Rated Event Loader

This chapter explains how to install the Oracle Communications Billing and Revenue Management (BRM) Rated Event (RE) Loader software.

You should be familiar with BRM concepts and architecture. See "Introducing BRM" and "BRM System Architecture" in *BRM Concepts*, and "Understanding Rated Event Loader".

Important: RE Loader is an optional feature that requires a separate license.

About Installing RE Loader

The Rated Event Loader and Extraction Tool installation package includes the following software:

- Rated Event Loader
- Event Extraction Tool

When you run installation, both applications are installed.

You can choose to install Event Extraction Manager separately by using the custom install. (See "Installing RE Loader".)

Important: If you are upgrading from a previous version of RE Loader, you must make sure that all available unrated events are rated by Pipeline Manager and loaded before installing this version of RE Loader.

System Requirements

RE Loader is available for the HP-UX IA64, Linux, AIX, and Solaris operating systems. For information on disk space requirements, see "Disk space requirements" in *BRM Installation Guide*.

Software Requirements

Before installing RE Loader, you must install:

- For Oracle systems:
  - Oracle database software. For compatible versions, see "BRM Software Compatibility" in *BRM Installation Guide*.

Important: The server and client must be installed on systems that have identical OS versions.

  - Oracle SQL Loader

Important: Configure and test the SQL Loader utility before you install RE Loader.

  - (For IMDB Cache-enabled systems) Oracle IMDB Cache 32-bit client libraries
  - Oracle database 32-bit libraries

Important: You must enable support for Java in the Oracle database to use RE Loader.

- Third-Party software, which includes the PERL libraries and JRE required for installing BRM components. See "Installing the Third-Party Software" in *BRM Installation Guide*.
- BRM

Note: The Oracle Data Manager (DM) does not have to be installed on the same system that RE Loader is installed on. See "About Configuring RE Loader".

About Configuring RE Loader

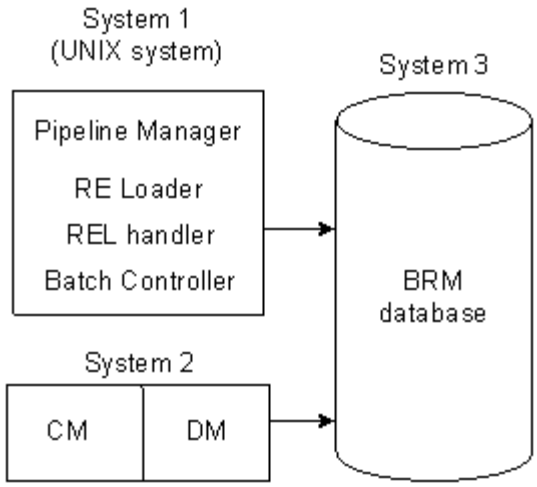
RE Loader uses Batch Controller, which needs access to the pipeline output files. Therefore, Pipeline Manager, Batch Controller, and RE Loader software should be installed on the same system that contains the pipeline output files.

If Pipeline Manager and RE Loader are on different systems, you need to map the Pipeline Manager output directories to a drive local to RE Loader.

Figure 31–1 shows the recommended configuration for installing RE Loader and its related features:

Note: Event Extraction Tool can be installed on any of these systems or on its own system.

Figure 31–1 Recommended Configuration for RE Loader Installation



Tip: Though it is possible to install RE Loader on a BRM system or the database system, you will get better performance if you install it on the Pipeline Manager system.

Installing RE Loader

To install RE Loader, perform the procedures in these sections:

1. Granting Execute Permission for dbms_lock
2. Granting Write Permission to the DM
3. Installing the RE Loader Package
4. Creating Your RE Loader Database Partitions
5. Returning DM Permissions to their Original Values

Granting Execute Permission for dbms_lock

Before you install RE Loader, you must grant execute permission to *pin_user* for **dbms_lock**:

1. Log in to your database as the SYS user:

   ```
   sqlplus sys/password@databaseAlias
   ```

2. Grant execute privileges to *pin_user*:

   ```
   grant execute on dbms_lock to pin_user
   ```

Granting Write Permission to the DM

When you install RE Loader on a system where BRM is not installed, you must grant the DM write permission before installing RE Loader.

Perform the following on all machines containing a DM:

1. In a text editor, open your DM configuration file:
   - For Oracle DM systems: *BRM_Home*/**sys/dm_oracle/pin.conf**

- For IMDB Cache DM systems: *BRM_Home*/**sys/dm_tt/pin.conf**

BRM_Home is the directory where you installed BRM components.

2. Write down the values of your **dd_write_enable_fields**, **dd_write_enable_objects**, **dd_write_enable_portal_objects**, and **dd_mark_as_portal** entries.
3. Set the values of the following entries to **1**:

```
- dm dd_write_enable_fields 1
- dm dd_write_enable_objects 1
- dm dd_write_enable_portal_objects 1
- dm dd_mark_as_portal 1
```

> **Note:** If any entry is not in the file, add it.

For more information, see comments in the DM **pin.conf** file.

4. Save and close the file.
5. Stop and restart the DM.

You can now install RE Loader.

Installing RE Loader on Non-IMDB Cache Enabled Systems

When installing RE Loader, the installer, by default, looks for the IMDB environment variable, even if IMDB Cache is not installed.

To install RE Loader on a non-IMDB cache enabled system:

1. Create an empty, dummy directory, using the following command:

```
mkdir -p DUMMY_DIR/lib
```

2. Set the environment variable to point to the dummy directory you have created, as follows:

```
setenv TIMESTEN_CLIENT_HOME DUMMY_DIR
```

Installing the RE Loader Package

To install RE Loader:

1. Go to the Oracle Software Delivery Cloud Web site http://edelivery.oracle.com.
2. Download the software to a temporary directory (*temp_dir*).

> **Important:** If you download to a Windows workstation, use **FTP** to copy the **.bin** file to a temporary directory on your UNIX server.

3. Increase the heap size used by the Java Virtual Machine (JVM). For information, see "Increasing heap size to avoid "Out of Memory" error messages" in *BRM Installation Guide*.
4. Go to the directory where you installed the Third-Party package and source the **source.me** file.

Caution: You must source the **source.me** file to proceed with installation, otherwise "suitable JVM not found" and other error messages appear.

Bash shell:

```
source source.me.sh
```

C shell:

```
source source.me.csh
```

5. Go to *temp_dir* and run the following command:

   7.5_RatedEventLoader_*platform*_**opt.bin**

   where *platform* is the operating system name.

   Note: You can use the **-console** parameter to run the installation in command-line mode. To enable a graphical user interface (GUI) installation, install a GUI application such as X Windows and set the DISPLAY environment variable before you install the software.

6. If you want to install Event Extraction Manager and RE Loader separately, either on this computer or on another computer, select custom install when asked to specify the setup type. Select the features you are installing and click **Next**.

   The features are:

   - **RatedEventLoader**
   - **EventExtractionTool**

7. Follow the instructions displayed during installation.

   Note: The installation program does not prompt you for the installation directory if BRM or RE Loader is already installed on the machine and automatically installs the package at the *BRM_Home* location.

8. Go to the directory where you installed RE Loader and source the **source.me** file:

   Bash shell:

   ```
   source source.me.sh
   ```

   C shell:

   ```
   source source.me.csh
   ```

9. Go to the *BRM_Home***/setup** directory and run the **pin_setup** script.

   Note: The **pin_setup** script starts all required BRM processes.

Your RE Loader installation is now complete.

Creating Your RE Loader Database Partitions

> **Caution:** You must perform this step to ensure that the new event tables have the same partitioning layout as your existing event tables. If you install several optional components, perform this step only after installing the last component.

To create partitions for RE Loader events:

1. On the system where BRM is installed, go to the *BRM_Home*/**apps/partition_utils** directory.
2. Run the **partition_utils** utility to enable delayed-event partitions:

   ```
   perl partition_utils.pl -o enable -t delayed -c storable_class
   ```

   where *storable_class* specifies the event classes for which you want partitioning.

   > **Important:** You must create partitions for all subclasses of a specific service event type that you want to load.

   For example, this command creates partitions for **/event/delayed/session/telco/gsm** delayed events:

   ```
   perl partition_utils.pl -o enable -t delayed -c /event/session/telco/gsm
   ```

For more information, see the following:

- "Enabling Delayed-Event Partitions" in *BRM System Administrator's Guide*
- "partition_utils" in *BRM System Administrator's Guide*.

Your RE Loader installation is now complete.

Returning DM Permissions to their Original Values

To return your DM permissions to their original values:

1. In a text editor, open your DM configuration file:
   - For Oracle DM systems: *BRM_Home*/**sys/dm_oracle/pin.conf**
   - For IMDB Cache DM systems: *BRM_Home*/**sys/dm_tt/pin.conf**
2. Restore the following entries to their original values (the values they had before you modified them). The default value for each entry is **0**:

   ```
   - dm dd_write_enable_fields
   - dm dd_write_enable_objects
   - dm dd_write_enable_portal_objects
   - dm dd_mark_as_portal
   ```

3. Save and close the file.
4. Stop and restart the DM.

What's Next?

Configure RE Loader. See "Configuring Rated Event Loader".

Uninstalling RE Loader

To uninstall RE Loader, run the **uninstaller.bin** utility from the *BRM_Home*/**uninstaller/RatedEventLoader** directory.

32

Configuring Rated Event Loader

This chapter describes how to set up Oracle Communications Billing and Revenue Management (BRM) Rated Event (RE) Loader to load events that have been rated by Pipeline Manager.

For an overview of loading pipeline events, see "Understanding Rated Event Loader".

You should be familiar with the following topics:

- Installing, configuring, and running Pipeline Manager. See "Installing Pipeline Manager" in *BRM Installation Guide*.
- Oracle In-Memory Database (IMDB) Cache. See "Using Oracle IMDB Cache Manager" in *BRM System Administrator's Guide*.
- Oracle database concepts and implementation.

Caution:

- Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
- Do not use SQL commands to change data in the database. Always use the API.

Setting Up Your System for RE Loader

To set up your system for RE Loader:

1. Install RE Loader:
   - For Oracle Data Manager (DM) systems, install one instance of RE Loader for each BRM database schema.
   - For IMDB Cache DM systems, install one instance of RE Loader for each logical partition in IMDB Cache.
2. Configure your system to use the correct Oracle library files. See "Configuring Oracle Libraries for RE Loader".
3. Configure the RE Loader configuration (**Infranet.properties**) file. See "Configuring the RE Loader Infranet.properties File".
4. Create RE Loader processing directories. See "Setting Up RE Loader Processing Directories".
5. Configure RE Loader to work with multischema systems. See "Setting Up RE Loader for Multischema Systems".

6. Configure RE Loader to work with IMDB Cache DM systems. See "Setting Up RE Loader for IMDB Cache-Enabled Systems".
7. Configure Batch Controller to run RE Loader automatically. See "Configuring RE Loader to Run Automatically".
8. Configure the RE Loader daemon for IMDB Cache-enabled systems. See "Configuring the start_rel_daemon Script for an Oracle IMDB Cache System".
9. Disable invoice event caching. See "Disabling Invoice Event Caching".
10. Set up delayed billing. See "Enabling a Billing Delay for CDRs".
11. Increase the maximum field length in input data files. See "Configuring Field Lengths for Input Data Files".
12. Configure RE Loader to load preprocessed rated events from ECE into BRM. See "Setting Up RE Loader to Load Preprocessed Rated Events from ECE into BRM".

Configuring Oracle Libraries for RE Loader

RE Loader requires Oracle 32-bit libraries, and Pipeline Manager requires Oracle 64-bit libraries. If RE Loader and Pipeline Manager reside on the same system, make sure both the 32-bit and 64-bit Oracle libraries are installed on your system.

To support the libraries on the same machine, set up your Oracle environment so that RE Loader points to the 32-bit Oracle libraries and Pipeline Manager points to the 64-bit Oracle libraries:

1. Install the 32-bit and 64-bit libraries in separate directories on the Pipeline Manager system.

   Important: For Oracle 11*g* R2 installations, install the full 32-bit client software in a separate *Oracle_Home* location. For more information, refer to the Oracle 11*g* R2 database installation documentation.

2. Set environment variables in the **pin_rel** and **SampleRelHandler_config.values** files to point to the 32-bit libraries. See "Setting the Oracle Library Paths".
3. Set up your default Pipeline Manager environment in the *Oracle_Home*/**.cshrc** file to use Oracle 64-bit libraries. See Example 32–1.

Setting the Oracle Library Paths

To set the Oracle library paths:

1. Open the *BRM_Home*/**apps/pin_rel/pin_rel** file in a text editor. *BRM_Home* is the directory where you installed BRM components.
2. Add the following *before* the command that executes Java.

```
setenv ORACLE_HOME Oracle_Home
setenv SHLIB_PATH ${ORACLE_HOME}/lib
setenv PATH ${PATH}:${ORACLE_HOME}/bin
```

   Note: For Oracle 11*g* R2 installations, *Oracle_Home* points to the location where you installed the full 32-bit client software.

3. Save and close the file.

4. Open the *BRM_Home*/**apps/pin_rel/SampleRelHandler_config.values** file in a text editor.
5. Add these lines between the FILETYPE and HANDLER_DIR variables:

> **Note:** The paths you set depend on your system setup.

```
$ENV{'ORACLE_HOME'} = "/u01/app/oracle/product/11i64";
$ENV{'SHLIB_PATH'} =
"/usr/lib:/opt/portal/7.5/lib:/u01/app/oracle/product/11i64/lib";
$ENV{'PATH'} =
".:/bin:/usr/bin:/usr/local/bin:/u01/app/oracle/product/11i64/bin:/opt/portal/7
.5/bin";
```

6. Save and close the file.

Example 32–1 Example of the Oracle 64-Bit Setting in the .cshrc File

```
setenv ORACLE_HOME /u01/app/oracle/product/10g64
setenv ORACLE_SID       PIND10g64
setenv NLS_LANG American_America.AL32UTF8
setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH:$ORACLE_HOME/lib32:$ORACLE_
HOME/rdbms/lib32
setenv LD_LIBRARY_PATH_64 $IFW_HOME/lib:$ORACLE_HOME/lib:$ORACLE_HOME/rdbms/lib
setenv ORACLE_BIN $ORACLE_HOME/bin
setenv ORACLE_DOC $ORACLE_HOME/odoc
setenv ORA_NLS33 $ORACLE_HOME/ocommon/nls/admin/data
alias ora 'cd $ORACLE_HOME'
set path = ($path $ORACLE_BIN)
```

Configuring the RE Loader Infranet.properties File

The **Infranet.properties** file contains configuration information for processing event data record (EDR) files, such as the location of the RE Loader processing directory, how to connect to the BRM database, and how to process specific event types. The RE Loader **Infranet.properties** file contains the following sections:

- Connection entries. This section specifies how to connect to your database. The entries are different for BRM systems with IMDB Cache DM than for BRM systems with Oracle DM.
- General processing entries. This section defines your RE Loader log file, how to connect to BRM and the BRM database, and the fields to process in your EDR files.
- Default processing entries. This section defines how to process any event type. You can override these values for specific events by using the storable class-specific entries.
- Storable class-specific processing entries. This section defines how to process specific event types (for example, **/event/delayed/session/gprs** events or **/event/delayed/session/telco/gsm** events). All configuration information in this section overrides your default entries.
- The RE Loader daemon entries. This section specifies how to run the RE Loader daemon. If you do not run the RE Loader daemon, you can ignore the entries in this section.

To configure your RE Loader **Infranet.properties** file:

1. Open the *BRM_Home*/**apps/pin_rel/Infranet.properties** file in a text editor.

2. Specify how to connect to your database by setting the entries shown in Table 32–1. Replace *LogicalPartID* with the logical partition number, such as **0.1.0.1**.

Table 32–1 Database Connection Entries

Entry	Description
infranet.rel.dbname.*LogicalPartID*	Specifies the data store alias in the **sys.odbc.ini** file. This entry applies to IMDB Cache-enabled systems only. You must create an entry for each logical partition in your system.
infranet.rel.userid.*LogicalPartID*	Specifies the user ID for connecting to the specified data store. The default is **pin**. This entry applies to IMDB Cache-enabled systems only. You must create an entry for each logical partition in your system.
infranet.rel.password.*LogicalPartID*	Specifies the password for connecting to the specified data store. The default is **pin**. This entry applies to IMDB Cache-enabled systems only. You must create an entry for each logical partition in your system.
infranet.rel.dbtype	Specifies the BRM database type. The default is **oracle**.
infranet.rel.dbname	Specifies the BRM database name. **Note**: Your database name is the TNSNAMES alias in the *Oracle_Home*/**network/admin/tnsnames.ora** file. The default is **pindb**.
infranet.rel.userid	Specifies the user ID for connecting to the BRM database. The default is **pin**.
infranet.rel.password	Specifies the password for connecting to the BRM database. The default is **pin**.

3. Set the general processing entries shown in Table 32–2.

Table 32–2 General Processing Entries

Entry	Description
infranet.log.file	Specifies the name of the RE Loader log file. The default is **rel.pinlog**.
infranet.log.name	Specifies the name of the application. The default is **REL** for RE Loader.
infranet.log.level	Specifies the log reporting level: ■ **1** specifies error-level reporting. ■ **2** specifies warning-level reporting. ■ **3** specifies debug-level reporting. The default is **1**. For more information, see "Setting the Reporting Level for Logging Messages" in *BRM System Administrator's Guide*.
infranet.log.logallebuf	Specifies whether RE Loader automatically logs all EbufExceptions. The default is **True**.
infranet.rel.use_end_time	Specifies whether RE Loader uses the start time or end time of the rated event for deciding the billing cycle. ■ **1** specifies that RE Loader uses the end time of the rated event for deciding the billing cycle. The default is **1**. ■ **0** specifies that RE Loader uses the start time of the rated event for deciding the billing cycle.

Table 32–2 (Cont.) General Processing Entries

Entry	Description
infranet.connection	Specifies the user login name. For example: `infranet.connection=pcp://root.0.0.0.1:password@localhost:11960/service/pcm_client` RE Loader uses this Connection Manager (CM) connection to log audit information. **Important:** RE Loader writes audit information to the database specified in this entry. If you use a multischema system, you might want to modify this entry to write audit information to the schema where the records are loaded.
infranet.login.type	Specifies whether RE Loader requires a login name and password to log in to BRM. ■ **0** specifies that a login name and password *are not* required. ■ **1** specifies that a login name and password *are* required. The default is **1**.
infranet.rel.dbhost	Specifies the database machine's host name.
infranet.rel.dbport	Specifies the database port number. The default is **1433**.
infranet.failover	In high-availability systems, specifies the secondary CM connection. For example: `infranet.failover.1 = pcp://root.0.0.0.db_no:password@failover_host:failover_port/service/pcm_client`
infranet.rel.polling_interval	Specifies the interval, in milliseconds, that RE Loader checks the database to see whether another process is loading. The default is **1000**. The polling interval depends on the number and size of your input files. If you have very large files, make the polling interval longer. If you have many small files, make the interval shorter.
infranet.rel.polling_time_out	Specifies the time, in milliseconds, that RE Loader waits to load events before exiting. The default is **600000**. The time-out period depends on the number and size of your input files and how many parallel RE Loader processes are running. If you have very large files or many processes, make the time-out period longer.
infranet.rel.partition_set_number	Specifies the partition set number, from 1 through 7. This entry applies only to BRM databases with multiple delayed partition sets. The default is **1**. ■ **1** uses delayed partition set P_1D to P_12D. ■ **2** uses delayed partition set P_1D to P_12D2. ■ **3** uses delayed partition set P_1D to P_12D3. ■ **4** uses delayed partition set P_1D to P_12D4. ■ **5** uses delayed partition set P_1D to P_12D5. ■ **6** uses delayed partition set P_1D to P_12D6. ■ **7** uses delayed partition set P_1D to P_12D7.
infranet.rel.updater_threads	Specifies the number of threads dedicated to the update and preupdate stored procedures. You can specify a fixed number of threads or configure RE Loader to adjust the number of threads based on the number of database objects to update. To specify a fixed number of threads, set the entry equal to the desired number of threads. To configure RE Loader to automatically adjust the number of threads, set the entry to **0**. RE Loader spawns the number of threads shown below: Less than 1,000 objects: 2 threads Between 1,000 and 200,000 objects: 4 threads More than 200,000 objects: 8 threads The default is **4**. **Note:** Specifying a number of threads that exceeds the number of CPUs in your system may cause deadlock due to a lack of system resources. If you set the **infranet.rel.updater_threads** entry to a value greater than 8, RE Loader returns a warning message and continues processing.

Table 32–2 (Cont.) General Processing Entries

Entry	Description
infranet.rel.validate_dbnumber	Specifies whether RE Loader performs an extra validation step to ensure that it is loading a call detail record (CDR) file into the correct database schema. The default is **True**. **Important:** Use this option only for debugging. In a production environment, set this to **False**. Setting it to **True** degrades performance while loading data into the database. For more information, see "Turning Off Database Verification to Improve Processing Performance".
infranet.rel.validate_indexes	Specifies whether RE Loader verifies that the database indexes are correct before loading data into the database. The default is **False**. **Note:** RE Loader does not validate indexes in IMDB Cache. For IMDB Cache-enabled systems, set this to **False**. **Important:** Use this option only for debugging. In a production environment, set this to **False**. Setting it to **True** degrades performance while loading data into the database.For more information, see "Turning Off Index Verification to Improve Database Loading Performance".
infranet.rel.max_increment_by	Specifies the number of database schemas in your system. This value is used by the POID generation algorithm to ensure that POIDs are unique across all databases schema in your system. The default is **20**. For more information, see "Preventing POID Errors in Multischema Systems".
infranet.rel.sort.limit	Defines the maximum number of CDRs that the preprocessing script can sort by account ID. This improves performance later during the balance updating process. If the number of CDRs in the input file is greater than the **infranet.rel.sort.limit** value, the preprocessing script does not sort the CDRs. The default is **100000**.
infranet.rel.custom_error_codes	Specifies the name of the custom error code file. The default name is **CustomErrorCodes.properties**. If you want to move this file from its default location, you must create a symbolic link between the name of the file and its new location. To create this link, go to the *BRM_Home*/**apps/pin_rel** directory and enter the following at the command prompt: `$ ln -s path_to_where_file_was_moved /CustomErrorCodes.properties ./CustomErrorCodes.properties`
infranet.rel.default.header.record_type	Specifies the header record type. The default is **010**.
infranet.rel.default.detail.record_type	Specifies the detail record type. The default is **020**.
infranet.rel.default.trailer.record_type	Specifies the trailer record type. The default is **090**.
infranet.rel.field.delimiter	Specifies the delimiter symbol. The default is **\t**, for tabs.
infranet.rel.header.position.storable_class	Specifies which field in the EDR file contains the storable class name. The default is **20**. **Note:** When you set this field to **0**, RE Loader uses the default storable class specified in **infranet.rel.default.storable_class**.
infranet.rel.header.position.creation_process	Specifies which field in the EDR file contains the name of the creation process (for example, whether the file contains prerated, rerated, or discount events). The default is **18**. **Note:** You can specify **0** if you do not need this field validated.
infranet.rel.header.position.sender	Specifies which field in the EDR file contains the sender. The default is **3**.
infranet.rel.header.position.recipient	Specifies which field in the EDR file contains the recipient. The default is **4**.
infranet.rel.header.position.file_sequence	Specifies which field in the EDR file contains the file sequence number. The default is **5**. **Note:** You can specify **0** if you do not need this field validated.
infranet.rel.header.position.creation_ timestamp	Specifies which field in the EDR file contains the creation timestamp. The default is **7**. **Note:** You can specify **0** if you do not need this field validated.

Table 32–2 (Cont.) General Processing Entries

Entry	Description
infranet.rel.header.position.object_cache_type	Set this value to **0**.
infranet.rel.trailer.position.record_count	Specifies the field position of the field that contains the total number of detail records in the output file. The default is **7**. The field position starts with 1.
infranet.rel.file_extension.disc.transform_script	By default, this entry is commented out.
infranet.rel.file_extension.disc.transform_flags	By default, this entry is commented out.

4. Set the default configuration entries shown in Table 32–3. The configuration information in this section applies to all event types except for those defined in the storable class-specific section.

Table 32–3 Default Configuration Entries

Entry	Description
infranet.rel.default.interim_directory	Specifies the RE Loader processing directory. This is the location where preprocessed events are temporarily stored before they are loaded. The default is *BRM_Home*/**apps/pin_rel**.
infranet.rel.default.supported_creation_processes	Specifies which creation processes are supported. By default, RE Loader supports all creation processes: ■ **RATING_PIPELINE** specifies that the file was last processed by the rating pipeline and therefore contains *prerated* events. ■ **RERATING_PIPELINE** specifies that the file was last processed by the rerating pipeline and therefore contains *rerated* events. ■ **PIN_REL_TRANSFORM_CDR** specifies that the file was last processed by the **pin_rel_transform_cdr.pl** script and therefore contains *discount* events. ■ **SUSPENSE_CREATE** specifies that the RE Loader process will create new suspense records in the suspended usage table. ■ **SUSPENSE_UPDATE** specifies that the RE Loader process will update existing suspense records in the suspended usage table.
infranet.rel.default.failure_script	Specifies the script called when RE Loader attempts to reload events that previously failed to load into the database. The default is **pin_rel_handle_interim_files.pl**.
infranet.rel.default.failure_flags	Specifies the flag passed to the failure script. You can specify the following flags in the default **pin_rel_handle_interim_files.pl** failure script: ■ **0** to do nothing. ■ **1** to rename the interim files by appending **.saved.timestamp** to the file name. ■ **2** to delete the temporary files. ■ **3** to move the unsuccessfully processed data files generated by ECE to the reject subdirectory. The default is **1**.
infranet.rel.default.preprocess_script	Specifies the name of the preprocessing script. The default is **pin_rel_preprocess_cdr.pl**.
infranet.rel.default.preprocess_flags	Specifies the flag passed to the preprocessing script. The default is **0**.

Table 32–3 (Cont.) Default Configuration Entries

Entry	Description
infranet.rel.default.load_util	Specifies the name of the load utility. For Oracle's SQL Loader, it also specifies whether the utility uses direct-path loading or conventional-path loading: ■ **Direct-path loading**. This is the fastest way to load events into the database. It can be 10% to 30% faster than conventional-path loading, depending on the file size, memory size, storage configuration, and storage performance. However, direct-path loading has limits for concurrent system activities. When an event is loaded in direct-path mode, the load utility locks the event's entire partition and some of the table's indexes. This prevents other operations from updating or reading the event table. Direct-path mode is recommended when the event table will have limited concurrent usage. ■ **Conventional-path loading**. This is the recommended loading mode if BRM will perform many concurrent operations on the event table. For example, use conventional-path loading if BRM is rerating events, performing billing-time taxation, or generating detailed invoices concurrently with RE Loader. Conventional mode is also recommended if you have small source files for RE Loader because the performance gained by using direct-path loading is surpassed by the mode's preprocessing and file-handling overhead. **Important:** If you use conventional-path loading, use the APPEND option in your RE Loader control files. Do not use the TRUNCATE option. To specify the load utility name and loading mode: ■ *UtilityName* **direct=true unrecoverable** specifies to use direct-path loading. This is the default. ■ *UtilityName* **direct=false** specifies to use conventional-path loading. The default is **sqlldr direct=true streamsize=5000000 readsize=10000000 unrecoverable**.
infranet.rel.default.preupdater_sproc	Specifies the name of the preupdate stored procedure. ■ For IMDB Cache-enabled systems, set this entry to **pin_rel_tt.pin_rel_tt_pre_updater_sp**. ■ For systems that use Oracle DM, set this entry to **pin_rel.pin_rel_pre_updater_sp**. The default is **pin_rel.pin_rel_pre_updater_sp**.
infranet.rel.default.preupdater_batch_size	Specifies the size of the preupdate batch. For IMDB Cache-enabled systems, this entry must be tuned for optimum performance and memory usage. Oracle recommends setting this entry to **25**. The default is **5**.
infranet.rel.default.preupdater_flags	Specifies the flag passed to the preupdate stored procedure. The default is **1**.
infranet.rel.default.updater_sproc	Specifies the name of the update stored procedure. ■ For IMDB Cache-enabled systems, set this entry to **pin_rel_tt.pin_rel_tt_updater_sp**. ■ For systems that use Oracle DM, set this entry to **pin_rel.pin_rel_updater_sp**. The default is **pin_rel.pin_rel_updater_sp**.
infranet.rel.default.updater_batch_size	Specifies the size of the update batch. For IMDB Cache-enabled systems, this entry must be tuned for optimum performance and memory usage. Oracle recommends setting this entry to **25**. The default is **5**.
infranet.rel.default.updater_flags	Specifies the flag passed to the update stored procedure. The default is **1**.
infranet.rel.default.success_script	Specifies the script called when RE Loader successfully loads a batch of events into the BRM database. The default is **pin_rel_handle_interim_files.pl**.

Table 32–3 (Cont.) Default Configuration Entries

Entry	Description
infranet.rel.default.success_flags	Specifies the flag passed to the success script. You can specify the following flags in the default **pin_rel_handle_interim_files.pl** script: ■ **0** to do nothing. ■ **1** to rename the interim files by appending **.saved.timestamp** to the file name. ■ **2** to delete the temporary files. The default is **1**.
infranet.rel.default.storable_class	Specifies the storable class you are loading. The default is **/event/delayed/session/gprs**. **Important**: If you use conventional-path loading, use the APPEND option in your RE Loader control files. Do not use the TRUNCATE option.
infranet.rel.default.creation_process	Specifies whether the file contains prerated, rerated, or discount events: ■ **RATING_PIPELINE** specifies that the file was last processed by the rating pipeline and therefore contains *prerated* events. ■ **RERATING_PIPELINE** specifies that the file was last processed by the rerating pipeline and therefore contains *rerated* events. ■ **PIN_REL_TRANSFORM_CDR** specifies that the file was last processed by the **pin_rel_transform_cdr.pl** script and therefore contains *discount* events. **Important:** RE Loader can dynamically source the creation process from the EDR header file. Uncomment this entry only if all of your EDR files come from the same creation process. The default is **RATING_PIPELINE**.
infranet.rel.default.ece_control_file_directory	Specifies the location of the control files generated by BRM Elastic Charging Engine (ECE). The default is *BRM_Home***/apps/pin_rel**.
infranet.rel.default.ece_data_file_directory	Specifies the location of the data files generated by ECE. The default is *BRM_Home***/apps/pin_rel**.
infranet.rel.ece_preprocessed	Specifies whether RE Loader uses ECE generated preprocessed control files and data files: ■ **TRUE** specifies that RE Loader uses ECE generated preprocessed control files and data files. ■ **FALSE** specifies that RE Loader uses pipeline generated input files. The default is **FALSE**.

5. If necessary, set the storable class-specific entries shown in Table 32–4. These settings override the default settings for the specified storable class.

> **Note:** For each storable class, only the **infranet.rel.storable_class.***classname***.number_of_tables** and **infranet.rel.storable_class.***classname***.table.***N***.name** entries are mandatory. RE Loader uses the default settings for any undefined storable class-specific entries.

When editing these entries:

- Create a set of entries for each event type you want to load.
- Replace *classname* with the appropriate storable class name. For example, use **event_delayed_session_gprs** for the **/event/delayed/session/gprs** storable class.
- Create a set of ***.table.***N***.* entries for each table. For example, if the storable class contains three tables, create a set of ***.table.1.*** entries, a set of ***.table.2.*** entries, and a set of ***.table.3.*** entries.

Table 32–4 Storable Class-Specific Configuration Entries

Entry	Description
infranet.rel.storable_ class.*classname*.interim_directory	RE Loader processing directory. This is the location where preprocessed events are temporarily stored before they are loaded.
infranet.rel.storable_ class.*classname*.supported_creation_ processes	Specifies whether the file contains prerated, rerated, or discount events.
infranet.rel.storable_ class.*classname*.failure_script	Specifies the script to call when RE Loader attempts to load events that previously failed to load into the database.
infranet.rel.storable_ class.*classname*.failure_flags	Specifies the flag to pass to the failure script.
infranet.rel.storable_ class.*classname*.preprocess_script	Specifies the name of the preprocessing script.
infranet.rel.storable_ class.*classname*.preprocess_flags	Specifies the flag to pass to the preprocessing script.
infranet.rel.storable_ class.*classname*.number_of_tables	Specifies the number of tables in the storable class. **Important**: This entry is mandatory for each event type.
infranet.rel.storable_ class.*classname*.table.*N*.name	Specifies the name of a storable class table. **Important**: This entry is mandatory for each event type.
infranet.rel.storable_ class.*classname*.table.*N*.load_util	Specifies the name of the load utility. For Oracle's SQL Loader, it also specifies whether the utility uses direct-path loading or conventional-path loading: ▪ *UtilityName* **direct=true unrecoverable** specifies to use direct-path loading. ▪ *UtilityName* **direct=false** specifies to use conventional-path loading. **Important:** If you use conventional-path loading, use the APPEND option in your RE Loader control files. Do not use the TRUNCATE option.
infranet.rel.storable_ class.*classname*.table.*N*.control_file	Specifies the control file to use when loading the data file into the database.
infranet.rel.storable_ class.*classname*.preupdater_sproc	Specifies the name of the preupdater stored procedure.
infranet.rel.storable_ class.*classname*.preupdater_batch_size	Specifies the preupdater batch size.
infranet.rel.storable_ class.*classname*.preupdater_flags	Specifies the flag to pass to the preupdater stored procedure.
infranet.rel.storable_ class.*classname*.updater_sproc	Specifies the name of the updater stored procedure.
infranet.rel.storable_ class.*classname*.updater_batch_size	Specifies the updater batch size.
infranet.rel.storable_ class.*classname*.updater_flags	Specifies the flag to pass to the updater stored procedure.
infranet.rel.storable_ class.*classname*.success_script	Specifies the script to call when RE Loader successfully loads a data file into the BRM database.
infranet.rel.storable_ class.*classname*.success_flags	Specifies the flag to pass to the success script when RE Loader successfully loads a data file into the BRM database.

6. (Optional) To run the RE Loader daemon, add or modify the RE Loader daemon entries shown in Table 32–5.

   When editing these entries, replace *event* with the appropriate event type.

Table 32–5 RE Loader Daemon Entries

Entry	Description
batch.check.interval	Specifies the time interval, in seconds, to monitor files from the pipeline output directory. The default is **5**.
batch.file.rename.extension	Specifies the file name extension that the RE Loader daemon uses to rename the interim files before processing them. The default is **.bc**.
batch.start.highload.time	Specifies the start time of your system's busiest period. Specify the hour, minute, and second, in *hhmmss* format, using the 24-hour clock.
batch.end.highload.time	Specifies the start time of your system's slowest period. Specify the hour, minute, and second, in *hhmmss* format, using the 24-hour clock.
batch.lock.socket.addr	Specifies the port address of the process.
batch.rel.archiveDir	Specifies the full path to the directory where a successfully processed file is archived. This is the default archive directory for all the event handlers.
batch.rel.rejectDir	Specifies the full path to the directory where an unsuccessfully processed file is stored. This is the default reject directory for all the event handlers.
batch.random.events	Specifies the name of the event type to process. If you have two or more event types, separate each with a comma, but no blank space. For example, TEL,SMS,GPRS.
event.**max.at.highload.time**	Specifies the highest number of the RE Loader threads that are permitted to run simultaneously for this event during the high-load time; that is, from **batch.start.highload.time** to **batch.end.highload.time**. For example, if *event*.**max.at.highload.time** is **2**, two threads are permitted to run simultaneously for this event during the high-load time.
event.**max.at.lowload.time**	Specifies the highest number of the RE Loader threads that are permitted to run simultaneously for this event during the low-load time; that is, from **batch.end.highload.time** to **batch.start.highload.time**. For example, if *event*.**max.at.lowload.time** is **2**, two threads are permitted to run simultaneously for this event during the low-load time.
event.**file.location**	Specifies the full path name of the directory to monitor for the arrival of new files that match the pattern in *event*.**file.pattern**.
event.**file.pattern**	Specifies the file name pattern to look for. You can use an asterisk (*) to represent zero or more characters in the file name. No other wildcards are supported.
event.**tt_node**	Specifies the IMDB Cache logical partition for this event type. This entry applies to IMDB Cache-enabled systems only. You must create an entry for each logical partition in your system.
event.**archiveDir**	(Optional) Specifies the full path to the directory where a successfully processed file is archived for a particular event handler. When multiple event handlers are configured, configure this entry to specify the directory that archives files from a particular event handler.
event.**rejectDir**	(Optional) Specifies the full path to the directory where an unsuccessfully processed file is stored for a particular event handler. When multiple event handlers are configured, configure this entry to specify the directory that stores files from a particular event handler.
event.**file.type**	Specifies the type of input CDR file. ■ **ECE_PRE_SPLIT** specifies that RE Loader uses ECE generated preprocessed control files and data files. ■ **STANDARD** specifies that RE Loader uses pipeline generated input files. The default is **STANDARD**. **Note**: You cannot specify both **ECE_PRE_SPLIT** and **STANDARD** in the same **Infranet.properties** file.

7. Save and close the file.

Setting Up RE Loader Processing Directories

The processing directory is where you run RE Loader. It must include all RE Loader execution scripts, configuration files, and RE handler files. Most systems require only one RE Loader processing directory, but you must create additional processing directories in the following situations:

- You have a multischema system. Each database schema in your system must have a corresponding instance of RE Loader and the RE Loader processing directory.
- You have an IMDB Cache system with multiple logical partitions. Each logical partition must have a corresponding instance of RE Loader and the RE Loader processing directory.
- You want to distribute load among multiple instances of RE Loader. If your system contains multiple pipelines that generate a large number of output files, you can increase database loading performance by using multiple RE Loader instances. In this case, each instance has its own processing directory and a corresponding pipeline output directory.

> **Important:** Do not configure multiple instances of RE Loader to process the same file type from the same directory. Doing so provides no advantage and can cause errors.

To set up processing directories, do the following in each instance of RE Loader:

1. In your *BRM_Home*/**apps/pin_rel** directory, create processing directories.

   For example, create a *BRM_Home*/**apps/pin_rel/GPRS** directory and a *BRM_Home*/**apps/pin_rel/GSM** directory.

2. Configure the **Infranet.properties** file. See "Configuring the RE Loader Infranet.properties File".
3. Copy all files from the *BRM_Home*/**apps/pin_rel** directory to each processing directory.
4. If RE Loader and Pipeline Manager are installed on separate systems, do the following:
   a. Go to the system where Pipeline Manager is installed.
   b. Mount the RE Loader processing directories onto the Pipeline Manager output directory.
5. Follow the instructions in "Configuring SE Loader for Standard Recycling".

Setting Up RE Loader for Multischema Systems

To set up RE Loader for multischema systems:

1. Install and configure one instance of Pipeline Manager for *each* BRM database schema and one instance for the multischema account router.

   For example, if your BRM system contains three schemas, install and configure four instances of Pipeline Manager.

   See "Installing Pipeline Manager" in *BRM Installation Guide*.

2. In the multischema account router instance of Pipeline Manager, configure the following:

a. Set the FCT_AccountRouter module's **Mode** registry entry to **ROUTER** and configure the module's **streams** registry entry to create an output stream for each schema Pipeline Manager instance. See "FCT_AccountRouter".

b. Set the DAT_AccountBatch module's **UseAsRouter** registry entry to **True**. See "DAT_AccountBatch".

See "Using Pipeline Manager with Multiple Database Schemas" for more information.

Setting Up RE Loader for IMDB Cache-Enabled Systems

If you are using an IMDB Cache-enabled system, you must perform these additional configuration steps:

- Install the 32-bit Oracle IMDB Cache client software on your RE Loader machine.
- Setting the Oracle Library Paths for Oracle IMDB Cache
- Configuring Your Data Store Connections
- Setting Up Pipeline Manager for IMDB Cache
- Setting Up RE Loader for Multischema Systems with IMDB Cache

Setting the Oracle Library Paths for Oracle IMDB Cache

To set the Oracle library paths for Oracle IMDB Cache:

1. Open the *BRM_Home*/**apps/pin_rel/pin_rel** file in a text editor.
2. Set the following environment variables. Add these lines *before* the command that executes Java.
   - Set SHLIB_PATH to the path of the 32-bit Oracle IMDB Cache client software library directory.
   - Set TT_LIB_PATH to the path of the 32-bit Oracle IMDB Cache client software library directory.
3. Save and close the file.
4. Open the *BRM_Home*/**apps/pin_rel/SampleRelHandler_config.values** file in a text editor.
5. Set the TIMESTEN_ARGS entry:

   ```
   $TIMESTEN_ARGS="-timesten LogicalPartID";
   ```

   where *LogicalPartID* is the logical partition number, such as **0.1.0.1**
6. Save and close the file.

Configuring Your Data Store Connections

You configure your data store connections by editing the **sys.odbc.ini** and **sys.ttconnect.ini** files. These files indicate how to connect to the database alias specified in the **infranet.rel.dbname.***LogicalPartID* RE Loader **Infranet.properties** entry.

To configure your data store connections:

1. On the machine where you are running RE Loader, open the *IMDB_32_Home*/**info/sys.odbc.ini** file in a text editor. Replace *IMDB_32_Home* with the installation directory of the 32-bit Oracle IMDB Cache client software.
2. Add the following entries *for each* logical partition in your IMDB Cache system:

```
[LogicalPartitionName]
TTC_Server=HostName_Active
TTC_Server_DSN=DataStoreName_Active
TTC_Server2=HostName_Standby
TTC_Server2_DSN=DataStoreName_Standby
```

where:

- *LogicalPartitionName* is the name you assigned to the logical partition. This value must match the value you entered in the **infranet.rel.dbname.***LogicalPartID* RE Loader **Infranet.properties** entry.
- *HostName_Active* is the IMDB Cache server's logical name where the active data store resides.
- *DataStoreName_Active* is the name of the active data store.
- *HostName_Standby* is the IMDB Cache server's logical name where the standby data store resides.
- *DataStoreName_Standby* is the name of the standby data store. In a high-availability system with clusterware, the standby data store name is the same as the active data store name.

Notes:

- The **TTC_Server2** and **TTC_Server2_DSN** entries are optional and apply only to high-availability systems.
- The **sys.odbc.ini** file includes other optional entries that you can set for each logical partition. For more information, see the Oracle IMDB Cache documentation.

For example, if your system contains two logical partitions, with each containing an active and a standby data store:

```
[IMDB_Partition_1]
TTC_Server=Server1Host
TTC_Server_DSN=tt_0.0.0.1
TTC_Server2=Server2Host
TTC_Server2_DSN=tt_0.0.0.1

[IMDB_Partition_2]
TTC_Server=Server2Host
TTC_Server_DSN=tt_0.1.0.1
TTC_Server2=Server1Host
TTC_Server2_DSN=tt_0.1.0.1
```

3. Save and close the file.
4. Open the *IMDB_32_Home***/info/systtconnect.ini** file in a text editor.
5. Add the following entries *for each* IMDB Cache server in your IMDB Cache system:

```
[HostName]
Network_Address=Address
TCP_Port=PortNumber
```

where:

- *HostName* is the IMDB Cache server's logical name. This value must match the value you set in the **sys.odbc.ini** file's **TTC_Server** entry.

- *Address* is the network address for the IMDB Cache instance.
- *PortNumber* is the port number for the IMDB Cache instance.

For example:

```
[Server1Host]
Network_Address=Server1Host.company.com
TCP_Port=53389

[Server2Host]
Network_Address=Server2Host.company.com
TCP_Port=53389
```

6. Save and close the file.

Setting Up Pipeline Manager for IMDB Cache

In an IMDB Cache-enabled system, you must configure Pipeline Manager to split events based on its target logical partition.

1. Install an instance of Pipeline Manager.

   See "Installing Pipeline Manager" in *BRM Installation Guide*.

2. Configure Pipeline Manager to create separate streams for each logical partition:

   a. Set the IRL_EventTypeSplitting_tt module's **Active** entry to **True**. See "IRL_EventTypeSplitting_tt".

   b. Configure the **IRL_EventTypeSplitting_tt.data** file to map logical partitions and service codes to output streams. See "Sending EDRs to an Output Stream Based on Logical Partition and Service Code".

   c. Set the DAT_AccountBatch module's **TimesTenEnabled** registry entry to **True**. See "DAT_AccountBatch".

   For an example of how to set up your Pipeline Manager registry file for IMDB Cache systems, see the *Pipeline_Home*/**conf/wireless_tt.reg** registry file. *Pipeline_Home* is the directory where you installed Pipeline Manager. This sample registry file is preconfigured for an IMDB Cache system with two logical partitions

Setting Up RE Loader for Multischema Systems with IMDB Cache

To set up RE Loader for multischema systems with IMDB Cache:

1. Install and configure one instance of Pipeline Manager for *each* BRM database schema and one instance for the multischema account router.

   For example, if your BRM system contains three schemas, install and configure four instances of Pipeline Manager.

   See "Installing Pipeline Manager" in *BRM Installation Guide*.

2. In the multischema account router instance of Pipeline Manager, configure the following:

   a. Set the FCT_AccountRouter module's **Mode** registry entry to **ROUTER** and configure the module's **streams** registry entry to create an output stream for each schema Pipeline Manager instance. See "FCT_AccountRouter".

   b. Set the FCT_AccountLPRouter module's **Active** entry to **True**.

The FCT_AccountLPRouter module must appear after the FCT_AccountRouter module in the pipeline registry file. See "FCT_AccountLPRouter".

c. Set the DAT_AccountBatch module's **UseAsRouter** registry entry to **True** and the module's **TimesTenEnabled** registry entry to **False**. See "DAT_AccountBatch".

For an example of how to set up the account router registry for IMDB Cache multischema systems, see the *Pipeline_Home***/conf/acc_router_tt.reg** registry file.

3. Configure each schema Pipeline Manager instance to create separate streams for each logical partition:

a. Set the IRL_EventTypeSplitting_tt module's **Active** entry to **True**. See "IRL_EventTypeSplitting_tt".

b. Configure the **IRL_EventTypeSplitting_tt.data** file to map logical partitions and service codes to output streams. See "Sending EDRs to an Output Stream Based on Logical Partition and Service Code".

c. Set the DAT_AccountBatch module's **TimesTenEnabled** registry entry to **True**. See "DAT_AccountBatch".

For an example of how to set up the registry for each schema Pipeline Manager instance, see the *Pipeline_Home***/conf/acc_post_router_schema1_tt.reg** registry file.

Setting Up RE Loader for Virtual Column-Enabled Systems

This section explains the setup required for RE Loader to work in a virtual column-enabled system. For information about enabling virtual columns in the BRM database, see the discussion on virtual columns in *BRM System Administrator's Guide*.

RE Loader populates some of the event tables. After you generate virtual columns on event tables in your BRM installation, you must run the **pin_gen_classid_values.pl** script. Running the script ensures that the proper mapping of BRM object types and their corresponding object IDs is created for your extended event objects in a virtual column-enabled system.

To set up RE Loader for virtual column-enabled systems:

1. Go to *BRM_Home***/setup/scripts**.
2. Open the **pin_gen_classid_values.pl** file and verify that the first line in the file is pointing to the location of Perl in your installation.
3. Run the Perl script **pin_gen_classid_values.pl**.

   Running the script regenerates the **classid_values.txt** file that is used by RE Loader. The **classid_values.txt** file has the mapping of BRM object types (**poid_types**) and their corresponding object IDs (**object_ids**).

If you have extended BRM objects and these extended objects are new event subclasses that impact RE Loader, you must create new SQL Loader (**sqlldr**) control files. To create new **sqlldr** control files, follow the steps for adding new event types for RE Loader to load in "Loading Prerated Events" in *BRM Configuring Pipeline Rating and Discounting*. In virtual column-enabled systems, the RE Loader **sqlldr** control files must use the keywords VIRTUAL_CHAR and VIRTUAL_CONSTANT in the section that specifies the data definition of rows and also in the constant section.

Configuring RE Loader to Run Automatically

To configure RE Loader to run automatically, do the following for *each instance of RE Loader*:

- Configuring the RE Loader Batch Handler
- Configuring Batch Controller

> **Note:** For more information, see "About Running RE Loader Automatically".

- Specify each RE Loader batch handler and handler settings in the Batch Controller configuration file. See "Handler Identification" in *BRM System Administrator's Guide*.

Configuring the RE Loader Batch Handler

> **Important:** If you use the ConfigurableValidityHandler batch handler for loading the validity periods of products, discounts, and resources that start on first usage, do not use the **SampleRelHandler_config.values** file as instructed below. Instead, configure the RE Loader batch handler in the ConfigurableValidityHandler configuration file (*BRM_Home*/**apps/pin_rel/ConfigurableValidityHandler_config.values**). ConfigurableValidityHandler runs both the **pin_rel** utility and the utility for loading validity data. See "Configuring the ConfigurableValidityHandler Batch Handler".

To configure the RE Loader batch handler:

1. Give the **SampleRelHandler.pl** file a unique name. You will configure Batch Controller to call the handler using this name.
2. Create the following subdirectories: An **archive** subdirectory where successfully processed files can be stored and a **reject** subdirectory where unsuccessfully processed files can be stored.
3. Open the **SampleRelHandler_config.values** file and modify the entries shown in Table 32–6.

Table 32–6 Mandatory RE Loader Batch Handler Configuration Entries

Entry	Description
$TIMESTEN_ARGS	(For IMDB Cache-enabled systems only.) Specifies the IMDB Cache logical partition name. The value uses the following format: **-timesten** *LPidentifier* where *LPidentifier* is the logical partition name specified in the **infranet.rel.dbname.***LogicalPartID* entry of the **pin_rel Infranet.properties** file.
$FILETYPE	Specifies the EDR file-name pattern to look for. Change the value of this entry if you want to load only specific files. The default is ***.dat.bc** (any data file processed by Batch Controller). Batch Controller runs the RE Loader batch handler for each file with a name that matches this pattern. **Tip:** You can use an asterisk (*) to represent zero or more characters in the file name. No other wildcards are supported.
$HANDLER_DIR	Specifies the full path to the directory containing the RE Loader batch handler, which is this processing directory.

Table 32–6 (Cont.) Mandatory RE Loader Batch Handler Configuration Entries

Entry	Description
$pinRELDir	Specifies the full path to the directory containing the RE Loader application, which is this processing directory.
$pinREL	Specifies the full path to the batch application executable.
$STAGING	Specifies the full path to the pipeline output directory. If you specify a directory other than the pipeline output directory, use the UNIX command that links the pipeline output directory to the input staging directory.
$PROCESSING	Specifies the full path to the directory from which EDR files are processed. The default is **$pinRELDir**. This must be the same directory specified in the following RE Loader **Infranet.properties** entries: ■ **infranet.rel.default.interim_directory** ■ **infranet.rel.storable_class.***classname***.interim_directory**
$ARCHIVE	Specifies the full path to the directory where a successfully processed file is archived. This is the **archive** subdirectory created in step 2. Change this value if you used a name other than the default, **$pinRELDir/archive**.
$REJECT	Specifies the full path to the directory where an unsuccessfully processed file is stored. This is the **reject** subdirectory created in step 2. Change this value if you used a name other than the default, **$pinRELDir/reject**.

4. Save and close the file.

Configuring Batch Controller

The RE Loader package includes Batch Controller. If your system already has Batch Controller installed, the RE Loader installer does not install another. Use the sample Batch Controller properties file (*BRM_Home***/apps/pin_rel/SampleBatchControllerInfranet.properties**) to configure Batch Controller.

The default configuration for the sample Batch Controller runs RE Loader batch handler whenever a rated EDR file appears in the pipeline output directory. You can change this setting to trigger the RE Loader batch handler at specified times or based on other kinds of occurrences by editing the event entries that trigger the RE Loader batch handler. For more information, see "Setting Activity Times and Triggers" in *BRM System Administrator's Guide*.

To configure Batch Controller:

1. Copy the *BRM_Home***/apps/pin_rel/SampleBatchControllerInfranet.properties** file to your *BRM_Home***/apps/batch_controller** directory and change its name to **Infranet.properties**.
2. Open the *BRM_Home***/apps/batch_controller/Infranet.properties** file.
3. Edit the BRM connection parameters.

   For more information, see "Common Properties File Entry Syntax" in *BRM System Administrator's Guide*.
4. Set the **relHandler.start.string** parameter to the path of the RE Loader batch handler and to the name of the handler script that you gave it when configuring the RE Loader batch handler. See step 1 in "Configuring the RE Loader Batch Handler".

   For example:

   ```
   relHandler.start.string    BRM_Home/apps/pin_rel/REL_handler_name.pl
   ```
5. (Optional) To change the number of RE Loader batch handler processes you want to run, set the maximum batch handler entries.

> **Note:** The number of RE Loader batch handler processes that are called depends on the number of EDR files in the pipeline output directory. You should test your RE Loader performance if you change these entries. See "About Running Multiple RE Loader Processes" for more information.

```
relHandler.max.at.highload.time 3
relHandler.max.at.lowload.time 3
```

6. Set the **cdrFileEvent.file.location** parameter to specify the location of the pipeline output directory:

   ```
   cdrFileEvent.file.location /export/Portal/integRate
   ```

7. Set the **cdrFileEvent.file.pattern** parameter to which files should be processed by Batch Controller:

   ```
   cdrFileEvent.file.pattern cdr*.dat
   ```

   > **Tip:** You can use an asterisk (*) as a wildcard character to represent zero or more characters in the file name.

8. Save and close the file.
9. Stop and restart Batch Controller.

For more information about configuring Batch Controller, see "Controlling Batch Operations" in *BRM System Administrator's Guide*.

Configuring the start_rel_daemon Script for an Oracle IMDB Cache System

You use the **start_rel_daemon** script to start the RE Loader daemon.

To configure **start_rel_daemon** for an Oracle IMDB Cache system:

1. Open the *BRM_Home***/bin/start_rel_daemon** file in a text editor.
2. Search the file for the following line:

   ```
   TT_LIB_PATH=__TIMESTEN_CLIENT_HOME__/lib
   ```

3. Replace **_TIMESTEN_CLIENT_HOME__/lib** with the full path of the 32-bit Oracle IMDB Cache client software library directory.
4. Save and close the file.

Disabling Invoice Event Caching

If your system uses both RE Loader and invoicing, you must disable invoice event caching to ensure that invoices contain event details.

To disable invoice event caching:

1. Open the CM configuration file (*BRM_Home***/sys/cm/pin.conf**).
2. Set the **event_cache** entry to **0**:

   ```
   - fm_inv   event_cache   0
   ```

Note: If this entry is set to any other value or is not present in the file, invoicing assumes there is data in the event cache and produces invoices without event details.

3. Save and close the file.
4. Stop and restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Enabling a Billing Delay for CDRs

RE Loader cannot load a CDR for the next billing cycle when billing has not been completed unless the billing delay entry (**config_billing_delay**) in both the CM and the billing application configuration file (*BRM_Home*/**apps/pin_billd/pin.conf**) has been enabled (uncommented). RE Loader requires **config_billing_delay** to be enabled so that it can attach events to the past bill cycle or current bill cycle according to the event time. If you do not enable **config_billing_delay**, the CDRs for the next billing cycle are rated successfully, but RE Loader cannot load them.

Note:

- If you use pipeline-triggered billing, you do not need to enable **config_billing_delay** because RE Loader loads CDRs only for the current billing cycle.
- If you do not use delayed billing, set **config_billing_delay** to **0**.

To enable **config_billing_delay**, see the description on "Setting Up Delayed Billing" in *BRM Configuring and Running Billing*.

Configuring Field Lengths for Input Data Files

Any value in the input data file that is longer than 255 characters must include its maximum size. If the maximum size is not specified, the value is truncated to 255 characters when it is loaded into the database.

Fields in the input data file that should not be loaded into the database are specified with the label FILLER in the SQL Loader control file. If the input data file contains a FILLER field with a value longer than 255 characters, SQL Loader will abort with an error indicating the field at fault. If this happens, add the maximum field size to the field entry in the SQL Loader control file. Use this syntax:

```
Field_name   FILLER CHAR(max_size)
```

For example:

```
DISCOUNT_INFO   FILLER CHAR(2000)
```

Setting Up RE Loader to Load Preprocessed Rated Events from ECE into BRM

To set up RE Loader to load the preprocessed ECE-rated events into the BRM database:

1. Open the *BRM_Home*/**apps/pin_rel/Infranet.properties** file in a text editor.
2. Add the following entries:

```
infranet.rel.ece_preprocessed = True
```

```
infranet.rel.default.ece_control_file_directory =controlFileDirectoryPath
infranet.rel.default.ece_data_file_directory =dataFileDirectoryPath
```

where:

- *controlFileDirectoryPath* is the path to the directory in which the ECE-generated control files are stored.
- *dataFileDirectoryPath* is the path to the directory in which the ECE-generated data files are stored.

See the values of **doneDirectoryPath** and **dataFileDirectoryPath** parameters in the *ECE_Home***/oceceserver/config/management/charging-settings.xml** file, where *ECE_Home* is the directory in which ECE is installed, for the path of the *controlFileDirectoryPath* and *dataFileDirectoryPath* directories.

3. Add the following entries:

```
batch.random.events event1,event2
. . .
event1.file.location BRM_Home/apps/pin_rel
event1.file.pattern BRMCDR*.out
event1.file.type ECE_PRE_SPLIT
. . .
event2.file.location BRM_Home/apps/pin_rel
event2.file.pattern BRMCDR*.out
event2.file.type ECE_PRE_SPLIT
```

where *eventN* is the event identifier for the handler.

4. Create a set of these entries for each event identifier you want to load.

For example:

```
batch.random.events TEL,SMS
. . .
TEL.file.location BRM_Home/apps/pin_rel
TEL.file.pattern BRMCDR*.out
TEL.file.type ECE_PRE_SPLIT
. . .
SMS.file.location BRM_Home/apps/pin_rel
SMS.file.pattern BRMCDR*.out
SMS.file.type ECE_PRE_SPLIT
```

5. Save and close the file.

Configuring Whether to Perform Redo Generation

By default, RE Loader skips redo generation when loading files into the BRM database. This optimizes loading performance, but it can cause you to lose data if your system shuts down ungracefully.

You can re-enable redo generation by removing the UNRECOVERABLE option from each RE Loader control file.

To enable redo generation, do the following for each table's control file:

1. Open the *BRM_Home***/apps/pin_rel/***control_file* file in a text editor, where *control_file* can be one of the files shown in Table 32–7.

Table 32–7 RE Loader Control Files

File Name	Description
event_bal_impacts_t.ctl	Control file for the EVENT_BAL_IMPACTS_T table.
event_delayed_act_wap_inter_t.ctl	Control file for the EVENT_DELAYED_ACT_WAP_INTER_T table.
event_delayed_session_gprs_t.ctl	Control file for the EVENT_DELAYED_SESSION_GPRS_T table.
event_sub_bals_t.ctl	Control file for the EVENT_SUB_BALS_T table.
event_sub_bal_imp_t.ctl	Control file for the EVENT_SUB_BAL_IMP_T table.
event_dlay_sess_tlcs_t.ctl	Control file for the EVENT_DLAY_SESS_TLCS_T table.
event_dlay_sess_tlcs_svc_cds_t.ctl	Control file for the EVENT_DLAY_SESS_TLCS_SVC_CDS_T table.
event_t.ctl	Control file for the EVENT_T table.
event_total_t.ctl	Control file for the EVENT_TOTAL_T table.
event_dlyd_session_tlco_gsm_t.ctl	Control file for the EVENT_DLYD_SESSION_TLCO_GSM_T table.

2. Comment out the UNRECOVERABLE option:

```
# UNRECOVERABLE
```

> **Caution:** Removing the UNRECOVERABLE option significantly decreases loading performance.

3. Save and close the file.

33

Loading Prerated Events

This chapter describes how to run Oracle Communications Billing and Revenue Management (BRM) Rated Event (RE) Loader and provides information about troubleshooting and customizing.

For an overview on how RE Loader works, see "Understanding Rated Event Loader".

For information about configuring RE Loader, see "Configuring Rated Event Loader".

Loading Events Automatically

You can load events automatically in one of the following ways:

- By running RE Loader automatically by using Batch Controller and the RE Loader batch handler. When a prerated event file is available, Batch Controller automatically starts the RE Loader batch handler, which runs the RE Loader utility (**pin_rel**).

  To configure Batch Controller and the RE Loader batch handler, see "Configuring the RE Loader Batch Handler".

  Important: Make sure you synchronize your rating and loading applications if you have configured the RE Loader batch handler to start the RE Loader utility to load rerated events. See "About Synchronizing Rating and Loading Applications" in *BRM Setting Up Pricing and Rating*.

- By running RE Loader daemon. See "Running the RE Loader Daemon".

Running the RE Loader Daemon

Important: If this is the first time you are running the RE Loader daemon on your Oracle IMDB Cache system, ensure that the parameters in the **start_rel_daemon** script are configured. See "Configuring the start_rel_daemon Script for an Oracle IMDB Cache System".

To run the RE Loader daemon:

1. Open the *BRM_Home*/**apps/pin_rel/Infranet.properties** file in a text editor. *BRM_Home* is the directory where you installed BRM components.

2. Specify how to run the RE Loader daemon by setting the entries given in Table 32–5, " RE Loader Daemon Entries".
3. Run the *BRM_Home*/**bin/start_rel_daemon** script.

   This script starts the RE Loader daemon.
4. Run the *BRM_Home*/**bin/stop_rel_daemon** script.

   This script stops the RE Loader daemon.

Loading Events Manually

Important: Make sure you synchronize your rating and loading applications when running RE Loader. See "About Synchronizing Rating and Loading Applications" in *BRM Setting Up Pricing and Rating*.

To manually load pipeline-rated events, run the RE Loader utility (**pin_rel**) from the RE Loader directory.

Manually Loading Events from One Directory

If you have only one directory and need to load more than one event type, you must make sure **pin_rel** can find the prerated event data record (EDR) file. **pin_rel** looks for the EDR file in the directory specified in the **Infranet.rel.rated_event_file** entry in the **Infranet.properties** file. Before you run RE Loader manually, make sure the input EDR file is in this specified directory. To do this, do one of the following each time you run RE Loader:

- Move the EDR file to the directory specified in the **Infranet.properties** file.
- Change the **Infranet.rel.rated_event_file** entry in the **Infranet.properties** file to point to the directory containing the EDR file.

To run **pin_rel**, see "Running RE Loader Manually".

Manually Loading Events from Multiple Directories

If you have set up multiple directories, run **pin_rel** in the directory that corresponds to the service event type to load.

Running RE Loader Manually

You can manually run RE Loader from the command line in the following ways:

- **pin_rel** *event_file_name*

  This command loads events from *event_file_name* into the BRM database and then updates the account balances, bill items, and journals.

  Account balances, bill items, and journals are updated after all events have been loaded. If an error occurs during the loading phase, RE Loader cancels the process and all events loaded in the session are deleted from the BRM database.

Note: The name of the file in the command line can be found in the pipeline registry file. For more information, see "Configuring EDR Output Processing".

- **pin_rel -timesten** *LogicalPartID event_file_name*

  Use this command for IMDB Cache-enabled systems. This command loads events from *event_file_name* into the BRM database and then updates the account balances, bill items, and journals in the specified IMDB Cache logical partition.

  Account balances, bill items, and journals are updated after all events have been loaded. If an error occurs during the loading phase, RE Loader cancels the process and deletes all loaded data from the BRM database. After you correct the error, run RE Loader again by using the default command, **pin_rel -timesten** *LogicalPartID event_file_name*.

- **pin_rel -override** *event_file_name*

  This command starts an RE Loader process if one is not already running.

  Only one RE Loader process can load the same database tables at the same time because each process locks the tables while loading them. When an RE Loader process is started, it checks the status of its last process and waits if the last process is not complete. However, if the process was manually canceled, the status may not indicate that the process has ended, even though it is no longer running. In this case, you use the **-override** option to start a new RE Loader process.

For more information about the **pin_rel** utility, see "pin_rel" in *BRM Setting Up Pricing and Rating*.

Monitoring and Maintaining RE Loader

To monitor and maintain RE Loader, perform the following:

- Troubleshooting Event Loading
- Preventing POID Errors in Multischema Systems
- Improving RE Loader Performance
- Customizing RE Loader
- Retrieving Data About Events You Load

Troubleshooting Event Loading

There are two distinct error-handling actions that RE Loader takes, depending on when the error occurs:

- If an error occurs while events are being loaded, the process is canceled and all events loaded in the session are deleted from the BRM database. The SQL loader errors are logged in a "bad" file (*BRM_Home***/apps/pin_rel/***EDR_file_name***.bad**) and a fatal error is recorded in the RE Loader log file (*Processing_directory***/rel.pinlog**).
- If an error occurs while RE Loader is updating account balances, bill items, or journals, the loaded events are left in the database and an error is recorded in the RE Loader log file (*Processing_directory***/rel.pinlog**). If RE Loader stops due to errors while updating account balances, bill items, or journals, correct the problem and run RE Loader again.

Some error messages are sent to the console. To find out if an error occurred during rated event loading, check the **rel.pinlog** log file. See "Checking the RE Loader Log Files for Error Codes".

RE Loader checks for status in two places:

- The **/batch/rel** session status object.

  This object stores the status of the last RE Loader process. When you start RE Loader, it checks that status. If you try to reload a file that RE Loader has already successfully updated, the file is rejected because the session status indicates that the update for that file is complete.

- The REL_SUB_PROCESSES_T table.

  This tables stores information about loading errors that occurred during the preupdating stage. See "Checking for Errors that Occurred during the PreUpdate Process".

Checking the RE Loader Log Files for Error Codes

RE Loader uses the SQL Loader utility, **sqlldr**, to load events into the BRM database. The **sqlldr** process creates a new log file for each input file so that log files from a previous process are not overwritten.

The log files and the temporary files created during preprocessing incorporate the name of the input file in their file names, making it easier to debug if an error occurs.

Error codes follow the fully qualified error code (FQEC) scheme, which consists of a major code that represents the component and a minor code that represents the error number. All BRM-defined errors use a minor code from 0 through 99, and all custom errors use minor codes 100 and above.

For information on how to create custom error codes for RE Loader scripts and utilities, see "Creating Custom Error Codes".

> **Note:** Because modifying a stored procedure can corrupt data and cause maintenance and upgrade problems, custom error codes cannot be created for stored procedures.

The major and minor error codes for each RE Loader component are shown in Table 33–1.

Table 33–1 RE Loader Major and Minor Error Codes

Component	Description	Major Code	BRM Reserved Minor Codes	Customer Reserved Minor Codes
All	Universal code for success.	0	N/A	N/A
RE Loader driver	**pin_rel** script and Java driver code.	1000	0 - 999	N/A
Failure script	Script that is called when RE Loader attempts to load a data file that previously failed to load into the BRM database.	2000	0 - 99	100 - 255
Transform script	**pin_rel_transform_cdr.pl** script, which converts pipeline discount files into EDR format.	3000	0 - 99	100 - 255
Preprocess script	**pin_rel_preprocess_cdr.pl** script, which preprocesses the data files and creates bulk-loadable (**.blk**) files.	4000	0 - 99	100 - 255

Table 33–1 (Cont.) RE Loader Major and Minor Error Codes

Component	Description	Major Code	BRM Reserved Minor Codes	Customer Reserved Minor Codes
Load utility	**sqlldr** utility, which loads data into the BRM database.	5000	0	1 - 999
Preupdate stored procedure	Stored procedure for updating the loaded data before releasing the partition to other RE Loader sessions.	7000	0 - 99	Not available
Update stored procedure	Stored procedure for updating account balances, bill items, and journals.	8000	0 - 99	Not available
Success script	Script that runs automatically when RE Loader successfully loads a data file into the BRM database.	9000	0 - 99	100 - 255
Database consistency check stored procedure	Stored procedure for verifying that the database indexes are correct before loading data into the database.	10000	0 - 99	Not available

Table 33–2 shows the BRM-defined error codes and messages, where *value* is the value returned in the error message:

Table 33–2 BRM-Defined Error Codes

RE Loader Error Number	Error Message
1000	REL encountered an error.
1002	The **infranet.rel.dbtype** properties value found is not supported: *value* Supported values are: *value*
1003	The **infranet.rel.partition_set_number** properties value found is not valid: *value* Valid values are between *value* and *value.*
1004	A table name properties value is missing for the given storable-class: *value*
1005	A duplicate table name properties value was found: *value*
1006	The **load_util** properties value is missing for the given storable-class: *value*
1007	A control file properties value is missing for the given storable-class: *value*
1008	The control file name could not be found in the command line.
1009	REL cannot be executed until the Event Extraction Manager is complete.
1010	An unexpected SQL exception has occurred.
1011	An error occurred while attempting to connect to the BRM database.
1012	An error occurred while attempting to connect to the CM. Please validate the **infranet.connection** property value and ensure the CM is running.
1013	An error occurred while attempting to perform an opcode call.
1014	An interrupt has occurred and caused an error.
1015	The following file was not found: *value*
1016	An unexpected I/O error was encountered.
1017	The POID selected from the database sequence exceeds the maximum supported range of 2^{44}: *value*
1018	REL failed to select the partition name from the database.
1019	The poid_db could not be found in the input file.
1020	The poid_db found in the input file does not match the BRM database number for this CM connection. Found: *value* Expected: *value*

Table 33–2 (Cont.) BRM-Defined Error Codes

RE Loader Error Number	Error Message
1021	The header record could not be found in the input file.
1022	The storable-class was not defined, or was not found in the header record.
1023	The time format found in the header record is not valid: *value*
1024	The creation process found in the header record is not supported: *value* Valid values are: *value*
1026	An invalid command-line was provided.
1027	The CM and JDBC BRM database connections are not configured to the same database schema.
1028	The REL session has timed out waiting for another REL session to complete.
1029	The file has previously completed successfully so it will not be loaded again: *value*
1030	The file is currently being processed by another REL session: *value*
1031	The *value* key is missing from the properties file.
1032	The *value* value is missing from the properties file.
1033	The configured number of tables for this storable-class does not match the configured tables: *value*
1034	A number formatting error was encountered in the properties value for: *value*
1035	The **infranet.rel.updater_threads** properties value found is not valid: *value* Valid values are between *value* and *value*. To have REL auto-choose an appropriate number of threads, use the value: *value*
1036	An error occurred while attempting to parse a number for: *value*
1038	Cannot have control file with 'TRUNCATE' option when running REL in parallel loading mode between multiple REL processes.

Table 33–3 shows the BRM-defined failure script error codes.

Table 33–3 Failure Script Error Messages

Failure Script Error Number	Error Message
2000	The failure script encountered an error. The given command-line was: *value*
2001	The failure script command-line given arguments are not supported. The given command-line was: *value*
2002	The failure script command-line given flags value provided is not supported. The given command-line was: *value*
2003	The failure script command-line given directory could not be read. The given command-line was: *value*

Table 33–4 shows the BRM-defined transform script error codes.

Table 33–4 Transform Script Error Messages

Transform Script Error Number	Error Message
3000	The transform script encountered an error.
3001	The transform script command-line given arguments are not supported. The given command-line was: *value*

Table 33–4 (Cont.) Transform Script Error Messages

Transform Script Error Number	Error Message
3002	The transform script command-line given input file could not be read. The given command-line was: *value*
3003	The transform script command-line given output file could not be created.
3004	The transform script command-line given negative discount carry over value is invalid. The given command-line was: *value*

Table 33–5 shows the BRM-defined preprocess script error codes.

Table 33–5 Preprocess Script Error Messages

Preprocess Script Error Number	Error Message
4000	The preprocess script encountered an error. The given command-line was: *value*
4001	The preprocess script command-line given arguments are not supported. The given command-line was: *value*
4002	The preprocess script failed to open a file.
4003	The preprocess script found the input file to be missing a balance record. The given command-line was: *value*
4004	The preprocess script found the input file to be missing a detail record. The given command-line was: *value*
4005	The preprocess script command-line given tables are not supported. The given command-line was: *value*
4006	The preprocess script command-line given increment_by value is not valid. The given command-line was: *value*
4007	The preprocess script did not find the expected number of records in the input file. The given command-line was: *value*
4008	The preprocess script found the input file to be missing an EDR record. The given command line was: *value* Used by SE Loader.
4009	The preprocess script did not find the expected EDR size for an EDR record. The given command line was: *value* Used by SE Loader.
4010	The preprocess script failed to parse fields mapping data for generating the control file. The given command line was: *value* Used by SE Loader.

Table 33–6 shows the BRM-defined load utility error codes.

Table 33–6 Load Utility Error Messages

Load Utility Error Number	Error Message
5000	The database load utility encountered an error.

Table 33–7 shows the BRM-defined insert stored procedure error codes.

Table 33–7 Insert Stored Procedure Error Messages

Insert Stored Procedure Error Number	Error Message
6000	The insert stored procedure encountered an error.

Table 33–8 shows the BRM-defined preupdate stored procedure error codes.

Table 33–8 Preupdate Stored Procedure Error Messages

Preupdate Stored Procedure Error Number	Error Message
7000	The preupdate stored procedure encountered an error.
7001	The preupdate stored procedure encountered an error on a select statement.
7002	The preupdate stored procedure encountered an error on an insert statement.
7003	The preupdate stored procedure encountered an error on an update statement.
7004	The preupdate stored procedure encountered an error on a delete statement.
7008	The preupdate stored procedure encountered a parsing error.
7010	The preupdate stored procedure could not find an item for an account.
7011	The preupdate stored procedure encountered an unexpected error.

Table 33–9 shows the BRM-defined update stored procedure error codes.

Table 33–9 Update Stored Procedure Error Messages

Update Stored Procedure Error Number	Error Message
8000	The update stored procedure encountered an error.
8001	The update stored procedure encountered an error on a select statement.
8002	The update stored procedure encountered an error on an insert statement.
8003	The update stored procedure encountered an error on an update statement.
8004	The update stored procedure encountered an error on a delete statement.
8008	The update stored procedure encountered a parsing error.
8009	The update stored procedure found its record is already being processed.
8010	The update stored procedure could not find an item for an account.
8011	The update stored procedure encountered an unexpected error.
8012	The update stored procedure encountered an invalid record count error.
8013	The update stored procedure encountered an error when updating the account balances.
8014	The update stored procedure encountered an error when updating the item balances.
8015	The update stored procedure encountered an error at TREL precommit.
8016	The update stored procedure encountered an error at TREL postcommit.

Table 33–10 shows the BRM-defined success script error codes.

Table 33–10 Success Script Error Messages

Success Script Error Number	Error Message
9000	The success script encountered an error. The given command-line was: *value*
9001	The success script command-line given arguments are not supported. The given command-line was: *value*
9002	The success script command-line given flags value provided is not supported. The given command-line was: *value*
9003	The success script command-line given directory could not be read. The given command-line was: *value*

Table 33–11 shows the BRM-defined database consistency check error codes.

Table 33–11 Database Consistency Check Error Messages

Database Consistency Check Error Number	Error Message
10000	The database consistency check encountered an error.
10005	The database consistency check found an unpartitioned index.
10006	The database consistency check found an incorrectly partitioned index.
10007	The database consistency check found an unusable index.

Checking for Errors that Occurred during the PreUpdate Process

Errors that occur during the preupdate stage of the loading process are stored in the REL_SUB_PROCESSES_T table. To check for values in the table, run SQL*Plus.

Table 33–12 shows the error codes that are stored in the REL_SUB_PROCESSES_T table:

Table 33–12 Error Codes Stored in the REL_SUB_PROCESSES_T Table

Status Code	Status Number	Description
ERROR_SELECTING	–20001	An error occurred when selecting data from a table or tables.
ERROR_INSERTING	–20002	An error occurred during the insert process.
ERROR_UPDATING	–20003	An error occurred during the update process.
ERROR_DELETING	–20004	An error occurred during the delete process.
ERROR_UNPARTITIONED_INDEX	–20005	An error occurred because the index is not partitioned.
ERROR_INCORRECT_PART_INDEX	–20006	An error occurred because the index is global partitioned.
ERROR_UNUSABLE_INDEX	–20007	The index partitions are unusable.
ERROR_PARSING	–20008	An error occurred during the data parsing process.
ERROR_ALREADY_BEING_ PROCESSED	–20009	An error occurred because the record is being processed by another thread.
ERROR_ITEM_NOT_IN_ACCOUNT	–20010	An error occurred because the item is already billed and **pre_updater_flag** is not enabled.

Table 33–12 (Cont.) Error Codes Stored in the REL_SUB_PROCESSES_T Table

Status Code	Status Number	Description
ERROR_UNEXPECTED	–20011	An unexpected error occurred in the pre-update procedure.
ERROR_UPDATE_ACCT_ BALANCES	–20013	An error occurred while updating account balances.
ERROR_UPDATE_ITEM_BALANCES	–20014	An error occurred while updating item balances.

Fixing Event Loading Errors

In order to troubleshoot event loading errors, check the RE Loader log file *BRM_Home*/**apps/pin_rel/rel.pinlog**, where *BRM_Home* is the directory in which you installed BRM components. See "Checking the RE Loader Log Files for Error Codes" for more information.

At times, when RE Loader fails, the **rel.pinlog** file does not list the error. If this occurs, check the status column in the BATCH_T table in the BRM database for the status of the REL process. Table 33–13 lists the status entries (and the corresponding code attributes).

Table 33–13 Status Entries in the BATCH_T Table

Status	Code Attribute
0	UPDATE_COMPLETE
1	LOAD_ERROR
2	UPDATE_ERROR
4	INSERT_ERROR
8	PREUPDATE_ERROR
16	REL_START
48	PRE_PROCESS
64	START_LOAD
80	LOADING
96	LOAD_COMPLETE
107	FAIL_TO_START_REL
240	PROCESS_LOADING
256	START_INSERT
512	INSERTING
768	INSERT_COMPLETE
1024	START_PREUPDATE
1280	PREUPDATING
1536	PREUPDATE_COMPLETE
3840	PROCESS_PREUPDATING
4096	START_UPDATE
8192	UPDATING
61440	PROCESS_UPDATING

The correct troubleshooting effort for an event loading error depends upon the error scenario:

- RE Loader fails to start:

  The RE Loader log file (**rel.pinlog**) displays the error code **107** (see Table 33–13)

  In this error scenario, RE Loader failed to sart. The error occurs if REL is not running.

  To troubleshoot this error, start REL using the following command:

  ```
  rel<rated event file>
  ```

- Load failure:

  The RE Loader log file (**rel.pinlog**) displays the error code **5000**. The status entry for the REL process in the BATCH_T table in the BRM database displays **1** (see Table 33–13).

  In this error scenario, RE Loader failed either before or during the loading of the events in the event file. The events are deleted from the event tables.

  To troubleshoot this error, reload the events normally by using the same command to process the original event file.

- RE Loader fails during the loading:

  The RE Loader log file (**rel.pinlog**) does not display any error. The status entry for the REL process in the BATCH_T table in the BRM database displays **80** (see Table 33–13).

  In this error scenario, RE Loader failed during the loading of the events and RE Loader was unable to update the session status or execute the cleanup process.

  Use the **-override** option to start a new process to reload the events. For example:

  pin_rel -override *event_file_name*

  where *event_file_name* is the event file.

- Error occurs during the preupdate stored procedure:

  The RE Loader log file (**rel.pinlog**) displays preupdate stored procedure error codes starting at **7000** and below **8000**. The status entry for the REL process in the BATCH_T table in the BRM database displays **8** (see Table 33–13).

  In this error scenario, RE Loader crashed during the execution of the preupdate stored procedure.

  To troubleshoot this error, reload the events normally by using the same command to process the original event file.

- RE Loader fails during the updating of events:

  The RE Loader log file (**rel.pinlog**) does not display any error. The status entry for the REL process in the BATCH_T table in the BRM database displays **8192** (see Table 33–13).

  In this error scenario, RE Loader crashed during the updating of the events and RE Loader was unable to update the session status or execute the cleanup process.

  To troubleshoot this error, reload the events normally by using the same command to process the original event file.

- Error occurs during the update stored procedure:

The RE Loader log file (**rel.pinlog**) displays update stored procedure error codes starting at **8000** and below **9000**. The status entry for the REL process in the BATCH_T table in the BRM database displays **2**.

In this error scenario, RE Loader successfully loaded the events but failed during the execution of the update stored procedure. The BATCH_REL_SUB_PROCESSES_T table lists the last commit, indicating the point at which the database update failed.

Reload the events normally. The update starts from this point.

Debugging Mismatches between Data Files and Control Files

RE Loader customizations can sometimes cause data files and control files to become unsynchronized, resulting in SQL Loader failures. To help you debug these situations, use the **pin_rel_enum_blk.pl** script, which enumerates fields in your bulk-loadable files. You can then manually compare the data file entries to the control file.

To debug mismatches between your data files and control files, enter the following commands:

```
% cd BRM_Home/apps/pin_rel
% pin_rel_enum_blk.pl file_name [Line_num]
```

where:

- *file_name* specifies the name of the bulk-loadable file. For example, **test2.blk**.
- *Line_num* specifies the line number of the bulk-loadable file that you want to enumerate. The default is **1**.

Preventing POID Errors in Multischema Systems

BRM multischema systems ensure that all POIDs are unique across all database schemas by using a POID-generation algorithm. This BRM algorithm sets each schema's starting sequence number to a unique value and then increments each sequence number by a set value. By default, BRM sets the increment value equal to the number of schemas in your system.

For example, if your system contains three schemas:

- Schema 1 uses a starting sequence number of 10000
- Schema 2 uses a starting sequence number of 10001
- Schema 3 uses a starting sequence number of 10002

The incremental value is 3.

This example results in the following POID numbers shown in Table 33–14:

Table 33–14 Example Schema POID Numbers

Time	POID for Schema 1	POID for Schema 2	POID for Schema 3
1	10000	10001	10002
2	10003	10004	10005
3	10006	10007	10008

When RE Loader loads a batch of objects into the BRM database, it reserves a group of POIDs as follows:

1. Changes the increment value by using the following equation:

   ```
   (Number of objects to load) x (Current increment value)
   ```

   For example, if RE Loader needs to load 2,000 objects into the database and the current increment value is 3, it changes the increment value to 2,000 x 3 = 6,000.

2. Allocates POIDs to objects.

3. Returns the increment value to its original value.

However, if a major error occurs during the allocation process, the increment value can remain at the incorrect high value. To catch these situations, you can configure RE Loader to check the database increment value against a specified maximum before it reserves a group of POIDs. When the increment value exceeds the specified maximum, RE Loader exits and logs an error message, notifying your database administrator to manually reset the increment value.

To configure RE Loader to compare the increment value against a specified maximum:

1. Open the *BRM_Home*/**apps/pin_rel/Infranet.properties** file in a text editor.

2. Set the **infranet.rel.max_increment_by** entry to the number of database schemas in your system:

   ```
   infranet.rel.max_increment_by = 20
   ```

   The default is **20**.

3. Save and close the file.

Improving RE Loader Performance

You can improve your RE Loader system performance by:

- Increasing the Number of Account Balance and Bill Item Updates
- Turning Off Index Verification to Improve Database Loading Performance
- Turning Off Database Verification to Improve Processing Performance
- Pruning Your RE Loader Control and Audit Tables

Increasing the Number of Account Balance and Bill Item Updates

RE Loader performance might be improved by increasing the number of account balance, bill item, and journal updates performed before committing the transaction.

You can modify the preupdate batch size and update batch size in the **Infranet.properties** file to specify how many updates to perform before committing the transaction. For example, if **updater_batch_size** is set to **5**, the stored procedure commits the transaction after every five updates. Increasing the number of updates might increase performance, but the updated account balances, bill items, and journals are not available until the transaction is committed. The default **batch_size** value is **5**.

> **Important:** Setting the **batch_size** value too high can result in deadlock. The value for best performance depends on your system configuration. You should test to find the best value for your system.

To change the **preupdater_batch_size** and **updater_batch_size** values:

1. Open the *BRM_Home*/**apps/pin_rel/Infranet.properties** file in a text editor.

> **Note:** If you have already set up your RE Loader processing directories, make sure you edit the **Infranet.properties** file in each directory.

2. If necessary, edit the **infranet.connection** entry to point to the correct database.

   For example:

   ```
   infranet.connection=pcp://root.0.0.0.1:password@localhost:37180/service/pcm_
   client
   ```

3. Specify the preupdater batch size value in the **preupdater_batch_size** entry.

   For example:

   ```
   infranet.rel.default.preupdater_batch_size = 8
   ```

4. Specify the updater batch size value in the **updater_batch_size** entry:

   ```
   infranet.rel.default.updater_batch_size = 8
   ```

5. Save and close the file.

Turning Off Index Verification to Improve Database Loading Performance

By default, RE Loader automatically verifies that your indexes are correct before loading data into the BRM database. This extra step helps you discover configuration errors when testing your system in a development environment.

In production systems, however, you should turn off index verification to improve database loading performance.

When configured to verify indexes, RE Loader performs the following before it runs the SQL Loader utility:

1. Checks whether the indexes to load are partitioned, local, and usable.
2. Performs one of the following:
   - If the indexes are incorrect, RE Loader aborts the loading process and logs which indexes encountered problems.
   - If the indexes are correct, RE Loader runs the SQL Loader utility to load events into the database.

When configured to skip verification, RE Loader automatically runs the SQL Loader utility to load events into the database. When the indexes are incorrect, SQL Loader fails and RE Loader logs only that the database load utility encountered an error.

To turn off index verification:

1. Open the *BRM_Home*/**apps/pin_rel/Infranet.properties** file in a text editor.
2. Set the **Infranet.rel.validate_indexes** entry to **False**:

   ```
   Infranet.rel.validate_indexes = False
   ```

3. Save and close the file.

Turning Off Database Verification to Improve Processing Performance

By default, RE Loader automatically verifies that it is loading events into the correct database schema by validating the database number in the EDR file's first account

object with the PCM database number. This extra step helps you discover configuration errors when testing your multischema system in a development environment.

In production systems, however, you should turn off database verification to improve RE Loader database loading performance.

To turn off database verification:

1. Open the *BRM_Home*/**apps/pin_rel/Infranet.properties** file in a text editor.
2. Set the **infranet.rel.validate_dbnumber** entry to **False**:

   ```
   infranet.rel.validate_dbnumber = False
   ```

3. Save and close the file.

Pruning Your RE Loader Control and Audit Tables

RE Loader control and audit tables grow indefinitely, so you should prune them periodically to increase system performance and reduce memory usage. To make pruning easier, you can use the RE Loader **purge_batch_rel_objects** stored procedure, which automatically prunes the tables for you.

To prune your control and audit tables, run the following commands in SQL*Plus:

```
sqlplus system/manager@DatabaseAlias
pin_rel.purge_batch_rel_objects(int:Number)
```

where *Number* specifies how many days worth of data to keep in the tables.

Customizing RE Loader

Some of the steps required to customize RE Loader should be performed by a programmer and database administrator. To customize RE Loader, you should be familiar with the following topics:

- BRM system architecture. See "BRM System Architecture", in *BRM Concepts*, and "Understanding Rated Event Loader".
- BRM storable classes. See "Understanding Flists and Storable Classes" in *BRM Developer's Guide*.
- BRM database configuration. See "Database Configuration and Tuning" in *BRM Installation Guide*.
- SQL and creating SQL control files. See your SQL documentation.

You can customize RE Loader by:

- Adding New Event Types for RE Loader to Load
- Creating Custom Error Codes

> **Important:** *Do not modify the **rel_updater_sp.sql** stored procedure or any other stored procedure.* Modifying a stored procedure can corrupt data and cause maintenance and upgrade problems. Stored procedures are delivered in source code format due to database limitations and are not designed to be modified. If you need to modify a stored procedure, you must obtain specific permission to do so from Oracle.

Important: Sub-balances are stored in events in an internal format that optimizes performance and storage efficiency. As a result, the table that stores sub-balances is not visible in the data dictionary. You should not base your customizations on this specific internal format. All sub-balance data is accessible by using the BRM API. If you need to access this internal format, contact Oracle.

Adding New Event Types for RE Loader to Load

When you offer a new service, you create a new storable class for the service event type. For information about adding a new service to BRM, see "Adding Support for a New Service" in *BRM Developer's Guide*.

To use RE Loader to load events from a new service or new service subclass, you must create a delayed event type for your new service and configure RE Loader to load it.

It is possible to load a subclass of a preconfigured service event type without configuring that subclass. However, BRM will not be aware of the subclass because the subclass events will be inserted into the parent class table. To track the activity of the subclass events, you configure RE Loader to load the specific subclass.

You must create a new delayed event type for RE Loader prerated events. The new event storable class type must start with **/event/delayed** so that BRM can distinguish it from real-time events. For example, **/event/delayed/session/***new_event_type*.

Important: Avoid loading prerated events by using RE Loader and another application such as an optional component or Universal Event Loader.

To add an event type for RE Loader to load:

1. If necessary, add the new event type storable class to BRM by using Storable Class Editor. See the Storable Class Editor Help. For information about storable classes, see "About Storable Classes and Storable Objects" in *BRM Developer's Guide*.

   Note: If you installed GSM Manager, the **/telephony**, **/fax**, **/data**, and **/sms** subclasses of **/event/delayed/session/telco/gsm** already exist in the BRM database and do not need to be created. However, if you want to track activity specific to one of these subclasses, you must perform this entire procedure.

2. Create partitions for the event type by running the **partition_utils** utility from the *BRM_Home***/apps/partition_utils** directory.

   For example, the following command creates partitions for **/event/delayed/session/telco/gsm** delayed events:

   ```
   partition_utils -o enable -t delayed -c /event/session/telco/gsm
   ```

   Important: You must create partitions for all subclasses of a specific service event type that you want to load.

   See "Enabling Delayed-Event Partitions" and "partition_utils" in *BRM System Administrator's Guide*.

3. Create a new control file for the new event type. A control file and format file specifies the format for a single database table (array or substruct). If you added new fields to an existing array or substruct, modify the control or format file for that table. If you added a new array or substruct, create a new control or format file for the new table. For instructions on creating a control or format file, see your Oracle documentation.
4. If you created or modified any control files, modify the RE Loader preprocess script (*BRM_Home*/**apps/pin_rel/pin_rel_preprocess_cdr.pl**) to read the new event fields from the EDR data and write the fields to the files that are loaded by SQL Loader. You can follow the steps used for **/event/session/telco/gsm** in the **pin_rel_preprocess_cdr.pl** file as a guide.
5. Create a new RE Loader directory corresponding to the pipeline output directory. See "Setting Up RE Loader Processing Directories".
6. Add the following entries to the RE Loader **Infranet.properties** file in each directory:
   - The new event type
   - A new service record type corresponding to the new event type
   - The new control file that loads the new event
   - The new event tables that hold the new event type

   For information, see "Configuring the RE Loader Infranet.properties File" and the **Infranet.properties** file.
7. If you are running RE Loader automatically, you must also do the following:
   a. Configure the RE Loader batch handler in the new directory to load the new event type. See "Configuring the RE Loader Batch Handler".
   b. Add entries for the new RE Loader batch handler in the Batch Controller configuration file. See "Handler Identification" in *BRM System Administrator's Guide*.

Creating Custom Error Codes

You can create custom error codes for RE Loader scripts and utilities by using the RE Loader **CustomErrorCodes.properties** file. You use this file to list your custom error codes and messages. All entries should follow the FQEC scheme and be grouped with the correct component. For more information, see "Checking the RE Loader Log Files for Error Codes".

To create custom error codes:

1. Modify the RE Loader script or utility to report the error. For more information, see the comments in the appropriate script or utility.
2. Open the *BRM_Home*/**apps/pin_rel/CustomErrorCodes.properties** file in a text editor.
3. Add your custom error code to the file, making sure you use a minor code in the customer-reserved range.

   For example, the following entry creates a custom error code for the load utility:

   ```
   5100 = Sample load utility error message for a custom return code of 100.
   ```

4. Save and close the file.

Retrieving Data About Events You Load

BRM stores information about events loaded by RE Loader in a **/batch/rel** object. This object contains the input file name, number of records loaded, and other session information. To display the event object, use Event Browser.

Note: If you use multiple database schemas, the **/batch/rel** object is created in the schema specified in the RE Loader **Infranet.properties** file.

Part VI

Pipeline Manager Reference

Part VI provides reference information about Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager. It contains the following chapters:

34

BRM Rating EDR Container Description

This chapter describes the rating EDR container fields that are used by Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager.

For more information, see "About Pipeline Rating".

For information on EDR-to-TAP mapping, see "TAP and EDR Field Mappings" in *BRM Configuring Roaming in Pipeline Manager*.

Naming Conventions

The following definitions listed in Table 34–1 are used in this document to describe the record format:

Table 34–1 Record Format Definitions

Symbol	Meaning
X	Alphanumeric, left-justified, filled with trailing spaces to the right.
Z	Numeric, left-justified, filled with trailing spaces to the right.
H	Hexadecimal value (0-9, A-F), right-justified, filled with leading zeros to the left.
9	Numeric, right-justified, filled with leading zeros to the left.
(m)	Specifies the length in characters: mandatory.
[n]	Specifies the decimal precision. Optional.

Oracle CDR Format

The Oracle CDR format is the standard file structure used by Pipeline Manager to process CDRs during the input and output processes.

The Oracle CDR format has the following characteristics:

- Each record is separated by a newline character (**\n**).
- Each record contains data for one service only.
- Each record contains a fixed number of fields.
- Each field is tab delimited (**\t**).
- Each field is in a specified position within a record. For example, the first field in the Header Record is the Record Type, the second is the Sender, and so forth.

Note: If a field does not have a value, the field is left blank. The result is a tab followed by another tab.

- Each field has a specified data type and format. For example, the **A number** must be a string that is 10 characters long**.**

To process CDRs, Pipeline Manager converts the CDRs to the internal EDR format by using the stream format, grammar, and mapping description files.

The stream format file describes the structure of the Oracle CDR format.

The following example shows the section of the stream format description file that describes the format of the Header Record. The Header Record is identified by the record type **010**. The fields are separated by a tab (**\t**), and the record is terminated by a newline character (**\n**). It specifies the list the fields in the record. The first field is the record type (RECORD_TYPE), the second is the record number (RECORD_NUMBER), and so forth. RECORD_TYPE uses the AscString() data type, RECORD_NUMBER uses the AscInteger( ) data type.

```
HEADER(SEPARATED)
{
Info
{
Pattern = "010.*\n";
FieldSeparator = '\t';
RecordSeparator = '\n';
}
RECORD_TYPE                     AscString();
RECORD_NUMBER                   AscInteger();
SENDER                          AscString();
RECIPIENT                       AscString();
SEQUENCE_NUMBER                 AscInteger();
ORIGIN_SEQUENCE_NUMBER          AscInteger();
CREATION_TIMESTAMP              AscDate();
TRANSMISSION_DATE               AscDate("%Y%m%d");
TRANSFER_CUTOFF_TIMESTAMP       AscDate();
UTC_TIME_OFFSET                 AscString();
SPECIFICATION_VERSION_NUMBER    AscInteger();
RELEASE_VERSION                  AscInteger();
ORIGIN_COUNTRY_CODE              AscString();
SENDER_COUNTRY_CODE              AscString();
DATA_TYPE_INDICATOR              AscString();
IAC_LIST                         AscString();
CC_LIST                          AscString();
UTC_END_TIME_OFFSET              AscString();
}
```

The grammar files are used to verify the data formats and to normalize the data. For example, if a field is supposed to be 10 characters, Pipeline Manager uses the grammar file to perform this check. If the data is of an incorrect format, the CDR is rejected.

The mapping files are used to map CDR fields to the EDR container fields.

The following example shows a section of the InMap description file, which is used during the input process. This example shows how the fields in the Header Record of a CDR are mapped to the EDR container fields.

```
HEADER
{
STD_MAPPING
```

```
{
RECORD_TYPE                   -> HEADER.RECORD_TYPE;
RECORD_NUMBER                 -> HEADER.RECORD_NUMBER;
SENDER                        -> HEADER.SENDER;
RECIPIENT                     -> HEADER.RECIPIENT;
SEQUENCE_NUMBER               -> HEADER.SEQUENCE_NUMBER;
ORIGIN_SEQUENCE_NUMBER        -> HEADER.ORIGIN_SEQUENCE_NUMBER;
CREATION_TIMESTAMP            -> HEADER.CREATION_TIMESTAMP;
TRANSMISSION_DATE             -> HEADER.TRANSMISSION_DATE;
TRANSFER_CUTOFF_TIMESTAMP     -> HEADER.TRANSFER_CUTOFF_TIMESTAMP;
UTC_TIME_OFFSET               -> HEADER.UTC_TIME_OFFSET;
SPECIFICATION_VERSION_NUMBER -> HEADER.SPECIFICATION_VERSION_NUMBER;
RELEASE_VERSION               -> HEADER.RELEASE_VERSION;
ORIGIN_COUNTRY_CODE           -> HEADER.ORIGIN_COUNTRY_CODE;
SENDER_COUNTRY_CODE           -> HEADER.SENDER_COUNTRY_CODE;
DATA_TYPE_INDICATOR           -> HEADER.DATA_TYPE_INDICATOR;
IAC_LIST                      -> HEADER.IAC_LIST;
CC_LIST                       -> HEADER.CC_LIST;
UTC_END_TIME_OFFSET           -> HEADER.UTC_END_TIME_OFFSET;
}
}
```

EDR Format Structure

The BRM EDR format consists of the following components:

1. Exactly one Header Record. Record type 010.
2. Zero or more Basic Records in no specific order:
   a. Basic Detail Record; for example, record type 020.
   b. More basic records might be defined in the future.
3. Zero or more Associated Records, related to one Basic Record, in the following order:
   a. Associated Service Extension Records; for example, record type 520.
   b. Associated CAMEL Extension Records. Record type 700.
   c. Associated BRM Balance Record. Record type 900.
   d. Associated Zone Breakdown Record; for example, record type 960.
   e. Associated Charge Breakdown Record; for example, record type 981.
   f. Associated Message Description Record. Record type 999.
4. Exactly one Trailer Record. Record type 090.

Example Structure

Table 34–2 contains an example BRM EDR structure.

Table 34–2 BRM EDR Example Structure

Record	Description
Header Record: 010	Once. Mandatory.
Basic Detail Record: 040	Once. Optional.
Associated GPRS Extension Record: 540	Once. Optional.

Table 34–2 (Cont.) BRM EDR Example Structure

Record	Description
Associated CAMEL Extension Record: 700	Once. Optional.
Associated BRM Balance Record: 900	Once. Optional.
Supplementary Balance Impact Packet Record: 600	*n* times. Mandatory 1-*n*.
Supplementary Sub-Balance Impact Packet Record: 605	*n* times. Mandatory 1-*n*.
Supplementary Sub-Balance Info Packet Record: 607	*n* times. Mandatory 1-*n*.
Associated Zone Breakdown Record: 961	*n* times. Optional.
Supplementary Zone Packet Record: 660	*n* times. Mandatory 1-*n*.
Associated Charge Breakdown Record: 981	*n* times. Optional.
Supplementary Charge Packet Record: 660	*n* times. Mandatory 1-*n*.
Associated Message Description Record: 999	*n* times. Optional.
Basic Detail Record: 070	Once. Optional.
Associated WAP Extension Record: 550	Once. Optional.
Associated BRM Balance Record: 900	Once. Optional.
Supplementary Balance Impact Packet Record: 600	*n* times. Mandatory 1-*n*.
Supplementary Sub-Balance Impact Packet Record: 605	*n* times. Mandatory 1-*n*.
Supplementary Sub-Balance Info Packet Record: 607	*n* times. Mandatory 1-*n*.
Associated Charge Breakdown Record: 981	*n* times. Optional.
Supplementary Charge Packet Record: 660	*n* times. Mandatory 1-*n*.
Basic Detail Record: 021	Once. Optional.
Associated GSM Extension Record: 520	Once. Optional.
Supplementary Service Event Record: 520	*n* times. Optional.
Basic Service Event Record: 520	*n* times. Optional.
Associated CAMEL Extension Record: 700	Once. Optional.
Associated Charge Breakdown Record: 981	*n* times. Optional.
Supplementary Charge Packet Record: 660	*n* times. Mandatory 1-*n*.
Basic Detail Record: 127	Once. Optional.
Associated Charge Breakdown Record: 981	*n* times. Optional.
Basic ...	None
Associated …	None
Supplementary …	None
Trailer Record: 090	Once. Mandatory.

Expected File Name

The BRM EDR file name uses a format of SOL42_*SenderRecipientSequence_number*.DAT. Table 34–3 describes the attributes used in the file name.

Table 34–3 EDR File Name Attributes

Item	Format	Description
Sender	X(5)	code for the sender of the file (for example, D00D1)
Recipient	X(5)	code for the recipient of the file (for example, SOL42)
Sequence_number	9(6)	sequence number of the file (000000 to 999999)

Example: "SOL42_D00D1SOL42004711.DAT"

Record Type Ranges

The Record Type Ranges listed in Table 34–4 are defined:

Table 34–4 Record Type Ranges

Range	Record type
000 - 009	Reserved for internal usage
010	Header Record
011 - 019	Reserved for Basic Address Records
020 - 089	Basic Detail Records
090	Trailer Record
091 - 099	Reserved for internal usage
100 - 299	Basic Detail Records
300 - 319	Basic Recharge Records
320 - 399	Free
400 - 499	Reserved for further Basic Record Types
500 - 599	Associated Service Extension Records
600 - 699	Supplementary Records (for former Sub-Blocks of Associated Records)
700 - 749	Associated CAMEL / IN Records
750 - 799	Reserved for further Associated Record Types
800 - 899	Free
900 - 949	Associated Balance Records
950 - 959	Reserved for further Record Types
960 - 969	Associated Zone Breakdown Records
970 - 998	Associated Charge Breakdown Records
999	Associated Message Description Record

Note: Not all of the given Record Types have been defined. Undefined values are reserved for future use.

Header Record (RECType 010)

This record is always the first record within a file. Table 34–5 describes the fields in the Header Record.

Table 34–5 Header Record Fields

Name	Format	Description
RECORD_ LENGTH	Integer	Optional for backward compatibility.
RECORD_TYPE	String	Extended to be 3 bytes long, first byte denotes the market; for example, GSM, ISDN.010. Derivation: Mandatory. Set by the first processor and left unchanged.
RECORD_NUMBER	9(9)	Sequence number of the record in the file. Ensures a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. Always **000000001**.
SENDER	X(10)	Unique identifier of the PLMN or physical (network) operator, which is sending the file; used to determine the network, which is the sender of the data. The full list of mobile codes in use is given in MoU TADIG PRD TD. 13: PLMN Naming Conventions. Specifies a unique NOSP_ID together with the RECIPIENT. Can also be used to determine the network operator responsible for the CDRs. Derivation: Optional, but should be defaulted if not present on the input side, for example, by own NO-Id, for example, 'DTAG'. Set by the first processor and left unchanged.
RECIPIENT	X(10)	Unique identifier of the PLMN or physical (network) operator to whom the file is being sent. See the MoU TADIG PRD TD. 13: PLMN Naming Conventions for a list of mobile codes. Specifies a unique NOSP_ID together with the SENDER. Can also be used to determine the reseller or service provider who is responsible for billing these events. Derivation: Optional, but should be defaulted; for example, by your own NO-Id, such as 'DTAG'. Set by the first processor and left unchanged.
SEQUENCE_NUMBER	9(6)	Unique reference that identifies each file sent by the VPLMN or logical sender to a particular HPLMN or logical recipient. It indicates the file number of the specific file type, starting at 1 and increments by one for each new file of that type sent. Separate sequence numbering must be used for test and chargeable data. Having reached the maximum value (999999), the number restarts at 1. Validates duplicate sequence numbers and sequence number gaps. **Note:** In the case of retransmission, this number does not increment. Range: 000001 - 999999 for test data and chargeable data. Derivation: Optional, if no sequence check is performed. Mandatory, if a sequence check is performed. Should be set by the first processor and can be changed by any following processor; for example, in case of recycling to assure a unique and linear sequence order to all following processors.

Table 34–5 (Cont.) Header Record Fields

Name	Format	Description
ORIGIN_SEQUENCE_NUMBER	9(6)	Original file sequence number as generated the first time. Identical content as SEQUENCE_NUMBER, but will never be changed. Used as a reference to the original file, if any processor has changed the file sequence number. Derivation: Mandatory, defaulted by SEQUENCE_NUMBER. Set by the first processor and left unchanged.
SEQ_CHECK_KEY	String	Derivation: Optional if no sequence check is performed. Mandatory if a sequence check is performed.
SEQ_GEN_KEY	String	Derivation: Optional if no sequence check is performed. Mandatory if a sequence check is performed.
CREATION_TIMESTAMP	YYYYMMDDHHMISS	Date and time on which the file was created. Not required by GSM MoU BA. 12, but might be useful for operational purposes. Can be used to validate that at least one file/stream has been generated every day. Optional Field Usage: It is possible to read/write dates in number of seconds since 01.01.1970 00:00:00; for example, 12345. The internal representation is the format YYYYMMDDHHMISS anyway. This is just an optional input/output format conversion. Derivation: Mandatory, defaulted with the FILESYSTEM-SYSDATE or a Transaction-Start-Timestamp. Set by the first processor and left unchanged.
TRANSMISSION_DATE	YYYYMMDD	Date on which the file was sent from the sender network to the recipient network or data clearing house. Can be used to calculate the run time of a file/stream between creation and transmission. Also used as a default TRANSFER_CUTOFF_TIMESTAMP. Derivation: Mandatory, defaulted with SYSDATE. Set by the first processor and left unchanged.
TRANSFER_CUTOFF_TIMESTAMP	YYYYMMDDHHMISS	Date and time used to select calls for transfer. All records available prior to the timestamp are transferred. This gives an indication to the recipient as to how current the information is. Can be used to validate that all CDRs are prior to this date and time. Their CHARGING_START_TIMESTAMP must be equal or less. Optional Field Usage: It is possible to read/write dates in number of seconds since 01.01.1970 00:00:00; for example, 12345. The internal representation is the format YYYYMMDDHHMISS anyway. This is just an optional input/output format conversion. Derivation: Mandatory, defaulted with TRANSMISSION_DATE. Set by the first processor and left unchanged.

Table 34–5 (Cont.) Header Record Fields

Name	Format	Description
UTC_TIME_OFFSET	X(5)+/-HHMI	All timestamps are sender (VPLMN) local time. So that the time can be equated to time in the recipient (HPLMN) local time, the sender gives the difference between local time and UTC time. UTC Time Offset = Local Time minus UTC Time. Can be used to translate the TRANSFER_CUTOFF_TIMESTAMP into a unified UTC time. This might be useful if a centralized rating and billing will take place. Example: Washington DC, USA 1000hrs10/10/97 UTC Time1500hrs10/10/97 UTC Time Offset= 10 - 15 = -0500 Madrid, Spain1600hrs10/10/97 UTC Time1500hrs10/10/97 UTC Time Offset= 16 - 15 = +0100 **Note:** Where dates are different, 24 is added to the time of the greater date. Derivation: Mandatory. Set by the first processor and left unchanged.
SPECIFICATION_VERSION_NUMBER	9(2)	Uniquely identifies the format. Different specification versions indicate that the record structure has changed; for example, field length, new fields, and new record types. Used for encoding different formats. Range: 01 Derivation: Mandatory. Set by the first processor and left unchanged.
RELEASE_VERSION	9(2)	Indicates the release version within the Specification Version Number. Different Release Versions indicates that only the content of fields has changed. Used for encoding different formats. Derivation: Mandatory. Set by the first processor and left unchanged.
ORIGIN_COUNTRY_CODE	X(8)	International access and country code, which applies within the country of the network where the CDR originated. Might be useful for an international billing center to distinguish between national and international calls; for example, within the basic detail record. Range: 0049; for example, for Germany. Derivation: Mandatory. Set by the first processor and left unchanged.

Table 34–5 (Cont.) Header Record Fields

Name	Format	Description
SENDER_COUNTRY_CODE	X(8)	International access and country code that applies within the country of the sender (VPLMN). This might be different from the originating code if the sender is a clearing house or third-party operator. Might be useful for an international billing center. Range: 0049; for example, for Germany Derivation: Mandatory. Set by the first processor and left unchanged.
DATA_TYPE_INDICATOR	X(1)	The type of data contained within the file; for example, test or chargeable data. Any customer billing processor should ignore test data or at least separate these streams. Values: T: Test Data Space: Chargeable Data Derivation: Mandatory. Set by the first processor and left unchanged.
IAC_LIST	X(30)	Comma-separated list of all international access codes used within this file. Used during number normalization to detect numbers already starting with these IACs. Those numbers will not be normalized anymore. Example: "001,002" for two IACs Derivation: Optional. Set by the first processor and left unchanged.
CC_LIST	X(30)	Comma-separated list of all country codes used within this file. Used during number normalization to detect all numbers already starting with these CCs. Those numbers are normalized by adding a default IAC. Example: "49,33,1" for two CCs Derivation: Optional. Set by the first processor and left unchanged.
TAP_DECIMAL_PLACES	Integer	Derivation: Optional, but mandatory for RAP output. Set by the input grammar.
OPERATOR_SPECIFIC INFO	String	Derivation: Optional, default = '' Stores a key that identifies the CDR used to generate a specific EDR. Useful for RAP or CIBER return. Must be set by an iScript.
CIBER_FILLER	String	Optional.

Table 34–5 (Cont.) Header Record Fields

Name	Format	Description
CIBER_RECORD_TYPE	String	Optional. See CIBER specs for usage.
RETURN_INDICATOR	String	Optional. See CIBER specs for usage.
CURRENCY	String	Optional. See CIBER specs for usage.
SETTLEMENT_PERIOD	String	Optional. See CIBER specs for usage.
CLEARINGHOUSE_ID	String	Optional. See CIBER specs for usage.
BATCH_REJECT_REASON	String	Optional. See CIBER specs for usage.
BATCH_CONTENTS	String	Optional. See CIBER specs for usage.
SENDING_CLEARINGHOUSE_BID	String	Optional. See CIBER specs for usage.
CREATION_PROCESS	String	Process that created output stream.
SCHEMA_VERSION	String	Version number for schema.
EVENT_TYPE	String	BRM event type.
RAP_FILE_SEQ_NO	String	Optional. Indicates the returned account procedure (RAP) file in which the recipient public data network (PMN) returned the TAP file batch to the sender PMN. This field is a unique reference. Used in TAP files.
QUERYABLE_FIELDS_MAPPING	String	Optional Calculated for suspense handling. Contains the database column names and data types that map to queryable fields. Use this format: *column_name*:*data_type*[;*column_name*:*data_type*[;...]]
BATCH_ID	String	Optional. Set to the actual file batch ID.
UTC_END_TIME_OFFSET	X(5)	Timezone where the call terminated. Derivation: Optional.
BATCH_CTRL_INFO_START_INDEX	Integer	BatchControlInfo block start index.
BATCH_CTRL_INFO_END_INDEX	Integer	BatchControlInfo block end index.
ACCOUNTING_INFO_START_INDEX	Integer	AccountingInfo block start index.
ACCOUNTING_INFO_END_INDEX	Integer	AccountingInfo block end index.

Table 34–5 (Cont.) Header Record Fields

Name	Format	Description
NETWORK_INFO_START_INDEX	Integer	NetworkInfo block start index.
NETWORK_INFO_END_INDEX	Integer	NetworkInfo block end index.
MESSAGE_DESCRIPTION_START_INDEX	Integer	MessageDescriptionInfoList block start index.
MESSAGE_DESCRIPTION_END_INDEX	Integer	MessageDescriptionInfoList block end index.
NOTIFICATION_START_INDEX	Integer	Notification block start index.
DELAYED_ERROR_BLOCK	String	Stores the block name that has the fatal error.
OBJECT_CACHE_TYPE	Integer	Cache residency type: ■ **0**: Convergent ■ **1**:- Prepaid ■ **2**: Postpaid
TAP_FILE_TYPE	String	Type of TAP file, TAP3 or TAP311.

Basic Detail Record (RECType 020-089, 100-299)

This record references a billable event. This basic record is the primary record within the BRM format structure. Table 34–6 lists the fields in the Basic Detail Record.

Table 34–6 Basic Detail Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long. First byte denotes the market. 020 MOC Switch Mobile Originating Call 021 TA_MOC TAP Mobile Originating Call (Roaming**) 022 CFW Mobile Switch Call Forwarding 023 RCF/RFD Mobile Roaming Call Forwarding 024 SMO Mobile Short Message Originating 025 SMT Mobile Short Message Terminating 026 VMO Mobile Voice Mail Originating 027 OAB Mobile Operator Assisted Call (Basic) 028 OAS Mobile Operator Service (Call Completion) 029 MSS Mobile Supplementary Service Event 030 MTC Switch Mobile Terminating Call 031 TA_MTC TAP Mobile Termination Call (Roaming**)
RECORD_TYPE (cont.)	String	040 SGSN_MOC Serving GPRS Support Node Originating 041 SGSN_MOT Serving GPRS Support Node Terminating 042 GGSN_MOC Gateway GPRS Support Node Originating 043 GGSN_MOT Gateway GPRS Support Node Terminating 044 GPRS_SMO GPRS - Short Message Originating 045 GPRS_SMT GPRS - Short Message Terminating 046 HSCSD_MOC Mobile HSCSD Originating Call 047 HSCSD_MOT Mobile HSCSD Terminating Call 048 TA_GPRSOC TAP GPRS Originating (Roaming**) 049 TA_GPRSTC TAP GPRS Termination (Roaming**) 050 SCU Basic Service Center Usage Record 060 VAS Basic Value Added/Event Record 070 WAP Basic WAP Record 120 POC ISDN/Public Switch Originating 121 DX_POC ISDN/Public Switch Orig.(data exchange) 122 PCF ISDN/Public Switch Call Forwarding 126 PVM ISDN/Public Switch Voice Mail Originating 127 POB ISDN/Public Operator Assisted Call (Basic) 128 POS ISDN/Public Operator Service (Call Com-pl) 130 PTC ISDN/Public Switch Termination Call 131 DX_PTC ISDN/Public Switch Term. (data exchange) 220 IOCBasic Internet Record Other record types might be defined when necessary. Derivation: Mandatory. Set by the first processor and left unchanged. **Note:** Only record types 021 and 031 are treated as roaming calls.

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
RECORD_NUMBER	9(9)	Sequence number of record in file. Ensures a linear sequence order for all records; for example, as a sorting criteria. Values: Minimum: 000000002 Maximum: 999999998 Derivation: Mandatory. Set by the first processor. Important: Following modules might change this record number; for example, if new record types are inserted.
DISCARDING	9(1)	Indicates if an EDR should be discarded or rejected. Values: 0: Proceed (default) 1: Reject 2-9: Discard The values from 2 to 9 represent different discarding reasons. Derivation: Mandatory. Might be set by any processor.
CHAIN_REFERENCE	X(10)	Identifies an EDR as part of a long event that has been split into multiple EDRs. Condition: Only present if more than one record is raised for a call (default=Spaces). Value: Any six-digit number. Derivation: Optional. Might only be set by the first processor.

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
SOURCE_ NETWORK_TYPE	X(1)	The source network type; for example, GSM 900. This is needed for specific implementation models such as some satellite operators where the network originating the chargeable record might be lost. **Note:** This is a temporary solution pending further developments. Values: Mobile-Networks: A: S-41 AMPS A B: S-41 AMPS B C: S-41 Satellite D: S-95 CDMA E: S-136 TDMA F: PDC G: GSM 900 H: GSM 1800 I: GSM 1900 J: GSM 90011800 K: GSM 90011900 L: GSM Satellite M: UMTS N: Telematic O: GPRS - GGSN P: GPRS - SGSN Intercarrier-Networks: W: Inroute X: Outroute T: Transit Z: undefined Fixed-Networks: 0: General Fixed Network 1: Analog 2: ISDN 3: ADSL 4: Multiplex Other-Networks: 9: Internet Other values might apply according to the related original input format. Derivation: Optional. Might be set by the first processor and might be changed by an interconnect rating processor.

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
SOURCE_NETWORK	X(14)	Network code from which the call or message was routed. This could either be PLMN_ID or any logical operator code. Used for interconnect rating. Condition: In case of interconnect rating it is overwritten by the network operator code related to the inroute. See TRUNK_INPUT. Derivation: Optional (only mandatory for the interconnect processor).
DESTINATION_ NETWORK_TYPE	X(1)	Indicates the destination network type; for example, GSM 900. This is needed for specific implementation models such as some satellite operators where the network terminating the chargeable record might be lost. **Note:** This is a temporary solution pending further developments.Values: See SOURCE_NETWORK_TYPE. Derivation: Optional. Might be set by the first processor and might be changed by an interconnect rating processor.
DESTINATION_ NETWORK	X(14)	Network towards which the call or message is routed. Condition: Where a short message has not been delivered or where optimal routing is not used, the field is set to spaces. In case of interconnect rating, it is overwritten by the network operator code related to the outroute. See TRUNK_OUTPUT. Derivation: Optional (only mandatory for interconnect rating).
TYPE_OF_A_ IDENTIFICATION	X(1)	Specifies if the number used to identify the subscriber within the network is an IMSI or an MSISDN. This type does not relate to the A Number representation. Values: A: Internet or Account Number (default for internet) C: Calling Card Number I: IMSI M: MSISDN (default for fixed networks) P: IP Number S: SIM-ICC (default for GSM) X: undefined Derivation: Optional. Set by the first processor and left unchanged.

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
A_MODIFICATION_ INDICATOR	H(2)	Specifies if the called or calling number has been modified by the VPLMN; for example, for privacy reasons. Can be used to evaluate how to handle the number internally; for example, print the last three digits in clear text (anonymize) or suppress the complete CDR during printing a detailed invoice. Condition: PLMNs are not forced to implement this parameter. If not implemented, the number must not be modified. Values: 00: Default setting (undefined) and normal 01: Social number 02: Anonymized number 04: Special number (for example, premium rate) 08: Modified number (for example, vanity routing or short number translation) Derivation: Optional. Set by the first processor and left unchanged.
A_TYPE_OF_ NUMBER	Z(1)	Type of address associated with a particular destination or calling number. Condition: Not all networks support this parameter. Values: 0: Nature of address unknown (default) 1: International number 2: National significant number 3: Network-specific number 4: Subscriber number 5: Abbreviated number Derivation: Optional, default=0. From bits 7 - 5 of octet 1 of the GSM Address String type as defined in TS GSM 09.02. Set by the first processor and left unchanged.

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
A_NUMBERING_ PLAN	X(1)	The numbering plan associated with a particular destination or calling number. Might be useful when analyzing the A Number for normalization reasons; for example, to interpret the structure of the number (for example, distinguish IP numbers. Condition: Not all networks support this parameter. Values: 0: Type of plan unknown (default) 1: ISDN telephony (CCITT E.164) 3: Data numbering plan (CCITT X.121) 4: Telex numbering plan (CCITT F. 69) 5: Reserved for national use 6: Land mobile numbering plan (CCITT E212) 8: National numbering plan 9: Private numbering plan A: Internet, IP-number v4 B: Internet, IP-number v6 Derivation: Optional. From bits 4 - 1 of octet 1 of the GSM Address String type as defined in TS GSM 09.02. The list of values is a comprehensive list of known values and some might not occur in practice. Set by the first processor and left unchanged.

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
A_NUMBER	X(40)	Identifies the billable party; for example, the ISDN number, call-line-identity, or source IP address. For MTC and MOC calls, this number contains the subscriber number to be billed, which has not automatically to be the originating number. Condition: Can be used as an alternative to the IMSI, but could also be an Internet account number. Values: Defined in TS GSM 03.03 or in international notation. Should always be: *International_access_codeCountry_codeNational_destination_ codeSubscriber_number* Examples: Fixed: +49410676810 Mobile: 01729183333 (Roaming-MOC: 0000*MNC/MCC*, 000026202) IPv4: 192.168.10.2 (always 4 token, each 3 decimals) IPv6: ABCD:10:2:1AF:0:1F0A:F:1F0 (always 8 token, each 4 hex) Derivation: Mandatory. From the GSM item MSISDN as defined in TS GSM 12.05. Set by the first processor and left unchanged, but is normalized: Fixed: 0049410676810 Mobile: 00491729183333 (Roaming-MOC: 0000*MNC/MCC*, 000026202) IPv4: 192168010002 (dots removed, filled with leading zeros) IPv6: ABCD0010000201AF00001F0A000F01F0 (colons removed, filled with leading zeros)
B_MODIFICATION_ INDICATOR	H(2)	See A_MODIFICATION_INDICATOR. Optional.
B_TYPE_OF_ NUMBER	Z(1)	See A_TYPE_OF_NUMBER. Mandatory.
B_NUMBERING_ PLAN	X(1)	See A_NUMBERING_PLAN. Optional, only if available.

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
B_NUMBER	X(40)	Identifies the called number. Condition: If there is no called number available (for example, Internet or Telematic), a dummy number has to be inserted instead; for example, 0049. If the rating does not involve zoning, profile-related features, or special numbers, the BRM rating works with even an incomplete called number. Values: Defined in TS GSM 03.03 or in international notation. Should always be: *International_access_codeCountry_codeNational_destination_ codeSubscriber_number* Examples: Fixed: 0049410676810 (normal) Fixed: 0049112 (emergency) Fixed: 004970012345678 (vanity) Fixed: 004911833 (special) Mobile: 00491729183333 (normal) Mobile: 0049172559183333 (mailbox) Mobile: 00490172112 (emergency) Mobile: 0049017222255 (special mobile-number) Mobile: 004911833 (special fixed-number) Mobile: 000026202 (Roaming-MTC: 0000*MNC/MCC*) IPv4: 192.168.10.2 (always 4 token, each 3 decimals) IPv6: ABCD:10:2:1AF:0:1F0A:F:1F0 (always 8 token, each 4 hex) Derivation: Mandatory. From the GSM item Called Number as defined in TS GSM 12.05. This item is of type Address String and is further expanded into the items type of number, numbering plan, and the number sent across the air-interface as defined in TS GSM 04.08 and 09.02. Set by the first processor and left unchanged, but is normalized. Fixed: 0049410676810 Mobile: 00491729183333 (Roaming-MOC: 0000*MNC/MCC*, 000026202) IPv4: 192168010002 (dots removed, filled with leading zeros) IPv6: BCD0010000201AF00001F0A000F01F0 (colons removed, filled with leading zeros)
DESCRIPTION	X(50)	Description text for this usage scenario; for example, call destination description for the B Number or service description used. Values: For example, "HAMBURG" or "004940" For example, "Travel Info" or "Wakeup Call" Value is related to the original input format. Derivation: Optional. Calculated by a rating processor or directly taken out of the original CDR stream. Might be changed by any processor.

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
C_MODIFICATION_INDICATOR	H(2)	See A_MODIFICATION_INDICATOR. Optional. Only mandatory if C Number is present.
C_TYPE_OF_NUMBER	Z(1)	See A_TYPE_OF_NUMBER. Optional. Only mandatory, if C Number is present.
C_NUMBERING_PLAN	X(1)	See A_NUMBERING_PLAN. Optional. Only mandatory if C Number is present.
C_NUMBER	X(40)	Third-party number; for example, where the call was first terminated in the case of terminated transit or routed, forwarded calls. This field contains the number initiating the call forwarding. **Note:** To avoid any doubt, the 'A to C' and 'C to B' legs of a call-forwarding scenario must be treated as separate calls and the originating and terminating records should never be chained together. Condition: Only present where it is available. Values: See B_NUMBER. Derivation: Optional. From the GSM item Called Number as defined in TS GSM 12.05. This item is of type Address String and is further expanded into the items type of number, numbering plan, and the number sent across the air-interface as defined in TS GSM 04.08 and 09.02. Set by the first processor and left unchanged.
USAGE_DIRECTION	X(1)	Describes the direction of the connection that was established. Can be used by any rating and post-processor to identify the direction of the event; for example, to determine a specific call scenario. Values: 0: Originated usage 1: Terminated usage 2: Roaming originated usage 3: Roaming terminated usage Examples: 0: MOC, mobile originated WAP, … 1: MTC, mobile terminated WAP, … 2: roaming MOC 3: roaming MOC Derivation: Mandatory. Set by the first processor and left unchanged.

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
CONNECT_TYPE	X(2)	Type of connection. Can be used to identify the type of the call; for example, to determine a specific call scenario. Values: 01: Mobile to Mobile 02: Mobile to Land 03: Land to Mobile 04: Land to Land 10: Effective POTS Call 11: Effective ISDN Call 12: Effective Call Diversion 13: Subscriber Procedure 14: Subscriber Service Command 15: Effective ISDN-E Call 16: Ineffective POTS Call 17: Ineffective ISDN Call 18: Ineffective Call Diversion 19: Ineffective ISDN-E Call 20: Non Call Related Output 30: Anonymous login Other values might apply according to the related original input format. Derivation: Mandatory. Set by the first processor and left unchanged.
CONNECT_SUB_ TYPE	X(2)	Detailed description of the connection or call type. Can be used to identify the type of the call; for example, to determine a specific call scenario. Values: 01: Mobile-Call 02: Local-Call (for example, BestCity, BestFriend, etc.) 03: National-Call 04: International-Call Derivation: Mandatory. Set by the first processor and left unchanged.
CALLED_COUNTRY_ CODE	String	Derivation: Optional, but should be set when the CONNECT_TYPE indicates an international call
BASIC_SERVICE	X(3)	Uniquely identifies the basic usage-related service. A service is defined by: ■ Service type ■ Service code

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
BASIC_SERVICE	X(1)	Specifies the type of basic service. Values: 0: Tele Service (for example, GSM Tele, ISDN, analog, standard, etc.) 1: Bearer Service 2: Supplementary Service 3: Telematic Service (only if present) 4: Internet Service 5: ISDN mobile (only if present) 6: Mailbox (only if present) 7: VPN mobile (only if present) 8: GPRS 9: Switch related (for example, for direct fixed network support) E: Event/VAS Services W: WAP Other service types might be defined when necessary. Derivation: Mandatory. Set by the first processor and left unchanged.

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
SERVICE_CODE	X(2)	The Service Code is either a Teleservice Code or Bearer Service Code as determined by Service Type. Values: **Note:** Other values might apply according to the related original input format.All Tele Services: 10: All Voice 11: Telephony (default) 12: Emergency calls 13: GPRS (default) 14: HSCSD (default) 15: WAP (default) 20: All Short Message Service 21: Short Message MT/PP 22: Short Message MO/PP 30: MHS 31: Advanced MHS access 40: All Videotext 41: Videotext access profile 42: Videotext access profile 2 43: Videotext access profile 3 50: All Teletext Transmission Services 51: Teletext (Circuit Switch) 52: Teletext (Packet Switch) 61: Facsimile Group 3 & alternative voice 62: Automatic Facsimile Group 3 63: Automatic Facsimile Group 4

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
SERVICE_CODE (continued)	X(2)	All Bearer Services: 10: 3.1 KHz Bearer Group 20: Circuit data asynchronous 21: Duplex Asynchronous 300bps data circuit 22: Duplex Asynchronous 1200bps data circuit 23: Duplex Asynchronous 1200/75bps data circuit 24: Duplex Asynchronous 2400bps data circuit 25: Duplex Asynchronous 4800bps data circuit 26: Duplex Asynchronous 9600bps data circuit 30: Circuit data synchronous 32: Duplex Synchronous 1200bps data circuit 34: Duplex Synchronous 2400bps data circuit 35: Duplex Synchronous 4800bps data circuit 36: Duplex Synchronous 9600bps data circuit 40: PAD access asynchronous 41: Duplex Asynchronous 300bps PAD access 42: Duplex Asynchronous 1200bps PAD access 43: Duplex Asynchronous 1200/75bps PAD access 44: Duplex Asynchronous 2400bps PAD access 45: Duplex Asynchronous 4800bps PAD access 46: Duplex Asynchronous 9600bps PAD access 50: PAD access synchronous 54: Duplex Synch. 2400bps PAD access 55: Duplex Synch. 4800bps PAD access 56: Duplex Synch. 9600bps PAD access 60: All alternate voice/data c.d.a 61: Alt. Voice/Asynch. 300bps unrest'd digital 62: Alt. Voice/Asynch. 1200bps unrest'd digital 63: Alt. Voice/Asynch. 1200/75bps unrest'd digital 64: Alt. Voice/Asynch. 2400bps unrest'd digital 65: Alt. Voice/Asynch. 4800bps unrest'd digital 66: Alt. Voice/Asynch. 9600bps unrest'd digital 70: Alternate voice data c.d.s 72: Alt. Voice/Synch. 1200bps unrest'd digital 74: Alt. Voice/Synch. 2400bps unrest'd digital 75: Alt. Voice/Synch. 4800bps unrest'd digital 76: Alt. Voice/Synch. 9600bps unrest'd digital 80: Voice followed by data c.d.a

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
SERVICE_CODE (continued)	X(2)	All Bearer Services: (continued) 81: Voice then Asynch. 300bps unrest'd digital 82: Voice then Asynch. 1200bps unrest'd digital 83: Voice then Asynch. 1200/75bps unrest'd digital 84: Voice then Asynch. 2400bps unrest'd digital 85: Voice then Asynch. 4800bps unrest'd digital 86: Voice then Asynch. 9600bps unrest'd digital 90: All Voice followed by data c.d.s 92: Voice then Synch. 1200bps unrest'd digital 94: Voice then Synch. 2400bps unrest'd digital 95: Voice then Synch. 4800bps unrest'd digital 96: Voice then Synch. 9600bps unrest'd digital
SERVICE_CODE (continued)	X(2)	All Telematic Service: 01: SMS 02: E-Mail 03: Pull-Service 04: Short-Fax 05: Push-Service All Internet Service: 10: all Internet 11: direct Access 12: WebMaster Emergency 13: Voice over IP 14: Fax over IP 20: E-Mail 30: Active Channel 40: Video on demand 41: Music on demand 50: Newsgroup access 62: Internet Fax other values Can be used according to the network. All Switch-related Services or WAP/GPRS Services: 01...99: see related switch documentation (values used 1:1) AA...ZZ: see related switch documentation (values used 1:1) All Event/VAS Services: 00: all default Event/VAS usage other Service Codes might be defined when necessary. Derivation: Mandatory. Set by the first processor and left unchanged.

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
QOS_REQUESTED	X(5)	The type of QoS requested by the Terminal Equipment (TE) at usage setup or the QoS requested to the Network Equipment (NE). The QoS Requested is typically a normalized billable QoS value. For detailed network-related QoS attributes, see the related Associated Service Extension Record. Condition: The use of a QoS might not be appropriate; for example, call forwarding cases, short message services. Applies only where appropriate information is available. Values for Radio Channel: H(2) 00: Half Rate Channel 01: Full Rate Channel 02: Dual Rate Channel, Half Rate Preferred 03: Dual Rate Channel, Full Rate Preferred Values for Quality of Service: X(5) xxxxx: Any alphanumeric representation of the NE Value is according to the related original input format. Derivation: Optional. From the GSM item RadioChanRequested, a component of RadioChanlnfo as defined in TS GSM 12.05. or directly out of the NE-interface. Encoded as per TS GSM 04.08. Set by the first processor and left unchanged.
QOS_USED	X(5)	The type of QoS negotiated by the network. The QoS used is typically a normalized billable QoS value. For detailed network-related QoS attributes, see the related Associated Service Extension Record. Condition: The use of a QoS might not be appropriate; for example, call forwarding cases, short message services. Applies only where appropriate information is available. Values for Radio Channel: H(2) 00: Half Rate Channel 01: Full Rate Channel Values for Quality of Service: X(5) xxxxx: any alphanumeric representation of the NE Values is according to the related original input format. Derivation: Optional. From the GSM item channel type, a component of RadioChannel-Used as defined in TS GSM 12.05. Encoded as per TS GSM 04.08 or directly out of the NE-interface. Set by the first processor and left unchanged.

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
CALL_ COMPLETION_ INDICATOR	X(3)	Indicates whether a call was correctly completed or not. Optionally defines a close cause reason code. Single-Byte Values X(1): if no reason code is available: C: Call completed, charged as usual D: Call dropped, treatment open, but will first be charged as usual T: Call completed, test call, will not be charged Optional Values after rating processor: 0: After rating processor: not rated yet: should be rated later by the billing post-processor: call completed 1: After rating processor: rated: should not be rated again by a billing post-processor - call completed 2: After rating processor: not rated yet: should be rated later by the billing post-processor: call dropped 3: After rating processor: rated: should not be rated again by a billing post-processor: call dropped Double-Byte Values X(2) - if a reason code is available: 00: normal 01: partial record 02: partial call re-establishment 03: unsuccessful call attempt 04: abnormal release 05: camel init call release 16: volume limit 17: time limit 18: network element switch 19: max. change condition 20: management intervention other values can be used according to the network. Derivation: Mandatory, default C. Set by the first processor and left unchanged.
LONG_DURATION_ INDICATOR	X(1)	Specifies which part of the call, in the case of split calls. Values: S: Single (only one record present) F: First (the first record in the row of split records) I: Intermediate (one of the middle records in the row of records) L: Last (the last record in the row of split records) Derivation: Mandatory, default S. Set by the first processor and left unchanged.

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
CHARGING_START_TIMESTAMP	YYYYMMDDHHMISS	Timestamp used for start of charging. In the mobile originated case, this is as determined by the VPLMN's charging rules. In the mobile terminated case, it is also at the discretion of the VPLMN, even though the information is for use in charging by the VPLMN. Format: YYYYMMDDHHMISS; for example, 19990518190357 Optional Field Usage: It is possible to read/write dates in number of seconds since 01.01.1970 00:00:00; for example, 12345. The internal representation is the format YYYYMMDDHHMISS anyway. This is just an optional input/output format conversion. Derivation: Mandatory. From the GSM item answer-time or seizure-time as defined in TS GSM 12.05. Set by the first processor and left unchanged.
CHARGING_END_TIMESTAMP	YYYYMMDDHHMISS	Timestamp used for end of charging. In the mobile originated case, this is as determined by the VPLMN's charging rules. In the mobile terminated case, it is also at the discretion of the VPLMN, even though the information is for use in charging by the VPLMN. Format: YYYYMMDDHHMISS; for example, 19990518190357 Optional Field Usage: It is possible to read/write dates in number of seconds since 01.01.1970 00:00:00; for example, 12345. The internal representation is the format YYYYMMDDHHMISS anyway. This is just an optional input/output format conversion. Derivation: Optional. Might be set by the first processor and left unchanged. **Note:** If not present, this value can be calculated by using the start timestamp and the duration.
CREATED_TIMESTAMP	Date	Optional. The time that the event was created in BRM.

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
UTC_TIME_OFFSET	X(5)+/-HHMI	All timestamps are VPLMN or originating network local time. So that the time can be equated to time in the HPLMN or recipient network, the sender gives the difference between local time and UTC time. Can be used to translate the CHARGING_START/END_TIMESTAMP into a unified UTC time. This might be useful if a centralized rating and billing will take place. Values: UTC Time Offset = Local Time minus UTC Time Example: Washington DC, USA 1000hrs 10/10/97 UTC Time 1500hrs 10/10/97 UTC Time Offset = 10 - 15 = -0500 Madrid, Spain 1600hrs 10/10/97 UTC Time 1500hrs 10/10/97 UTC Time Offset = 16 - 15 = +0100 Sydney, Australia 0100hrs 11/10/97 UTC Time 1500hrs 10/10/97 UTC Time Offset = (01 + 24) - 15 = +1000 (Note that where dates are different, 24 is added to the time of the greater date.) Derivation: Mandatory. Set by the first processor and left unchanged.
DURATION	9(15)	Duration-based charge indicates that the field represents a Chargeable Duration. Can be used to evaluate all duration-based functions; for example, determination of the price model rating steps. Condition: For event-based charges or an inter-network account charge, the field is not relevant. URC.01 implementation of the TD.17 item Chargeable Units. Maximum Value: 999999999999999 Derivation: Mandatory, default 0. Set by the first processor and left unchanged but might be patched by some kind of rating processors.
TOTAL_CALL_ EVENT_DURATION	Integer	The total duration of the event. This should be set for all time-based services; for example, telephony. Mandatory. The default value is **0**.

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
DURATION_UoM	X(3)	Unit of Measurement associated with Chargeable Quantity Value. Can be used to interpret the quantity value, but usually not needed, because the quantity itself is sufficient for all rating steps. Values: SEC: Seconds (default) MIN: Minutes HRS: Hours or any other applicable value describing a timed quantity unit of measurement. Derivation: Mandatory, default 'SEC'. Set by the first processor and left unchanged.
VOLUME_SENT_	9(15)	In addition to the basic duration quantity value, a special volume might be defined to keep an additional rating relevant measurement. This is typically BYTES sent by the initiator (A number). Maximum Value: 999999999999999 Can be used to evaluate additional volume-based functions; for example, determination of the price model rating steps. Derivation: Mandatory, default 0. Set by the first processor and left unchanged but might be patched by some kind of rating processors.
VOLUME_SENT_ UoM	X(3)	The Unit of Measurement associated with VOLUME_SENT. Can be used to interpret the quantity value, but usually not needed because the quantity itself is sufficient for all rating steps. Values: BYT: Bytes/Characters (default) KBY: Kilobytes MBY: Megabytes GBY: Gigabyte or any other applicable value describing a metered quantity unit of measurement. Derivation: Mandatory, default 'BYT'. Set by the first processor and left unchanged.
VOLUME_RECEIVED	9(15)^	In addition to the basic duration value, a special volume might be defined to keep an additional rating relevant measurement. This is typically BYTES received by the initiator (A Number). Maximum Value: 999999999999999 Can be used to evaluate an additional volume-based functions; for example, determination of the price model rating steps. Derivation: Mandatory, default 0. Set by the first processor and left unchanged but might be patched by some kind of rating processors.

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
VOLUME_ RECEIVED_UoM	X(3)	The Unit of Measurement associated with VOLUME_RECEIVED value. Can be used to interpret the quantity value, but usually not needed because the quantity itself is sufficient for all rating steps. Values: ■ BYT: Bytes/Characters (default) ■ KBY: Kilobytes ■ MBY: Megabytes ■ GBY: Gigabyte ■ Any other applicable value describing a metered quantity unit of measurement Derivation: Mandatory, default 'BYT'. Set by the first processor and left unchanged.
NUMBER_OF_UNITS	9(15)	One of the following: ■ Original charged units (for example, beats, clicks) as applied by the sender ■ Rounded total volume charged by the sender ■ Number of events associated with this record (for example, number of SMS messages or number of internet hits/clicks) Might be useful for analyzing how many units the event was originally treated by or for storing a fourth quantity. Condition: Applies only if available. Alternative URC.01 implementation of the TD.17 item Chargeable Units. Maximum Value: 4294967296 Derivation: Mandatory, default 0. Set by the first processor and left unchanged, but might be changed by some type of rating processors.

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
NUMBER_OF_ UNITS_UoM	X(3)	The Unit of Measurement associated with NUMBER_OF_UNITS value. Can be used to interpret the quantity value, but usually not needed because the quantity itself is sufficient for all rating steps. Values: CLK: Clicks (anonymous quantity) (default) MSG: Messages PAG: Pages PAC: Packets PIC: Pieces RTS: Points MTR: Meters KMR: Kilometer SPD: Speed TRN: Transactions or any other applicable value describing a metered quantity unit of measurement. Derivation: Mandatory, default CLK. Set by the first processor and left unchanged.
RETAIL_IMPACT_ CATEGORY	X(10)	Impact category defining the usage scenario specific rate; for example, the zone value used for customer rating. Values: 00000: undefined impact category (default) 00001 - 99999: user defined Alternatively a user-defined string can be used as a value. Derivation: Optional, but mandatory after any rating processor, default 00000. Might be changed by any processor.
RETAIL_CHARGED_ AMOUNT_VALUE	9(11)	The charge for the event (for example, the retail price). This includes any toll charge but does not include any CAMEL invocation fee. Values: Space: No price given, like NULL in a database Variable floating point format: Given value, might be 0.000. The floating decimal point must be set. Minimum: -9999999999 Maximum: 99999999999 Examples: '00000000125' for 125,00 '00000012.50' for 12,50 '-0012.56780' for -12,5678 Derivation: Optional, but mandatory after any customer rating processor.

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
RETAIL_CHARGED_ AMOUNT_ CURRENCY	X(3)	Currency code as defined within the associated rate plan; for example, DEM or EUR. Derivation: Optional, but mandatory whenever the RETAIL_CHARGED_ AMOUNT_VALUE is set. Use the three-digit ISO currency code.
RETAIL_CHARGED_ TAX_TREATMENT	X(1)	Charges might be inclusive or exclusive of tax. Can be used to interpret the amount value and to distinguish between net and gross charges. Values: Y: Tax included in the charge N: Tax not included in the charge (default) Derivation: Optional, but mandatory whenever the RETAIL_CHARGED_ AMOUNT_VALUE is set.
RETAIL_CHARGED_ TAX_RATE	9(4)	Defines the tax rate applicable to the charge. Because some national legal definitions dictate that the tax rate applicable is determined by the invoice date, there is a possibility that the rate on the invoice might differ from the rate on the transfer. However, the likelihood of this happening is extremely low. Can be used to interpret the amount value and to convert between net and gross charges. Values: 0000 through 9999 (2 fixed decimals) Example: 16.00% 1600 Derivation: Optional, but mandatory whenever the RETAIL_CHARGED_ AMOUNT_VALUE is set.
RETAIL_CHARGED_ TAX_VALUE	Decimal	Derivation: Calculated, default = 0.0
WHOLESALE_ IMPACT_CATEGORY	X(10)	Wholesale/Advice of Charge - Impact category used for purchase rating. Values: 00000: undefined impact category (default) 00001 - 99999: user-defined Derivation: Optional. Might be changed by any processor.

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
WHOLESALE_ CHARGED_ AMOUNT_VALUE	9(11)	Wholesale/Advice of Charge: charge for the event (for example, the wholesale price). This includes any toll charges. Can be used to keep the original purchase charge, to evaluate a record-based margin together with the charged amount value. Values: Space: No price given, like NULL in a database Floating point format: Given value, might be 0.000. If no floating point exists, the last three digits are always taken as decimals) Minimum: -9999999999 Maximum: 99999999999 Examples: '00000012500' for 12,50 '-0001200100' for -1.200,10' '00000012.50' for 12,50 '00012.50000' for 12,50 Derivation: Optional. Usually transmitted by the sender (origin network operator) of the file, but might also be recalculated by any processor to represent the purchase charge.
WHOLESALE_ CHARGED_ AMOUNT_ CURRENCY	X(3)	See RETAIL_CHARGED_AMOUNT_CURRENCY.
ZONE_ DESCRIPTION	String	Calculated. Used by zoning and rating modules.
IC_DESCRIPTION	String	Used by the zoning and rating modules.
ZONE_ENTRY_ NAME	String	Calculated. Used by zoning and rating modules.
WHOLESALE_ CHARGED_TAX_ TREATMENT	X(1)	See RETAIL_CHARGED_TAX_TREATMENT.
WHOLESALE_ CHARGED_TAX_ RATE	9(4)	See RETAIL_CHARGED_TAX_RATE.
WHOLESALE_ CHARGED_TAX_ VALUE	Decimal	Derivation: Calculated, default = 0.0

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
TARIFF_CLASS	X(10)	Tariff Class contains tariff information as represented within the original CDR format; for example, wholesale tariff model identification. Can be used to evaluate the original rate plan configuration in conjunction with the BASIC_AoC_AMOUNT_VALUE. Condition: Only present if original purchase rate plan information is available. Values: Dependent on the original format. No format conversion will take place. See the appropriate documentation of the original format. Derivation: Optional (but should by mandatory for all mobile (GSM) related records). Might be set by any processor.
TARIFF_SUB_CLASS	X(10)	Contains detailed tariff information as represented within the original CDR format; for example, wholesale zone identification. Can be used to evaluate the origin rate plan configuration in conjunction with the WHOLESALE_CHARGED_AMOUNT_VALUE. Condition: Only present if origin purchase rate plan information is available. Values: Dependent on the original format. No format conversion will take place. See the appropriate documentation of the original format. Derivation: Optional (but should by mandatory for all mobile (GSM) related records). Might be set by any processor.
USAGE_CLASS	X(5)	Specifies a format-related usage scenario; for example, call forwarding, roaming, mailbox request, or local calls. Can be used to evaluate the origin call scenario. The call class can be used to convert a scenario into a combined zone value or to identify specific rating specialties. Therefore the call class consists of original record fields. Values: Dependent on the original format. No format conversion or normalization will take place. The content is derived from any rule-based translations of any available raw event attributes, to represent all possible usage scenarios of the origin format. 00000: undefined usage class Derivation: Mandatory. Should be set by the first processor and left unchanged.

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
USAGE_TYPE	X(5)	Specifies a customer-related usage scenario; for example, customer-to-customer call, birthday call, or closed-user-group calls. Can be used to evaluate an A Number-customer and B Number-customer related scenario (using direct access to specific customer-info-fields). The call type can be used to convert a scenario into a combined zone value or to calculate a record-based discount when estimating the charging amounts. Values: User definable values might be used. The content of the field depends on the rule-based configuration. 00000: undefined usage type Derivation: Optional. Might be changed by any rating or billing processor.
EVENT_TYPE	String	BRM event type.
SERVICE_TYPE	String	BRM service type. When the service has a subscription service, the string is separated by semicolons; for example, /service/gsm/data;/service/gsm)
BILLCYCLE_PERIOD	YYYYMMBC	Defines the next open billcycle period this event belongs to. Can be used to group or split the EDR stream into billcycle-related smaller portions. Condition: Only present if a rating or pre-billing processor evaluates the next billcycle period for the A number customer. Values: YYYY: The actual year of the next open billcycle period. MM: The actual month of the next open billcycle period. BC: The billcycle identifier. Derivation: Optional, but should be mandatory for any pre-billing processor.
PREPAID_ INDICATOR	9(2)	Specifies if the event is a prepaid event. Can be used to identify prepaid scenarios within a mixed post-/prepaid environment. Values: Default: no prepaid scenario prepaid scenario Derivation: Mandatory. Set by the first processor and left unchanged.

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
NUMBER_ ASSOCIATED_ RECORDS	9(2)	Number of associated records attached to this basic detail record. Can be used to evaluate how many associated records have to be read ahead. Values: 00: No associated records attached, next record is a basic one. 01-99: Number of associated records followed by this record. Derivation: Mandatory. Must be changed by any processor if new associated records are added.
NUMBER_OF_CDRS	Integer	Number of CDRs that were compiled into this EDR during call assembly. Derivation: Optional. Calculated.
ERROR_REJECT_ TYPE	String	Used by the FCT_Reject to reject the DETAIL to another stream than the standard reject stream. Derivation: Optional, default = ' '
OPERATOR_ SPECIFIC_INFO	String	Stores a key to identify the CDR used to generate a specific EDR. Useful for RAP or CIBER return. Derivation: Optional, default = ' ' Must be set by an iScript.
DISCOUNT_KEY	String	N/A

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
GEOGRAPHICAL_ LOCATION	String	Conditional in TAP 3.10. Indicates the geographical location of the terminal equipment. Format: This field contains comma-separated tag-value pairs that indicate the geographical location of the serving network, serving BID, serving location description, longitude, and latitude. The tag values of the corresponding fields are as follows: ■ ServingNetwork: 1 ■ ServingBID: 2 ■ ServingLocationDescription: 3 ■ Longitude: 4 ■ Latitude: 5 Example 1: If the TAP field values are as follows: ■ ServingNetwork: **AIRTEL** ■ ServingBID: **AIRBID** ■ ServingLocationDescription: **Bangalore** ■ Longitude: **111** ■ Latitude: **103** The value of DETAIL.GEOGRAPHICAL_LOCATION would be: **1,AIRTEL, 2,AIRBID, 3,Bangalore, 4,111,5,103** Example 2: If the TAP field values are as follows: ■ ServingNetwork: **AIRTEL** ■ ServingBID: **AIRBID** ■ Latitude: **103** The value of DETAIL.GEOGRAPHICAL_LOCATION would be: **1,AIRTEL, 2,AIRBID, 5,103**
FRAUD_MONITOR_ INDICATOR	String	Conditional in TAP 3.10. Indicates that the chargeable subscriber is flagged for fraud information collection purposes. Possible values: **1 -** Fraud Monitored Subscriber If the field is present, it should have a value of **1**.
ORIGINAL_BATCH_ ID	String	Optional, but might be set equal to the original file batch ID.
BATCH_ID	String	Optional, but might be set equal to the recycle or rerate file batch ID.
NE_CHARGING_ START_TIMESTAMP	Date	Network Element date/time stamp. Time at which the call started. Derivation: Optional.
NE_CHARGING_ END_TIMESTAMP	Date	Network Element date/time stamp. Time at which the call ended. Derivation: Optional.
UTC_NE_START_ TIME_OFFSET	X(5)	Optional.

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
UTC_NE_END_ TIME_OFFSET	X(5)	Optional.
UTC_END_TIME_ OFFSET	X(5)	Timezone where the call terminated. Derivation: Optional.
INCOMING_ROUTE	X(30)	Incoming route name. Optional.
ROUTING_ CATEGORY	X(20)	Category denoting the routing of the call to the destination party. Optional.
DISCARD_REASON	String	The reason for discarding the EDR. This field is set by FCT_Discard.
CREDIT_LIMIT_ CHECK	Integer	Specifies whether to perform credit limit checking on the EDR: ■ 1 = Perform a credit limit check ■ 0 = Skip the credit limit check Mandatory.
CREDIT_LIMIT_ CHECK_RESULT	Integer	Specifies whether the EDR passed or failed the credit limit check: ■ 1 = The EDR passed the simple credit limit check ■ 0 = The EDR failed the simple credit limit check Mandatory.
UNRATED_ QUANTITY	Decimal	Unrated quantity filled in after credit limit check.
REFRESH_BALANCE	Integer	Specifies whether the latest balance information should be retrieved from the database. When this field is set, the discounting module calls the balance module to get the latest balance information from the database, whether or not a balance packet is present in the EDR.
OBJECT_CACHE_ TYPE	Integer	Cache residency type. ■ **0**: Convergent ■ **1**: Prepaid ■ **2**: Postpaid
DELAYED_ERROR_ BLOCK	String	Stores the block name that has the fatal error.
EVENT_ID	String	Used by Revenue Assurance.
ITEM_TAG	String	Used by FCT_ItemAssign. Calculated.
RERATE_TAG	Integer	Used for re-rating
DROPPED_CALL_ QUANTITY	Decimal	Duration of a dropped call.

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
DROPPED_CALL_ STATUS	Integer	Status of a dropped call. ■ 0 = No dropped call service-level ERA associated with the service. ■ 1 = The call is a dropped call. ■ 2 = Continuation call. ■ 3 = Both a dropped call and a continuation call. ■ 4 = Does not meet the criteria for either a dropped call or a continuation call.
NET_QUANTITY	Decimal	Contains the summation of the BALANCE_PACKET.PIN_QUANTITY for the associated RUM.
INTERN_attributes (shown below)	As shown below	The following INTERN_ attributes are used by specific modules to temporarily store calculated values. They are all mandatory, but only from a definition point of view. The content value of these fields will be filled automatically by the appropriate modules. All other modules should use these values as read only.
INTERN_ZONE_ MODEL	Integer	The internal zone model used by zoning, rating, and discounting Mandatory. Calculated.
INTERN_ NETWORK_MODEL	String	The internal network model code. Mandatory. Calculated.
INTERN_ NETWORK_ OPERATOR	String	The internal network operator code. Used by Interconnect aggregation. Mandatory. Calculated.
INTERN_APN_ GROUP	String	The internal APN group code used by zoning. Mandatory. Calculated.
INTERN_ TERMINATING_ SWITCH_ IDENTIFICATION	String	The internal terminating switch ID. Used by output mapping Mandatory. Calculated.
INTERN_BILLING_ CURRENCY	String	The internal billing currency used by exchange rate conversion. Mandatory. Calculated.
INTERN_HOME_ CURRENCY	String	The internal home currency used by exchange rate conversion. Mandatory. Calculated.
INTERN_SLA_USC_ GROUP	String	The internal customer-related SLA-based usage scenario map group code. Mandatory. Calculated.
INTERN_SLA_RSC_ GROUP	String	The internal customer-related SLA-based rate service class map group code Mandatory. Calculated.
INTERN_SLA_ IRULE_SET	String	The internal customer-related SLA-based irule_set-code. Mandatory. Calculated.

Table 34–6 (Cont.) Basic Detail Record Fields

Name	Format	Description
INTERN_PROCESS_STATUS	Integer	Possible values are ■ 0 = normal (default) ■ 1 = recycling ■ 2 = recycling-test Mandatory. Calculated.
INTERN_BALANCE_GROUP_ID	String	The balance group of the service to which the event belongs. Optional.
INTERN_SERVICE_BILL_INFO_ID	String	The billinfo of the service's balance group. Optional.
INTERN_DISCOUNT_OWNER_ACCT_ID	String	Optional.
INTERN_SERVICE_BILL_INFO_ID	String	The billinfo of the service's balance group. Optional.
ACCOUNT_ID	String	The POID of the Customer A account. Optional.
TB_RECORD_NUMBER	Integer	The Trigger Billing Record number used by the Trigger Bill Output Mapping to assign the array index set by the grammar. Optional.
RAP_FILE_SEQ_NO	String	Indicates the Returned Account Procedure (RAP) file in which the Recipient PMN returned the TAP file record to the Sender PMN. This field is a unique reference. Used in TAP files. Optional.
PROFILE_LABEL_LIST	String	A list of unique labels of all shared profiles having attributes matching a specific EDR field or event attribute. Optional. Calculated.
DROPPED_CALL_QUANTITY	Decimal	When the EDR is flagged as a continuation call, this field stores the duration of the associated dropped call.
DROPPED_CALL_STATUS	Integer	Specifies whether the EDR is for a normal call (**0**), a dropped call (**1**), a continuation call (**2**), both a dropped call and a continuation call (**3**), or an already processed EDR (**4**).

Associated Revenue Assurance Extension Record

Table 34–7 lists the fields in the Associated Revenue Assurance Extension Record. This record is optional with an occurrence of 0 or 1 time only.

Table 34–7 Associated Revenue Assurance Extension Record Fields

Name	Format
BATCH_ID	String
CDR_FILE_NAME	String
START_TIME	String
EDR_STATUS	String

Table 34–7 (Cont.) Associated Revenue Assurance Extension Record Fields

Name	Format
REVENUE_STREAM	String
OUTPUT_STREAM	String
DISCOUNT_AMOUNT	Decimal
OLD_DISCOUNT_AMOUNT	Decimal
CHARGED_AMOUNT	Decimal
OLD_CHARGED_AMOUNT	Decimal

Associated GSM/Wireline Extension Record (RECType 520)

This record is optional and will be generated only if the related Basic Detail Record indicates a GSM or Wireline service. Table 34–8 describes the fields in the Associated GSM/Wireline Extension Record.

Table 34–8 Associated GSM/Wireline Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long. Value: 520 - GSM/Wireline Extension Record Derivation: Mandatory. Set by the first processor and left unchanged.
RECORD_NUMBER	9(9)	Sequence number of record in file. Used to ensure a linear sequence order for all records; for example, as a sorting criteria. Values: Minimum: 000000002 Maximum: 999999998 Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
PORT_NUMBER	X(24)	Identifies the customer to charge; for example, the IMSI or SIM number. Condition: For Value Added Services and APLMN Service Center Usage, either the IMSI or MSISDN might be supplied, although one of them must be supplied and, where available, the IMSI is preferred. For normal mobile calls, the SIM number is preferred. Derivation: Optional. From the GSM item served IMSI as defined in TS GSM 12.05. Defined in TS GSM 03.03.

Table 34–8 (Cont.) Associated GSM/Wireline Extension Record Fields

Name	Format	Description
DEVICE_NUMBER	X(24)	Identifies the equipment used by the subscriber during the call; for example, the International Mobile Equipment Identity number (IMEI). Condition: Not present where the terminal equipment is not involved in the call; for example, in forwarded call cases. It is not mandatory for the VPLMN to transfer this information. Derivation: Optional. From the GSM item IMEI as defined in TS GSM 12.05. Defined in TS GSM 03.03. **Note:** Even though the IMEI is 16 digits in length, the check digit is not transmitted.
A_NUMBER_USER	X(40)	The customer who owns the number from which the call was originated, for terminated calls. Not used for rating, but could be used on invoices. Condition: There is no calling number present where it is unavailable. Could be different from the A Number; for example, in case of VPN calls. For VPN calls, the A Number contains the party to be billed, and this field contains the user initiating the call. Values: See A_NUMBER. Derivation: Optional. From the GSM item Calling Number as defined in TS GSM 12.05. This item is of type Address String and is further expanded into the items type of number, numbering plan, and the number sent across the air-interface as defined in TS GSM 04.08 and 09.02 or in international notation. Set by the first processor and left unchanged.
DIALED_DIGITS	X(40)	The number dialed by the customer when establishing a call, or the number to which the call is forwarded or transferred. Can be used for managing disputes. Condition: There might be no called number for the basic service emergency call but operators might optionally insert the digits 112 or their national emergency number into this field. The notation should always be local; for example, 04106768124. Values: See B_NUMBER. Derivation: Optional. From the GSM item Calling Number as defined in TS GSM 12.05. This item is of type Address String and is further expanded into the items type of number, numbering plan, and the number sent across the air-interface as defined in TS GSM 04.08 and 09.02 or in international notation. Set by the first processor and left unchanged.
BASIC_DUAL_ SERVICE	X(3)	A dual service can be used in context with twin or duo cards.

Table 34–8 (Cont.) Associated GSM/Wireline Extension Record Fields

Name	Format	Description
VAS/PRODUCT_ CODE	X(10)	A classification of Value Added Services as generated by the sender. Can be used to map to a specific internal service code to implement specific usage scenarios for any rating purposes. Values: VMAIL: Voice Mail Services SEC: Secretarial Services OPER: Telephonic Operator Services FI: Financial Information TRAVEL: Travel Information This is not a definitive list and might be added to through MoU-TADIG from time to time or might be user defined. Derivation: Optional. From the GSM item vasCode as defined in GSM TD17. Set by the first processor and left unchanged.
ORIGINATING_ SWITCH_ IDENTIFICATION	X(15)	Identifies the MSC or SwitchID handling the origin of the call. In case of mobile roaming calls (GSM), this field contains the MOC-related MCC/MNC. In case of wireline networks, this field contains the primary switch that generated this CDR. Can be used by any interconnect rating processor to uniquely identify the trunk names but will only be used if the trunk names are only unique within the related switch. See TRUNK_INPUT and TRUNK_OUTPUT. Can also be used to normalize the A Number for MOC roaming. In case of roaming, this field contains the MCC/MNC. Encoding: Encoded as one of the following according to the requirements of the sender: ■ The MSISDN of the MSC as per GSM 03.03; for example, 44836100456. ■ The signaling point code as per GSM 03.03; for example, 253464. ■ The MCC/MNC (TADIG, PLMN) for mobile roaming calls; for example, 26201. ■ MCC = Mobile Country Code, MNC = Mobile Network Code ■ A name; for example, "HELSINKI": Must be uppercase. ■ A switchID as set up within a local fixed network structure. Derivation: Optional, only mandatory in case of interconnect records (for intercarrier rating/billing reasons,) or for mobile (roaming) records. **Note:** The switch might not be needed if the trunk names are unique within the total network). Set by the first processor and left unchanged.

Table 34–8 (Cont.) Associated GSM/Wireline Extension Record Fields

Name	Format	Description
TERMINATING_ SWITCH_ IDENTIFICATION	X(15)	Identifies the MSC or Switch ID handling the termination of the call. In case of mobile roaming calls (GSM), this field contains the MTC-related MCC/MNC. In case of wireline networks, this field contains the secondary switch or is empty. Can be used to normalize the B Number in case of MTC-roaming cause. In case of roaming, this field contains the MCC/MNC. Encoding: Encoded as one of the following according to the requirements of the sender: ■ The MSISDN of the MSC as per GSM 03.03; for example, 44836100456. ■ The signaling point code as per GSM 03.03; for example, 253464. ■ The MCC/MNC (TADIG, PLMN) for mobile roaming calls; for example, 26201 ■ MCC = Mobile Country Code; MNC = Mobile Network Code ■ A name; for example, "HELSINKI": Must be uppercase. ■ A switch ID as set up within a local fixed network structure. Derivation: Optional, only mandatory in case mobile (roaming) records. Set by the first processor and left unchanged.
TRUNK_INPUT	X(15)	Trunk identification, inroute address in network switches. Used for interconnect rating to identify the inroute leg of a call. The inroute leg references a related network operator from which the call was received and how to treat this inroute leg in case of intercarrier rating. Encoding: Must uniquely identify a bundled line trunk: ■ Within the given ORIGINATING_SWITCH_IDENTIFICATION. ■ With the global network structure. Derivation: Optional, only mandatory in case of interconnect records (for intercarrier rating/billing reasons). Set by the first processor and left unchanged.
TRUNK_OUTPUT	X(15)	Trunk identification, outroute address in network switches. Can be used by any interconnect rating processor to identify the outroute leg of a call. The outroute leg references a related network operator to which the call was routed or terminated and how to treat this outroute leg in case of intercarrier rating. Encoding: Must uniquely identify a bundled line trunk: ■ Within the given TERMINATING_SWITCH_IDENTIFICATION. ■ With the global network structure. Derivation: Optional, only mandatory in case of interconnect records. Set by the first processor and left unchanged.

Table 34–8 (Cont.) Associated GSM/Wireline Extension Record Fields

Name	Format	Description
LOCATION_AREA_INDICATOR	X(10)	Identifies the MSC responsible for handling the call and the location of the equipment making or receiving the call. The definition of these items can be found in the Data Dictionary under MSC Identification, Location Area, and Cell Id. Can be used to map to a specific internal service code to implement a event-dependent rating. Condition: Is not available if not supported by the network or the call does not terminate at the equipment; for example, in call forwarding cases. Values: The Location Area Code is a two-octet string as defined in TS GSM 04.08. For the TAP, the octets are converted to a decimal number in the range 0x00000000 to 0xFFFFFFFF. Derivation: Optional. From the GSM item locationAreaCode as defined in TS GSM 12.05 or directly taken from the sender (VAS). Set by the first processor and left unchanged.
CELL_ID	X(10)	The cell from which the call originated. Can be used to identify the location of the caller. Condition: Operators might not transfer the cell identity. Only available if the call originates or terminates from a mobile phone; for example, not available in call divert cases. Values: The cell identity is a two-octet string as defined in TS GSM 04.08. However, an original hex value is copied. Derivation: Optional. From the GSM item Cell Id as defined in TS GSM 12.05. Set by the first processor and left unchanged.
MS_CLASS_MARK	9(1)	The power capability of the equipment making or receiving the call. Mobiles and transmobiles usually have class 2 capability, handhelds class 4, and PCN applications class 5. Some transmobiles have reduced capability and are classified as class 3. Usually not used. Condition: Only available if supported by the network and the call originates or terminates from the equipment. Is not available in call forwarding cases. Values: **1.** Class Mark 2 **2.** Class Mark 3 **3.** Class Mark 4 **4.** Class Mark 5 Other values might apply according to the related original input format. Derivation: Optional. From the GSM item msclassmark as defined in TS GSM 12.05. Set by the first processor and left unchanged.

Table 34–8 (Cont.) Associated GSM/Wireline Extension Record Fields

Name	Format	Description
TIME_BEFORE_ ANSWER	9(5)	The number of seconds until a call was successfully established, defined by the time between the call setup attempt and call answer. Can be used as a QoS parameter. Values: Minimum: 00000 Maximum: 99999 Derivation: Optional. Set by the first processor and left unchanged.
BASIC_AoC_ AMOUNT_VALUE	9(11)	A monetary amount assigned to the event by any rating processor and charged to the recipient of the file. This does not include any surcharges. Used for roaming or interconnect rating. Can be used to keep the original purchase charge, to evaluate a record based margin together with the charged amount value. Values: Space: No price given, like NULL in a database Floating point format: Given value, might be 0.000. If no floating point exists, the last three digits are always taken as decimals) Minimum: -9999999999 Maximum: 99999999999 Example: '00000012500' for 12,50 '-0001200100' for -1.200,10 '00000012.50' for 12,50 '00012.50000' for 12,50 Derivation: Optional. Usually handed over by the sender of the file, but might also be recalculated by any processor to represent the purchase charge.
BASIC_AoC_ AMOUNT_ CURRENCY	X(3)	See RETAIL_CHARGED_AMOUNT_CURRENCY.

Table 34–8 (Cont.) Associated GSM/Wireline Extension Record Fields

Name	Format	Description
ROAMER_AoC_ AMOUNT_VALUE	9(11)	A monetary amount assigned to the event by any rating processor and charged to the recipient of the file. This is typically a special add-on or surcharge. **Note:** The total wholesale charge of a roaming event should be calculated as: Basic AoC Amount + Roamer AoC Amount Used for roaming and interconnect rating. Can be used to keep the original purchase charge, to evaluate a record based margin together with the charged amount value. Values: Space: No price given, like NULL in a database Floating point format: Given value, might be 0.000. If no floating point exists the last 3 digits are always taken as decimals) Minimum: -999999999 Maximum: 99999999999 Example: '00000012500' for 12,50 '-0001200100' for -1.200,10 '00000012.50' for 12,50 '00012.50000' for 12,50 Derivation: Optional. Usually handed over by the sender (origin network operator) of the file, but might also be recalculated by any processor to represent the purchase charge.
ROAMER_AoC_ AMOUNT_ CURRENCY	X(3)	See RETAIL_CHARGED_AMOUNT_CURRENCY.
NUMBER_OF_ SUPPLEMENTARY_ SERVICE_PACKETS	9(2)	Defines the number of Supplementary Service Records following these base fields. For example, 05 means that 5 records are following. Can be used to evaluate how the record structure continues. Values: 00 - 99: Either zero or N records are following Derivation: Mandatory. Dependent on the input how many supplementary service records are present.
NUMBER_OF_BS_ PACKETS	Integer	Defines the number of Basic Service Records following the Supplementary Service Record. Values: Default = 0 Derivation: Mandatory. Dependent on the number of basic service records present.
SERVING_ NETWORK	String	Conditional in TAP 3.10 files. Indicates the network in which the call event was originally created. This field is a unique identifier.

Table 34–8 (Cont.) Associated GSM/Wireline Extension Record Fields

Name	Format	Description
B_CELL_ID	X(10)	Cell ID of the B party receiving the call. Derivation: Optional.
A_TERM_CELL_ID	X(10)	Cell ID of the A party when the call terminated. Derivation: Optional.
CALL_REFERENCE	String	CallReference item. Optional

Supplementary Service Event Record (RECType 520)

This optional record is used for all non-call related supplementary service actions. The information attributable to a supplementary service event includes basic event information, location information, equipment information, and details of the supplementary service used.

The record applies only to mobile calls (GSM). Derived from the GSM item parameters as defined in TS GSM 12.05.

Table 34–9 describes the fields in the Supplementary Service Event Record.

Table 34–9 Supplementary Service Event Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long. Value: 620 GSM/Wireline Extension Record Derivation: Mandatory. Set by the first processor and left unchanged.
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Values: Minimum: 000000002 Maximum: 999999998 Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
ACTION_CODE	H(1)	Qualifies the way in which the supplementary service is used. Values: 0: Registration 1: Erasure 2: Activation 3: Deactivation 4: Interrogation 5: Invocation 6: Registration of Password 9: Switch related Other values might apply according to the related original input format.

Table 34–9 (Cont.) Supplementary Service Event Record Fields

Name	Format	Description
SS_EVENT	H(2)	Uniquely defines the supplementary service or a group of supplementary services. Values: 00: All supplementary services 10: All line identification services 11: Calling number identification presentation 12: Calling number identification restriction 13: Connected number identification presentation 14: Connected number identification restriction 15: Malicious Call Identification 20: all call forwarding 21: Call forwarding unconditional 28: All conditional Call Forwarding 29: Call forwarding on mobile subscriber busy 2A: Call forwarding on no reply 2B: Call forwarding on subscriber not reachable 30: All call offering services 31: Call transfer 32: Mobile Access Hunting 40: all call completion services 41: Call waiting 42: Call hold 43: Completion of calls to busy subscribers 50: All multiparty services 51: multiparty service 60: All community of interest services 61: closed user groups 70: all charging supplement services 71: Advice of charge (charging) 72: Advice of charge (information) 80: All additional info transfer services 81: User to user signaling 90: All call barring 91: All Barring of outgoing Call Services 92: Barring of all outgoing calls 93: Barring of all outgoing international calls 94: Barring of all OG international except HPLMN 99: All Barring of incoming Call Services

Table 34–9 (Cont.) Supplementary Service Event Record Fields

Name	Format	Description
SS_EVENT (contd)	N/A	Uniquely defines the supplementary service or a group of supplementary services (contd). 9A: Barring of all incoming calls 9B: Barring of all IC calls when outside HPLMN all Switch related Services: 01...59: see related switch documentation (values used 1:1) Other values might apply according to the related original input format.
SS_PARAMETERS	String	Optional.
THIRD_PARTY_ NUMBER	String	Optional.
CLIR_INDICATOR	Integer	Optional.
CHARGING_START_ TIMESTAMP	Date	Optional.
CHARGING_END_ TIMESTAMP	Date	Optional.
UTC_END_TIME_ OFFSET	X(5)	Timezone where the call terminated. Derivation: Optional.
BASIC_SERVICE_ CODE_LIST	String	Optional.

Associated Roaming Extension Record

Table 34–10 lists the fields in the Associated Roaming Extension Record.

Table 34–10 Associated Roaming Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	None
RECORD_NUMBER	Integer	None
TAP_FILE_SEQ_NO	Integer	None
RAP_FILE_SEQ_NO	Integer	None
RAP_RECORD_TYPE	String	None
SENDER	String	None
RECIPIENT	String	None
TAP_FILE_PATH	String	None
START_MISSING_SEQ_NUM	Integer	None
END_MISSING_SEQ_NUM	Integer	None
SUSPENSION_TIME	Date	None
PORT_NUMBER	String	None
TOTAL_TAX_REFUND	Decimal	None
TOTAL_DISCOUNT_REFUND	Decimal	None
GUARANTEED_BIT_RATE	String	None

Table 34–10 (Cont.) Associated Roaming Extension Record Fields

Name	Format	Description
MAXIMUM_BIT_RATE	String	None
HSCSD_INDICATO	String	None
SMS_ORIGINATOR	String	None
SMS_DESTINATION_NUMBER	String	None
DISCOUNTABLE_AMOUNT	Decimal	None
DISCOUNT_CODE	Integer	None
NETWORKACCESS_IDENTIFIE	String	None
ISM_SIGNALLING_CONTEXT	Integer	None
IMSI	String	None
HOME_BID	String	None
HOMELOCATION_DESCRIPTION	String	None
MOBILE_ID_NUMBER	String	None
MOBILE_DIR_NUMBER	String	None
TOTAL_ADVISEDCHARGE	Decimal	None
TOTAL_ADVISEDCHARGE_REFUND	Decimal	None
TOTAL_COMMISSION	Decimal	None
TOTAL_COMMISSION_REFUND	Decimal	None
ITEM_OFFSET	Integer	None
ERROR_CODE	Integer	None
TOTAL_SEVERE_RETURN_VALUE	Decimal	None
RETURN_DETAILS_COUNT	Integer	None
CLIR_INDICATOR	String	None

Associated RAP Extension Record

Table 34–11 lists the fields in the Associated RAP Extension Record.

Table 34–11 Associated RAP Extension Record Fields

Name	Format	Description
PATH_ITEMID	Integer	None
ITEM_OCCURRENCE	Integer	None
ITEM_LEVEL	Integer	None

Basic Service Event Record (RECType 520)

This optional record is used to store related TAP data.

The record applies only to mobile calls (GSM). Derived from the GSM item parameters as defined in TS GSM 12.05.

Table 34–12 lists the fields in the Basic Service Event Record.

Table 34–12 Basic Service Event Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long. Value: 520 GSM/Wireline Extension Record Derivation: Mandatory. Set by the first processor and left unchanged.
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Values: Minimum: 000000002 Maximum: 999999998 Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
CHAIN_REFERENCE	String	Mandatory.
LONG_DURATION_INDICATOR	String	Mandatory.
BASIC_SERVICE	String	None
QOS_REQUESTED	String	None
QOS_USED	String	None
CHARGING_START_TIMESTAMP	Date	None
CHARGING_END_TIMESTAMP	Date	None
UTC_TIME_OFFSET	String	None
NUMBER_OF_UNITS	Decimal	None
WHOLESALE_CHARGED_AMOUNT_VALUE	Decimal	None
WHOLESALE_CHARGED_TAX_RATE	Integer	None
WHOLESALE_CHARGED_TAX_VALUE	Decimal	None
SPEECH_VERSION_REQUESTED	String	None
SPEECH_VERSION_USED	String	None
TRANSPARENCY_INDICATOR	String	None
FNUR	String	None
AIUR_REQUESTED	String	None
USER_PROTOCOL_INDICATOR	Integer	None
DATA_VOLUME_REFERENCE	String	None

Most-Called Information

This block contains the aggregated amount, duration, and occurrences of the most-called numbers. The number will be listed in the LIST attribute. The values listed

in Table 34–13 can be used in EVAL expressions to give discounts based on most-called numbers.

Table 34–13 Most-Called Information Fields

Name	Format	Description
AMOUNT	Decimal	Aggregated amount.
COUNT	Decimal	Aggregated occurrences.
LIST	Integer	Number.
QUANTITY	String	Aggregated duration.

HSCSD Information Packet Record

This optional record is used to store related TAP data.

The record applies only to mobile calls (GSM). Derived from the GSM item parameters as defined in TS GSM 12.05.

High Speed Circuit Switched Data allows users subscribing to the General Bearer Service to use higher transmission rates by using multiple traffic channels simultaneously. This group element must contain Basic HSCSD parameters as at call setup and may also contain details of changes to those parameters.

Table 34–14 lists the fields in the HSCSD Information Packet Record.

Table 34–14 HSCSD Information Packet Record

Name	Format	Description
NUMBER_OF_CHANNELS	String	NumberOfChannels item. Mandatory.
CHANNEL_CODING_OK_LIST	Integer	ChannelCodingAcceptable list (comma-separated integers). Mandatory.
CHANNEL_CODING_USED	Integer	ChannelCoding item. Mandatory.
NUMBER_OF_CHANNELS_ USED	Integer	NumberOfChannelsUsed item. Mandatory.
PM_LIST	Block	Optional. HSCSDParameterModification list.
AIUR	Integer	AiurRequested item.
MAX_NUMBER_OF_CHANNELS	Integer	NumberOfChannels item. Optional.
CHANNEL_CODING_USED	Integer	ChannelCoding item. Mandatory.
NUMBER_OF_CHANNELS_ USED	Integer	NumberOfChannelsUsed item. Mandatory.

Table 34–14 (Cont.) HSCSD Information Packet Record

Name	Format	Description
INITIATING_PARTY	Integer	InitiatingParty item. Mandatory.
MODIFICATION_TIMESTAMP_	Date	ModificationTimestamp item. Mandatory.
UTC_TIME_OFFSET	String	NumberOfChannels item.

Associated GPRS Extension Record (RECType 540)

This record stores GPRS service information. This record is optional and will be generated only if the related Basic Detail Record indicates a GPRS service.

Table 34–15 describes the fields in the Associated GPRS Extension Record.

Table 34–15 Associated GPRS Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long. Value: 540 Associated GPRS Extension Record
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
PORT_NUMBER	X(24)	Identifies the customer IMSI or SIM number. Option: For Value Added Services and APLMN Service Center Usage, either the IMSI or MSISDN might be supplied, although one of them must be supplied and, where available, the IMSI is preferred. For normal mobile calls, the SIM number is preferred. Derivation: Optional. From the GSM item served IMSI as defined in TS GSM 12.05. Defined in TS GSM 03.03.
DEVICE_NUMBER	X(24)	Identifies the equipment used by the subscriber during the call; for example, the International Mobile Equipment Identity number (IMEI). Condition: Not present where the terminal equipment is not involved in the call; for example, in forwarded call cases. It is not mandatory for the VPLMN to transfer this information. Derivation: Optional. From the GSM item IMEI as defined in TS GSM 12.05. Defined in TS GSM 03.03. **Note:** Even though the IMEI is 16 digits in length, the check digit is not transmitted.

Table 34–15 (Cont.) Associated GPRS Extension Record Fields

Name	Format	Description
A_NUMBER_USER	X(40)	The customer who owns the number from which the call was originated, for terminated calls. Not used for rating, but could be used on invoices. Condition: There is no calling number present where it is unavailable. Could be different from the A Number; for example, in case of VPN calls. For VPN calls, the A Number contains the party to be billed, and this field contains the user initiating the call. Values: See A_NUMBER. Derivation: Optional. From the GSM item Calling Number as defined in TS GSM 12.05. This item is of type Address String and is further expanded into the items type of number, numbering plan and the number sent across the air-interface as defined in TS GSM 04.08 and 09.02 or in international notation. Set by the first processor and left unchanged.
DIALED_DIGITS	X(40)	The number dialed by the customer when establishing a call or the number to which the call is forwarded or transferred. Can be used for managing disputes. Condition: There might be no called number for the basic service emergency call but operators might optionally insert the digits 112 or their national emergency number into this field. The notation should always be local; for example, 04106768124. Values: See B_NUMBER. Derivation: Optional. From the GSM item Calling Number as defined in TS GSM 12.05. This item is of type Address String and is further expanded into the items type of number, numbering plan and the number sent across the air-interface as defined in TS GSM 04.08 and 09.02 or in international notation. Set by the first processor and left unchanged.
VAS/PRODUCT_ CODE	X(10)	A classification of Value Added Services as generated by the sender. Can be used to map to a specific internal service code to implement specific usage scenarios for any rating purposes. Values: VMAIL: Voice Mail Services SEC: Secretarial Services OPER: Telephonic Operator Services FI: Financial Information TRAVEL: Travel Information This is not a definitive list and might be added to through MoU-TADIG from time to time or might be user defined. Derivation: Optional. From the GSM item vasCode as defined in GSM TD17. Set by the first processor and left unchanged.

Table 34–15 (Cont.) Associated GPRS Extension Record Fields

Name	Format	Description
ORIGINATING_ SWITCH_ IDENTIFICATION	X(15)	Identifies the MSC or SwitchID handling the origin of the call. In case of mobile roaming calls (GSM), this field contains the MOC-related MCC/MNC. In case of wireline networks, this field contains the primary switch that generated this CDR. Can be used by any interconnect rating processor to uniquely identify the trunk names but will only be used if the trunk names are only unique within the related switch. See TRUNK_INPUT and TRUNK_OUTPUT. Can also be used to normalize the A Number for MOC roaming. In case of roaming, this field contains the MCC/MNC. Encoding: Encoded as one of the following according to the requirements of the sender: ■ The MSISDN of the MSC as per GSM 03.03; for example, 44836100456. ■ The signaling point code as per GSM 03.03; for example, 253464. ■ The MCC/MNC (TADIG, PLMN) for mobile roaming calls; for example, 26201. ■ MCC = Mobile Country Code; MNC = Mobile Network Code ■ A name; for example, "HELSINKI": Must be uppercase. ■ A switchID as set up within a local fixed network structure. Derivation: Optional, only mandatory in case of interconnect records (for intercarrier rating/billing reasons,) or for mobile (roaming) records. Set by the first processor and left unchanged. The switch might not be needed if the trunk names are unique within the total network).
TERMINATING_ SWITCH_ IDENTIFICATION	X(15)	Identifies the MSC or Switch ID handling the termination of the call. In case of mobile roaming calls (GSM), this field contains the MTC-related MCC/MNC. In case of wireline networks, this field contains the secondary switch or is empty. Can be used to normalize the B Number in case of MTC-roaming cause. In case of roaming, this field contains the MCC/MNC. Encoding: Encoded as one of the following according to the requirements of the sender: ■ The MSISDN of the MSC as per GSM 03.03; for example, 44836100456. ■ The signaling point code as per GSM 03.03; for example, 253464. ■ The MCC/MNC (TADIG, PLMN) for mobile roaming calls; for example, 26201. ■ MCC = Mobile Country Code; MNC = Mobile Network Code ■ A name; for example, "HELSINKI": Must be uppercase. ■ A switch ID as set up within a local fixed network structure. Derivation: Optional, only mandatory in case of mobile (roaming) records. Set by the first processor and left unchanged.

Table 34–15 (Cont.) Associated GPRS Extension Record Fields

Name	Format	Description
MS_CLASS_MARK	9(1)	The power capability of the equipment making or receiving the call. Mobiles and transmobiles usually have class 2 capability, handhelds class 4, and PCN applications class 5. Some transmobiles have reduced capability and are classified as class 3. Usually not used. Condition: Only available if supported by the network and the call originates or terminates from the equipment. Is not available in call forwarding cases. Values: 1. Class Mark 2 2. Class Mark 3 3. Class Mark 4 4. Class Mark 5 Other values might apply according to the related original input format. Derivation: Optional. From the GSM item msclassmark as defined in TS GSM 12.05. Set by the first processor and left unchanged.
ROUTING_AREA	X(10)	Routing Area at the time of record creation (S-CDR only).
LOCATION_AREA_ INDICATOR	X(10)	Identifies the MSC responsible for handling the call and the location of the equipment making or receiving the call. The definition of these items can be found in the Data Dictionary under MSC Identification, Location Area, and Cell Id. Can be used to map to a specific internal service code to implement a event-dependent rating. Condition: Is not available if not supported by the network or the call does not terminate at the equipment; for example, in call forwarding cases. Values: The Location Area Code is a two-octet string as defined in TS GSM 04.08. For the TAP, the octets are converted to a decimal number in the range 0x00000000 to 0xFFFFFFFF. Derivation: Optional. From the GSM item locationAreaCode as defined in TS GSM 12.05 or directly taken from the sender (VAS). Set by the first processor and left unchanged.
CHARGING_ID	Decimal	PDP context identifier used to identify this PDP context in different records created by GSNs. This field is a charging identifier which can be used together with GGSN address to identify all records produced in SGSN(s) and GGSN involved in a single PDP context. Charging ID is generated by GGSN at PDP context activation and transferred to context requesting SGSN. At inter-SGSN routing area update, charging ID is transferred to the new SGSN as part of each active PDP context. Different GGSNs allocate the charging ID independently of each other and might allocate the same numbers. The CGF and/or BS might check the uniqueness of each charging ID together with the GGSN address and optionally (if still unambiguous) with the record opening timestamp.
SGSN_ADDRESS	X(64)	Current SGSN Address used. Optional.

Table 34–15 (Cont.) Associated GPRS Extension Record Fields

Name	Format	Description
GGSN_ADDRESS	X(64)	IP Address of the GGSN currently used. Optional.
WLAN_ADDRESS	String	Optional.
APN_ADDRESS	X(64)	The logical name of the connected access point to the external packet data network. APN comprises network identifier and operator identifier. This field contains the logical Access Point Name used to determine the actual connected access point. APN comprises network identifier and operator identifier. APN can also be a wildcard, in which case SGSN selects the access point address. See GSM 03.03 [4] and GSM 03.60 [8] for more information about APN format and access point decision rules.
NODE_ID	X(64)	Name of the recording entity; for example, could be the charging gateway name.
TRANS_ID	9(10)	Sequence number which the recording entity generates (NODE_ID). The number is allocated sequentially including all CDR types. It links together the CDR of a same recording entity.
SUB_TRANS_ID	9(10)	Partial record sequence number. This field contains a running sequence number which links the partial records generated for a PDP context/GPRS session. It can be used in post-processing to detect missing CDRs for a GPRS session. It links together the CDRs/events of a same session.
NETWORK_ INITIATED_PDP	9(1)	Network Initiated PDP context. The network initiates a context when it calls an ME. Values: 0: False 1: True
PDP_TYPE	X(4)	Defines the PDP type; for example, X.25, IP, PPP, or IHOSS:OSP (see GSM 09.60 for exact format).
PDP_ADDRESS	X(64)	PDP address of the served IMSI (Ipv4, IPv6, X.121).
PDP_REMOTE_ ADDRESS	X(255)	List of PDP address of remote host (comma-separated value, G-CDR only, X25 only).
PDP_DYNAMIC_ ADDRESS	9(1)	Indicates that the PDP address has been dynamically allocated for that particular PDP context. This field is missing if address is static; for example, part of PDP context subscription. Dynamic address allocation might be relevant for charging; for example, the duration of PDP context as one resource offered and possibly owned by network operator. Values: 0: False 1: True
DIAGNOSTICS	X(255)	Includes a more detailed technical reason for the release of the connection and might contain one of the following: ■ A MAP error from GSM 09.02 [17] ■ A Cause from GSM 04.08 [16] The diagnostics might also be extended to include manufacturer and network-specific information.

Table 34–15 (Cont.) Associated GPRS Extension Record Fields

Name	Format	Description
CELL_ID	X(10)	The cell from which the call originated. Can be used to identify the location of the caller. Condition: Operators might not transfer the cell identity. Only available if the call originates or terminates from a mobile phone; for example, not available in call divert cases. Values: The cell identity is a two-octet string as defined in TS GSM 04.08. However, an original hex value is copied. Derivation: Optional. From the GSM item Cell Id as defined in TS GSM 12.05. Set by the first processor and left unchanged.
CHANGE_ CONDITION	9(1)	The condition that triggers the creation of this volume container as defined by ETSI. Values: 0: Quality of Service Change 1: Tariff Change 2: Record Closed
QoS_REQUESTED_ PRECEDENCE	X(1)	The priority applicable to a GPRS connection. Condition: Mandatory within groups GSM Quality Of Service Requested. Values: 0: Unspecified 1: High Priority 2: Normal Priority 3: Low Priority Derivation: GSM item QoS Precedence (GSM 12.15).
QoS_REQUESTED_ DELAY	X(1)	The transfer delay applicable to a GPRS connection. Condition: Mandatory within groups GSM Quality Of Service Requested. Values: 0: Delay class 1 1: Delay class 2 2: Delay class 3 3: Delay class 4 Derivation: GSM item QoSDelay (GSM 12.15).

Table 34–15 (Cont.) Associated GPRS Extension Record Fields

Name	Format	Description
QoS_REQUESTED_ RELIABILTY	X(1)	The reliability applicable to a GPRS connection. Condition: Mandatory within groups GSM Quality Of Service Requested. Values: 0: Unspecified Reliability 1: Acknowledged GTP 2: Unacknowledged GTP/acknowledged LLC 3: Unacknowledged GTP/ acknowledged RLC 4: Unacknowledged GTP/LLC/RLC 5: Unacknowledged unprotected data Derivation: GSM item QoS Reliability (GSM 12.15).
QoS_REQUESTED_ PEAK_ THROUGHPUT	X(2)	The peak throughput applicable to a GPRS connection. Condition: Mandatory within groups GSM Quality Of Service Requested. Values: 0: Unspecified 1: Up to 100 octets per seconds 2: Up to 200 octets per seconds 3: Up to 400 octets per seconds 4: Up to 800 octets per seconds 5: Up to 1600 octets per seconds 6: Up to 3200 octets per seconds 7: Up to 6400 octets per seconds 8: Up to 12800 octets per seconds 9: Up to 25600 octets per seconds Derivation: GSM item QoS Peak Throughput (GSM 12.15).

Table 34–15 (Cont.) Associated GPRS Extension Record Fields

Name	Format	Description
QoS_REQUESTED_ MEAN_ THROUGHPUT	X(2)	The mean throughput applicable to a GPRS connection. Condition: Mandatory within groups GSM Quality Of Service Requested. Values: 0: Best Effort 1: Mean 100 octets per hour 2: Mean 200 octets per hour 3: Mean 500 octets per hour 4: Mean 1000 octets per hour 5: Mean 2000 octets per hour 6: Mean 5000 octets per hour 7: Mean 10000 octets per hour 8: Mean 20000 octets per hour 9: Mean 50000 octets per hour 10: Mean 100000 octets per hour 11: Mean 200000 octets per hour 12: Mean 500000 octets per hour 13: Mean 1000000 octets per hour 14: Mean 2000000 octets per hour 15: Mean 5000000 octets per hour 16: Mean 10000000 octets per hour 17: Mean 20000000 octets per hour 18: Mean 50000000 octets per hour Derivation: GSM item QoS Mean Throughput (GSM 12.15).
QoS_USED_ PRECEDENCE	X(1)	Quality of Service Precedence class. See QoS_REQUESTED_PRECEDENCE.
QoS_USED_DELAY	X(1)	QOS delay class, defined by ETSI. See QoS_REQUESTED_DELAY.
QoS_USED_ RELIABILTY	X(1)	QOS reliability class, defined by ETSI. See QoS_REQUESTED_RELIABILTY.
QoS_USED_PEAK_ THROUGHPUT	X(2)	QOS peak throughput class, defined by ETSI. See QoS_REQUESTED_PEAK_THROUGHPUT.
QoS_USED_MEAN_ THROUGHPUT	X(2)	QOS mean throughput class, defined by ETSI. See QoS_REQUESTED_MEAN_THROUGHPUT.
NETWORK_ CAPABILITY	X(10)	MS network capability information element of the served MS on PDP context activation or on GPRS attachment as defined in GSM 04.08 [16]. Condition: Optional. Derivation: GSM item network capability (GSM 04.08 [16]).

Table 34–15 (Cont.) Associated GPRS Extension Record Fields

Name	Format	Description
SGSN_CHANGE	9(1)	Indicates that this is the first record after an inter-SGSN routing area update. Condition: Mandatory. Values: 0: default, if this is not the 1st record 1: indicates the first record after an inter SGSN-change
START_SEQUENCE_NO	String	Optional.
END_SEQUENCE_NO	String	Optional.
B_CELL_ID	X(10)	Cell ID of the B party receiving the call. Derivation: Optional.
A_TERM_CELL_ID	X(10)	Cell ID of the A party when the call terminated. Derivation: Optional.
PDP_CONTEXT_START_TIMESTAMP	Date	Conditional in TAP 3.10. Indicates the start time of the PDP context when the Call Event Details (GPRS Call) represents an intermediate or last partial of a PDP context. Used in TAP files. Format: *CCYYMMDDHHMMSS*
PDP_UTC_TIME_OFFSET	String	Conditional in TAP 3.10. Indicates the UTC time offset for PDP_CONTEXT_START_TIMESTAMP.
SERVICE_USED_CHARGING_START_TIMESTAMP	Date	Conditional in TAP 3.10. Indicates the start time for charging GPRS calls. This field is present when the value is not the same as the associated Call Event Start Timestamp field (DETAIL.ASS_GPRS_EXT.GS_PACKET.CHARGING_START_TIMESTAMP). Used in TAP files. Format: *CCYYMMDDHHMMSS*
SERVICE_USED_UTC_TIME_OFFSET	String	Conditional in TAP 3.10. Indicates the UTC time offset for SERVICE_USED_CHARGING_START_TIMESTAMP.
TYPE_OF_CONTROLLING_NODE	Integer	Conditional in TAP 3.10.
GPRS_SERVICE_USAGE_PACKET	Block	*n* times. Optional. Mandatory.
CHARGING_START_TIMESTAMP	Date	Optional.

Table 34–15 (Cont.) Associated GPRS Extension Record Fields

Name	Format	Description
CHARGING_END_ TIMESTAMP	Date	Optional.
UTC_TIME_OFFSET	String	Optional.
QOS_REQUESTED_ PRECEDENCE	String	Optional.
QOS_REQUESTED_ DELAY	String	Optional.
QOS_REQUESTED_ RELIABILITY	String	Optional.
QOS_REQUESTED_ PEAK_ THROUGHPUT	String	Optional.
QOS_REQUESTED_ MEAN_ TRHOUGHPUT	String	Optional.
QOS_USED_ PRECEDENCE	String	Optional.
QOS_USED_DELAY	String	Optional.
QOS_USED_ RELIABILITY	String	Optional.
QOS_USED_PEAK_ TRHOUGHPUT	String	Optional.
QOS_USED_MEAN_ TRHOUGHPUT	String	Optional.
VOLUME_RECEIVED	Decimal	Mandatory.

Table 34–15 (Cont.) Associated GPRS Extension Record Fields

Name	Format	Description
VOLUME_SENT	Decimal	Mandatory.
UMTS_QOS_ REQUESTED	String	Optional. Identifies the UMTS Quality of Service requested for GPRS calls. Used in TAP files. Format: This field contains comma-separated tag-value pairs of the following TAP fields with their respective tags as shown below: ■ QoS Traffic Class: 1 ■ QoS Max Bit Rate Uplink: 2 ■ Qos Max Bit Rate Downlink: 3 ■ Qos Guaranteed Bit Rate Downlink: 4 ■ Qos Guaranteed Bit Rate Uplink: 5 ■ Qos Allocation Retention Priority: 6 The fields QoS Traffic Class, QoS Max Bit Rate Uplink, Qos Max Bit Rate Downlink are Mandatory. The others are optional. Example 1: If the TAP field values are as follows: ■ QoS Traffic Class: **3** ■ QoS Max Bit Rate Uplink: **63** ■ Qos Max Bit Rate Downlink: **128** ■ Qos Guaranteed Bit Rate Downlink: **61** ■ Qos Guaranteed Bit Rate Uplink: **250** ■ Qos Allocation Retention Priority: **3** The value of the EDR field is **1,3,2,63,3,128,4,61,5,250,6,3** Example 2: If the TAP field values are as follows: QoS Traffic Class: **2** QoS Max Bit Rate Uplink: **56** Qos Max Bit Rate Downlink: **128** Qos Guaranteed Bit Rate Uplink: **250** The value of the EDR field is: **1,2,2,56,3,128,5,250**
UMTS_QOS_USED	String	Optional. Identifies the UMTS Quality of Service used for GPRS calls. Used in TAP files. The description for this field is identical to the description for UMTS_QOS_ REQUESTED.

Associated WAP Extension Record (RECType 570)

Stores information for WAP events. This record is optional and will only be generated if the related Basic Detail Record indicates a WAP service.

Table 34–16 describes the fields in the Associated WAP Extension Record.

Table 34–16 Associated WAP Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long. Value: 570 Associated WAP Extension Record
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
PORT_NUMBER	X(24)	Identifies the customer IMSI or SIM number. Option: For Value Added Services and APLMN Service Center Usage either the IMSI or MSISDN might be supplied, although one of them must be supplied and, where available, the IMSI is preferred. For normal mobile calls, the SIM number is preferred. Derivation: Optional. From the GSM item served IMSI as defined in TS GSM 12.05. Defined in TS GSM 03.03.
DEVICE_NUMBER	X(24)	Identifies the equipment used by the subscriber during the call; for example, the International Mobile Equipment Identity number (IMEI). Condition: Not present where the terminal equipment is not involved in the call; for example, in forwarded call cases. It is not mandatory for the VPLMN to transfer this information. Derivation: Optional. From the GSM item IMEI as defined in TS GSM 12.05. Defined in TS GSM 03.03. **Note:** Even though the IMEI is 16 digits in length, the check digit is not transmitted.
SESSION_ID	X(64)	Session ID as provided by the WAP gateway.
RECORDING_ENTITY	X(64)	Name of the recording Entity; for example, the WAP gateway or mediation device.
TERMINAL_CLIENT_ID	X(64)	The served WAP terminal client ID (WAP gateway user identity).
TERMINAL_IP_ADDRESS	X(64)	IP address of the WAP terminal.
DOMAIN_URL	X(255)	URL implementing the service.
BEARER_SERVICE	X(3)	See BASIC_SERVICE.
BEARER_SERVICE_CODE	X(2)	See SERVICE_CODE.

Table 34–16 (Cont.) Associated WAP Extension Record Fields

Name	Format	Description
HTTP_STATUS	9(3)	HTTP status code from the origin server or servlet. Values: 100: CONTINUE 101: SWITCHING_PROTOCOLS 200: SUCCESS 201: CREATE 202: ACCEPTED 203: NON-AUTHORITATIVE_INFORMATION 204: NO_CONTENT 205: RESET_CONTENT 206: PARTIAL_CONTENT 300: MULTIPLE_CHOICE 301: MOVED_PERMANENTLY 302: FOUND 303: SEE_OTHER 304: NOT_MODIFIED 305: USE_PROXY 307: TEMPORARY_REDIRECT 400: BAD_REQUEST 401: UNAUTHORIZED 402: PAYMENT_REQUIRED 403: FORBIDDEN 404: NOT_FOUND 405: METHOD_NOT_ALLOWED 406: NOT_ACCEPTABLE 407: PROXY_AUTHENTICATION_REQUIRED 408: REQUEST_TIMEOUT 409: CONFLICT 410: GONE 411: LENGTH_REQUIRED 412: PRECONDITION_FAILED 413: REQUEST_ENTITY_TOO_LARGE 414: REQUEST-URI_TOO_LONG 415: UNSUPPORTED_MEDIA_TYPE 416: REQUESTED_RANGE_NOT_SATISFIABLE 417: EXPECTATION_FAILED 500: INTERNAL_SERVER_ERROR 501: NOT_IMPLEMENTED 502: BAD_GATEWAY 503: SERVICE_UNAVAILABLE 504: GATEWAY_TIMEOUT 505: HTTP_VERSION_NOT_SUPPORTED

Table 34–16 (Cont.) Associated WAP Extension Record Fields

Name	Format	Description
WAP_STATUS	9(3)	The WSP/WAP status code. Values: 16: CONTINUE 17: SWITCHING_PROTOCOLS 20: OK, SUCCESS 33: CREATED 34: ACCEPTED 35: NON-AUTHORITATIVE_INFORMATION 36: NO_CONTENT 37: RESET_CONTENT 38: PARTIAL_CONTENT 48: MULTIPLE_CHOICE 49: MOVED_PERMANENTLY 50: MOVED_TEMPORARILY 51: SEE_OTHER 52: NOT_MODIFIED 53: USE_PROXY 55: TEMPORARY_REDIRECT 64: BAD_REQUEST 65: UNAUTHORIZED 66: PAYMENT_REQUIRED 67: FORBIDDEN 68: NOT_FOUND 69: METHOD_NOT_ALLOWED 70: NOT_ACCEPTABLE 71: PROXY_AUTHENTICATION_REQUIRED 72: REQUEST_TIMEOUT 73: CONFLICT 74: GONE 75: LENGTH_REQUIRED 76: PRECONDITION_FAILED 77: REQUEST_ENTITY_TOO_LARGE 78: REQUEST-URI_TOO_LONG 79: UNSUPPORTED_MEDIA_TYPE 80: REQUESTTED_RANGE_NOT_SATISFIABLE 81: EXPECTATION_FAILED 96: INTERNAL_SERVER_ERROR 97: NOT_IMPLEMENTED 98: BAD_GATEWAY 99: SERVICE_UNAVAILABLE 100: GATEWAY_TIMEOUT 101: HTTP_VERSION_NOT_SUPPORTED

Table 34–16 (Cont.) Associated WAP Extension Record Fields

Name	Format	Description
ACKNOWLEDGE_STATUS	9(1)	Acknowledge status of the response. Values: 1: OK acknowledgment has been received. 2: Response terminated by the server. 3: Response terminated by the terminal. 4: Acknowledgment has not been received. 5: Acknowledgment is not used with this connection type.
ACKNOWLEDGE_TIME	YYYYMMDDHHMISS	Time of the acknowledgment.
EVENT_NUMBER	X(60)	Assigned user event number as generated by the WAP gateway.
GGSN_ADDRESS	X(64)	IP Address of the GGSN currently used.
SERVER_TYPE	X(64)	A description of the type of server providing the service.
CHARGING_ID	Decimal	PDP context identifier used to identify this PDP context in different records created by GSNs. This field is a charging identifier which can be used together with GGSN address to identify all records produced in SGSN(s) and GGSN involved in a single PDP context. Charging ID is generated by GGSN at PDP context activation and transferred to context requesting SGSN. At inter-SGSN routing area update, charging ID is transferred to the new SGSN as part of each active PDP context. Different GGSNs allocate the charging ID independently of each other and might allocate the same numbers. The CGF and/or BS might check the uniqueness of each charging ID together with the GGSN address and optionally (if still unambiguous) with the record opening timestamp.
WAP_LOGIN	X(24)	Login used during the WAP session. This might occur in addition to the MSISDN; for example, this field might contain a user name of a session which has been opened within a WAP session. Condition: Optional. Might be mandatory for specific WAP scenarios. Derivation: Set by the first processor and left unchanged.
IDENTIFIER	String	Mandatory.
TYPE	Integer	Mandatory.

Associated CAMEL Extension Record (RECType 700)

In the following associated record of the sol42 format extended CAMEL service information could be stored. This record is optional and is attached to any other associated service extension record.

Table 34–17 lists the Associated CAMEL Extension Record fields.

Table 34–17 Associated CAMEL Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long Value: 700: Associated CAMEL Extension Record
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
SERVER_TYPE_OF_NUMBER	Z(1)	Optional, but should be defaulted to '0'. Could be used by some modules to perform number-normalization.
SERVER_NUMBERING_PLAN	X(1)	Optional.
SERVER_ADDRESS	X(40)	Identifies the server interrogated. Mandatory.
SERVICE_LEVEL	Z(1)	Identifies the level of CAMEL service provided [0-3]. Mandatory.
SERVICE_KEY	Z(10)	Identifies the CAMEL service Logic to be applied. Mandatory.
DEFAULT_CALL_HANDLING_INDICATOR	Z(1)	Indicates whether or not a CAMEL call encountered default handling. Values: 0: Continue the call 1: Release the call
MSC_TYPE_OF_NUMBER	Z(1)	Optional, but should be defaulted to '0'. Could be used by some modules to perform number-normalization.
MSC_NUMBERING_PLAN	X(1)	Optional.
MSC_ADDRESS	X(40)	Identifies the MSC that generated the CAMEL reference number (might be different from SERVER_ADDRESS). Mandatory.
CAMEL_REFERENCE_NUMBER	X(20)	In association with the MSC_ADDRESS, provides a unique identifier for each CAMEL invocation. Mandatory.
CAMEL_INITIATED_CF_INDICATOR	Z(1)	Optional, but should be defaulted to 0 (1=CAMEL call forwarding).
CAMEL_MODIFICATION_LIST	X(20)	Optional, comma-separated string of integers.
DEST_GSMW_TYPE_OF_NUMBER	Z(1)	Optional, but should be defaulted to '0'. Could be used by some modules to perform number-normalization.

Table 34–17 (Cont.) Associated CAMEL Extension Record Fields

Name	Format	Description
DEST_GSMW_NUMBERING_PLAN	X(1)	Optional.
DEST_GSMW_NUMBER	X(40)	Optional, used to identify CAMEL redirection destination (could contain an MSISDN, IP, LOGIN, etc.) when the primary extension is of type GSM (for example, ASS_GSMW_EXT).
DEST_GSMW_NUMBER_ORIGINAL	X(40)	Optional, DEST_GSM_NUMBER as received (before normalization).
DEST_GPRS_APN_ADDRESS	X(64)	Optional (but might be mandatory for specific zoning scenarios; for example, if an apn_group is used) when the primary extension is of type GPRS (ie., ASS_GPRS_EXT).
DEST_GPRS_PDP_REMOTE_ADDRESS	X(255)	Optional.
CSE_INFORMATION	X(40)	Optional, the information downloaded by the CAMEL server.
GSM_CALL_REFERENCE_NUMBER	X(20)	Optional.
EXCHANGE_RATE	Decimal	Contains the exchange rate which has been used to convert the Incoming currency to the internal currency as indicated in the field CHARGED_ CURRENCY_TYPE. Can be used to convert the virtual currency SDR (which is used in conjunction of TAP) to internal currencies and convert the Charge back to SDR after Rating. This would be a typical usage for Interconnection Rating. Values: Variable floating point format: Given value, might be 0.000. The floating decimal point must be set. Minimum: -9999999999 Maximum: 99999999999 Examples: '00000000125' for 125,00 '00000012.50' for 12,50 '-0012.56780' for -12,5678 Derivation: Optional, defaulted 00000000001 (=1,00).

Associated Suspense Extension Record (RECType 720)

Table 34–18 describes the fields in the Associated Suspense Extension Record. This record is optional and can appear once.

Table 34–18 Associated Suspense Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	Mandatory. Default = **720**
RECORD_NUMBER	Integer	Mandatory. Auto-generated.
SUSPENSE_STATUS	Integer	Mandatory. Calculated.

Table 34–18 (Cont.) Associated Suspense Extension Record Fields

Name	Format	Description
SUSPENSE_REASON	Integer	The suspense reason. Mapped from the error code. Mandatory. Calculated.
SUSPENSE_SUBREASON	Integer	Mandatory. Calculated.
RECYCLE_KEY	String	Search key for choosing EDRs to recycle. Optional.
ERROR_CODE	Integer	Mandatory. Calculated.
SUSPENSE_ID	Integer	Original suspense POID ID. Mandatory when recycling.
PIPELINE_NAME	String	The name of the pipeline, derived from the pipeline registry. Mandatory when recycling. Calculated.
SOURCE_FILENAME	String	The source file name. The same as INTERNAL.STREAM_NAME. Mandatory. Calculated.
SERVICE_CODE	String	Equal to DETAIL.INTERN_SERVICE_CODE. Mandatory. Calculated.
EDR_RECORD_TYPE	String	Equal to DETAIL.RECORD_TYPE. Mandatory. Calculated.
EDR_BUF	String	A stored representation of the EDR container including fields overwritten and enriched by the pipeline. Mandatory when recycling. Calculated.
UTC_OFFSET_SECONDS	Integer	This value enables Suspense Management Center to represent call record times using the same time zone used in the records themselves. The value represents the offset between the time zone of the EDR and UTC in seconds. Mandatory. Calculated.
EDR_SIZE	Integer	The size of DETAIL.ASS_SUSPENSE_EXT.EDR_BUF. Mandatory. Calculated.
QUERYABLE_FIELDS	String	The queryable field values defined in the registry. Separated by tab characters. Mandatory. Calculated.
OVERRIDE_REASONS	String	Optional. May be set equal to the override reason code during recycling.
ACCOUNT_POID	String	Optional. Calculated.
SUSPENDED_FROM_BATCH_ID	String	Optional. Calculated.
PIPELINE_CATEGORY	String	Mandatory. Calculated.
RECYCLING_MODE	Integer	Mandatory. Calculated. Equal to DETAIL.INTERN_PROCESS_STATUS

Associated Content Extension Record (RECType 550)

This optional record is used to store related TAP data. Table 34–19 describes the fields in the Associated Content Extension Record.

Table 34–19 Associated Content Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	Mandatory. Default = 550.
RECORD_NUMBER	Integer	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
FRAUD_MONITOR_INDICATOR	String	Optional, but should be defaulted to '0'. Could be used by some modules to perform number-normalization.
RAP_FILE_SEQ_NO	String	Optional.
ORDER_PLACED_TIMESTAMP	Date	Optional.
ORDER_PLACED_UTC_TIME_OFFSET	String	Mandatory if ORDER_PLACED_TIMESTAMP is present.
REQUESTED_DELIVERY_TIMESTAMP	Date	Optional.
REQ_DELIVERY_UTC_TIME_OFFSET	String	Mandatory if REQUESTED_DELIVERY_TIMESTAMP is present.
ACTUAL_DELIVERY_TIMESTAMP	Date	Optional.
ACT_DELIVERY_UTC_TIME_OFFSET	String	Mandatory if ACTUAL_DELIVERY_TIMESTAMP is present.
TOTAL_TRANSACTION_DURATION	Integer	Optional.
TRANSACTION_STATUS	Integer	Optional.
CHARGED_PARTY_INFO	Block	Charged party information block. Mandatory.
ID_LIST	N.A.	ChargedPartyId list. Mandatory.
HOMEID_LIST	N.A.	ChargedPartyHomeId list. Optional.
LOCATION_LIST	N.A.	ChargedPartyLocation list. Optional.
EQUIPMENT	N.A.	ChargedPartyEquipment block. Optional.
SERVING_PARTIES_INFO	Block	ServingPartiesInformation block. Mandatory.

Table 34–19 (Cont.) Associated Content Extension Record Fields

Name	Format	Description
PROVIDER_LIST	N.A.	ContentProviderId list. Optional.
ISP_LIST	N.A.	InternetServiceProviderId list. Optional.
NETWORK_LIST	N.A.	Network list. Optional.
SERVICE_USED_LIST	N.A.	ContentServiceUsed list. Mandatory.
CONTENT_TRANSACTION_CODE	Integer	ContentTransactionCode item. Mandatory.
OBJECT_TYPE	Integer	ObjectType item. Optional.
TRANSACTION_DESCRIPTION_SUPP	Integer	TransactionDescriptionSupp item. Optional.
TRANSACTION_DETAIL_DESCRIPTION	String	TransactionDetailDescription item. Optional.
TRANSACTION_IDENTIFIER	String	TransactionIdentifier item. Mandatory.
TRANSACTION_AUTH_CODE	String	TransactionAuthCode item. Optional.
DATA_VOLUME_INCOMING	Integer	DataVolumeIncoming item. Optional.
DATA_VOLUME_OUTGOING	Integer	DataVolumeOutgoing item. Optional.
TOTAL_DATA_VOLUME	Integer	TotalDataVolume item. Optional.
CHARGE_REFUND_INDICATOR	Integer	ChargeRefundIndicator item. Optional.
CONTENT_CHARGING_POINT	Integer	ContentChargingPoint item. Optional.
PAID_INDICATOR	Integer	PaidIndicator item. Optional.
PAYMENT_METHOD	Integer	PaymentMethod item. Optional.

Table 34–19 (Cont.) Associated Content Extension Record Fields

Name	Format	Description
ADVISED_CHARGE_CURRENCY	String	AdvisedChargeCurrency item. Optional.
ADVISED_CHARGE	Decimal	AdvisedCharge item. Optional. Mandatory in the AdvisedChargeInformation block.
COMMISSION	Decimal	Commission item. Optional.

Associated Location Extension Record

The OutGrammar stores information of Content and Location from the EDR container into the output TAP blocks. This is performed using ASN calls of iScript in TAP version 3.10 OutGrammar.

Table 34–20 describes the fields in the Associated Location Extension Record.

Table 34–20 Associated Location Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	Mandatory. Must be set to 560.
RECORD_NUMBER	Integer	Mandatory. Auto-generated.
FRAUD_MONITOR_INDICATOR	String	FraudMonitorIndicator item. Optional.
RAP_FILE_SEQ_NO	String	RapFileSequenceNumber item. Optional.
REC_ENTITY_CODE	Integer	RecEntityCode item. Mandatory.
CALL_REFERENCE	String	CallReference item. Optional.
GMLC_ADDRESS	String	N/A
TRACKING_CUSTOMER_INFORMATION	Block	TrackingCustomerInformation block. Optional.
ID_LIST	N.A.	TrackingCustomerId list. Mandatory.
HOME_ID_LIST	N.A.	TrackingCustomerHomeId list. Mandatory.
LOCATION_LIST	N.A.	TrackingCustomerLocation list. Mandatory.
EQUIPMENT	N.A.	TrackingCustomerEquipment block. Optional.
LCS_SP_INFORMATION LCSSP_INFO	Block	LCSSPInformation block. Optional.

Table 34–20 (Cont.) Associated Location Extension Record Fields

Name	Format	Description
ID_LIST	N.A.	LCSSPId list. Mandatory.
ISP_LIST	N.A.	InternetServiceProviderId list. Optional.
NETWORK_LIST	N.A.	Network list. Optional.
TRACKED_CUSTOMER_INFORMATION	Block	TrackedCustomerInformation block. Optional.
ID_LIST	N.A.	TrackedCustomerId list. Mandatory.
HOME_ID_LIST	N.A.	TrackedCustomerHomeId list. Mandatory.
LOCATION_LIST	N.A.	TrackedCustomerLocation list Mandatory.
EQUIPMENT	N.A.	TrackedCustomerEquipment block. Optional.
LOCATION_SERVICE_USAGE	Block	LocationServiceUsage block. Mandatory.
LCSQosRequested	Block	Mandatory.
LCS_REQUEST_TIMESTAMP	Date	Mandatory.
LCS_REQ_UTC_OFFSET	String	LCSRequestTimestamp item. Mandatory.
H_ACCURACY_REQUESTED	Integer	HorizontalAccuracyRequested item. Optional.
V_ACCURACY_REQUESTED	Integer	VerticalAccuracyRequested item. Optional.
RESPONSE_TIME_CATEGORY	Integer	ResponseTimeCategory item. Optional.
TRACKING_PERIOD	Integer	TrackingPeriod item. Optional.
REQ_TRACKING_FREQUENCY	Integer	TrackingFrequency (requested) item. Optional.
LCSQosDelivered	Block	Optional.
LCS_TRANS_STATUS	Integer	LCSTransactionStatus item. Optional.
H_ACCURACY_DELIVERED	Integer	HorizontalAccuracyDelivered item. Optional.
V_ACCURACY_DELIVERED	Integer	VerticalAccuracyDelivered item. Optional.

Table 34–20 (Cont.) Associated Location Extension Record Fields

Name	Format	Description
RESPONSE_TIME	Integer	ResponseTime item. Optional.
POSITIONING_METHOD	Integer	PositioningMethod item. Optional.
DEL_TRACKING_PERIOD	Integer	TrackingPeriod item. Optional.
DEL_TRACKING_FREQUENCY	Integer	TrackingFrequency (delivered) item. Optional.
AGE_OF_LOCATION	Integer	AgeOfLocation item. Optional.
CHARGING_TIMESTAMP	Date	ChargingTimeStamp item. Optional.
CHARGING_UTC_OFFSET	String	ChargeInformationList data is stored in DETAIL.ASSOCIATED_CHARGE_ BREAKDOWN.CHARGE_PACKET. Mandatory if LCSRequestTimestamp is given.

Associated Value Added Service (VAS) Extension Record (RECType 710)

A Value Added Service (VAS) item represents usage of value added services outside of a standard call; i.e., unrelated to either a Mobile Originated Call or a Mobile Terminated Call. VAS consists of Chargeable Subscriber and Value Added Service Used, which are mandatory; conditionally, Network Type and RAP File Sequence Number; optionally, Operator Specific Information.

Table 34–21 describes the fields in the Associated Value Added Service Extension Record.

Table 34–21 Associated Value Added Service (VAS) Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	Mandatory. Default = 710.
RECORD_NUMBER	Integer	Mandatory. Auto-generated.
VAS_CODE	Integer	Mandatory.
VAS_SHORT_DESC	String	Optional.
VAS_DESC	String	Optional.
CHARGING_START_TIMESTAMP	Date	Optional.
CHARGING_END_TIMESTAMP	Date	Optional.
UTC_TIME_OFFSET	String	Optional.

Associated BRM Balance Record (RECType 900)

Stores data to be loaded into the BRM database.

Associated BRM Billing Records might occur more than once for each Basic Detail Record. This is the case if more than one balance is affected by one event.

Table 34–22 describes the fields in the Associated BRM Balance Record.

Table 34–22 Associated BRM Balance Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long. Values: 900: Associated BRM Balance Record Usage: Determination of the different record types. Derivation: Mandatory. Set by the first processor and left unchanged.
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
ACCOUNT_POID	X(255)	POID of the account. Example: 1 /account 123456789 0 Derivation: Mandatory.
SERVICE_POID	X(255)	POID for the service. Example: 1 /service/ip/gprs 123456789 0 Derivation: Mandatory.
ITEM_POID	X(255)	POID of the item object affected due to this event. Applies only to the balance array element that impacts currency resources. This might be different from the PIN_FLD_ITEM_OBJ field in the base **/event** class. Example: 1 /item/misc 123456789 0 Derivation: Mandatory.
ORIGINAL_EVENT_POID	X(255)	POID of the original recorded event. Set only if the event has been extracted for pipeline rerating. Example: "1 /event/delayed 123456 0" Derivation: Optional.

Table 34–22 (Cont.) Associated BRM Balance Record Fields

Name	Format	Description
PIN_TAX_LOCALES	X(255)	Used for tax calculation. Values: "*order_origin* | *order_accept* | *ship_from* | *ship_to*" (Note that these fields are separated by pipes (|).) Each of these values (*order_origin*, *order_accept*, *ship_from*, *ship_to*) is an address in the following format: *city*;*zipcode*;*state*;*country*;[*geocode*,*location_mode*,*international_indicator*] **Note:** Be aware of the semicolon separators and enclosing brackets. For example, "cupertino;95014;CA;US;[5723121,2,0]"Derivation: Optional. *order_origin*, *order_accept*, and *ship_from* addresses are all the same and are derived from account profile object tax supplier information. *ship_to* is the address in the first element of the account's NAMEINFO array. *geocode* is either a geocode or NPA-NXX (the first 6 digits of the phone number). *location_mode* is 1 if it is a geocode and 2 if its NPA-NXX. *international_indicator* is 0 (US) or 1 (International). **Important:** This field might not be implemented in this release.
PIN_TAX_SUPPLIER_ID	X(255)	POID of the **/profile/tax_supplier** object used to tax this event. NULL if there is no tax supplier specified. Derivation: Optional. **Important:** This field might not be implemented in this release.

Table 34–22 (Cont.) Associated BRM Balance Record Fields

Name	Format	Description
PIN_PROVIDER_ID	X(255)	POID of the remittance service provider account. Example: 1 /account 123456789 0 Derivation: Optional. **Important:** This field might not be implemented in this release.
PIN_INVOICE_DATA	X(255)	Stores the data in the event that is include in the invoice. T The data is mapped to BRM fields and is stored as a string in the format: "@INTEGRATE#PIN_FLD_CALLING_NUMBER#PIN_FLD_CALLED_NUMBER#PIN_FLD_SVC_TYPE# PIN_FLD_SVC_CODE#PIN_FLD_NUMBER_OF_UNITS#PIN_FLD_USAGE_CLASS#PIN_FLD_DNIS#PIN_FLD_BAL_IMPACTS" Arrays follow this format: PIN_FLD_BAL_IMPACTS = *ID,PIN_FLD_RATE_TAG,PIN_FLD_QUANTITY*> For example: @INTEGRATE#004917165210#0049171235292#T1#11#0#USAGE#GSMThing#<1 /item 3456 1#1234,10.0#rateTag#1.000000#|1 /item 5678 1#6789#20.0#rateTag#1.000000>
NUMBER_OF_BALANCE_IMPACT_PACKETS	9(2)	Specifies the number of packets following these base fields (dynamic structure); for example, 03 means that 3 packets are following. Must be used to evaluate how the record structure continues. Values: 00 - 99: optionally, 0..n packets might follow Derivation: Mandatory.

Supplementary Balance Impact Packet Record (RECType 600)

The packets can optionally be used to store an event-related balance impact array. Within this structure, N-times PIN_BALANCE_IMPACTs are created, each containing one balance impact per RESOURCE_ID and optionally per GL_ID.

This record is used for evaluation event-related balance impacts together with the REL to map rating-internal Charge-Packets to BRM related balance impacts.

- **Condition**: Only relevant if present. If present, a mapping to all PIN-related values has to take place.
- **Derivation**: Optional. From the BRM object **/event/PIN_FLD_BAL_IMPACTS**. Will be optionally generated by a post-processor. If not present, the mapping will take place within the Rated Event (RE) Loader.

  Table 34–23 describes the Supplementary Balance Impact Packet Record fields.

Table 34–23 Supplementary Balance Impact Packet Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long. Value: 600
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
ACCOUNT_POID	String	POID of the account that the balance impact applies to. Derivation: Mandatory. Calculated.
BAL_GRP_POID	String	Balance group that the balance impact applies to. Derivation: Mandatory. Calculated.
OBJECT_CACHE_TYPE	Integer	Mandatory. Calculated.
ITEM_POID	String	POID of the item that the balance impact applies to. Derivation: Mandatory. Calculated.
PIN_RESOURCE_ID	9(9)	Numeric value of the resource that is impacted; for example, 840 for US dollars. Values: Any configured BRM resource ID. Derivation: Mandatory.
PIN_RESOURCE_ID_ORIG	Integer	Optional.
PIN_IMPACT_CATEGORY	X(255)	Name of the BRM impact category that was used to generate this balance impact for the rated event. Values: Any configured BRM impact category. Derivation: Mandatory.
PIN_IMPACT_TYPE	Integer	Mandatory. Calculated.
PIN_GL_ID	9(9)	GLID associated with this balance impact. Values: Any configured BRM general ledger ID. Derivation: Optional, default 0. Derived from IFW_RATEPLAN_CNF.GLACCOUNT. Might be mapped from the BRM object **/event/PIN_FLD_BAL_IMPACTS.PIN_FLD_GL_ID**.

Table 34–23 (Cont.) Supplementary Balance Impact Packet Record Fields

Name	Format	Description
RUM_ID	Integer	Mandatory. Calculated. Default = 0.
PIN_OFFERING_POID	String	Optional. Calculated.
PIN_TAX_CODE	X(255)	Tax code for the rate that was used. When taxes do not apply, this field is set to 0. Derivation: Optional. From IFW_RATEPLAN_CNF.GLACCOUNT->TAXCODE. Might be mapped from the BRM object **/event/PIN_FLD_BAL_IMPACTS.PIN_FLD_GL_ID**.
PIN_RATE_TAG	X(255)	Description of the rate used. Same as the PIN_FLD_DESCR in **/rate**. Can be used to more precisely describe the balance impact detail; for example, the following concatenated, comma-separated rating-related Charge-Packet values used: TIMEZONE, DAY_CODE, TIME_INTERVAL. Values: Free defined text value. Derivation: Optional, default empty. From the BRM object **/event/PIN_FLD_BAL_IMPACTS.PIN_FLD_RATE_TAG**. The post-mapping processor might decide what mapping-rule applies to this attribute.
PIN_LINEAGE	X(255)	Lineages of event fields if zone map is used in rate plan selection. Can be used to more precisely describe the balance impact detail; for example, the following concatenated, comma-separated rating-related Charge-Packet values used: ZONEMODEL, SERVICE_CODE, SERVICE_CLASS, IMPACT_CATEGORY,.RESOURCE, RUMGROUP, PRICEMODEL. Values: Free defined text value. Derivation: Optional, default empty. From the BRM object **/event/PIN_FLD_BAL_IMPACTS.PIN_FLD_LINEAGE**. The post-mapping processor might decide what mapping-rule applies to this attribute.

Table 34–23 (Cont.) Supplementary Balance Impact Packet Record Fields

Name	Format	Description
PIN_NODE_LOCATION	X(255)	Lineage information for the product. See description in products array of **/account**. Can be used to more precisely describe the balance impact detail; for example, the following concatenated, comma-separated rating-related Charge-Packet values used: REVENUEGROUP, DISCOUNTMODEL. Values: Free defined text value. Derivation: Optional, default empty. From the BRM object **/event/PIN_FLD_BAL_IMPACTS.PIN_FLD_NODE_LOCATION**. The post-mapping processor might decide what mapping-rule applies to this attribute.
PIN_QUANTITY	9(15)	Charged quantity value (beats, duration, volume), as calculated via the related RATEPLAN. Contains the rounded quantity value as it has been calculated during rating. Values: Maximum: 999999999999999 **Note:** In case of Multiple-RUM rating, this value might not be totalizable because different UoMs can logically not be aggregated. In this case, the value is set to 0.Derivation: Optional, default 0. From the BRM object **/event/PIN_FLD_BAL_IMPACTS.PIN_FLD_QUANTITY**. The post-mapping processor might decide what mapping-rule applies to this attribute; for example, multiple charge packet values might be totalized.
PIN_AMOUNT	9(11)	Amount of impact for one resource to the account balance. The value might be either positive or negative. The value is added to the PIN_FLD_CURRENT_BAL field of the PIN_FLD_BALANCES array in the account object specified by PIN_FLD_ACCOUNT_OBJ. **Note:** In case of Multiple-RUM rating, this value might be a totalized value. Values: Space: No price given, like NULL in a database Variable floating point format: Given value, might be 0.000. The floating decimal point must be set. Minimum: -9999999999 Maximum: 99999999999 Examples: '00000000125' for 125,00 '00000012.50' for 12,50 '-0012.56780' for -12,5678 Derivation: Mandatory. From the BRM object **/event/PIN_FLD_BAL_IMPACTS.PIN_FLD_AMOUNT**. The post-mapping processor might decide what mapping-rule applies to this attribute; for example, multiple charge packet values might be totalized. **Note:** This value does not include any granted discounts.

Table 34–23 (Cont.) Supplementary Balance Impact Packet Record Fields

Name	Format	Description
PIN_AMOUNT_ORIG	Decimal	Optional.
PIN_PERCENT	Decimal	Optional
PIN_AMOUNT_DEFERRED	Decimal	Optional. Calculated.
PIN_DISCOUNT	9(11)	The discount applied to this balance impact. Can be used to determine the total charge amount value. Note: The AMOUNT value never contains this DISCOUNT value. Values: Space: No price given, like NULL in a database Variable floating point format: Given value, might be 0.000. The floating decimal point must be set. Minimum: -9999999999 Maximum: 99999999999 Examples: '00000000125' for 125,00 '00000012.50' for 12,50 '-0012.56780' for -12,5678 Derivation: Mandatory, default 0.
PIN_INFO_STRING	X(2000)	Stores the price model type.

Supplementary Sub-Balance Impact Packet Record (RECType 605)

Stores balance impacts for sub-balances.

Table 34–24 describes the fields in the Supplementary Sub-Balance Impact Packet Record.

Table 34–24 Supplementary Sub-Balance Impact Packet Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long. Value: 605 Mandatory.
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
BAL_GRP_POID	String	Balance group that the balance impact applies to. Derivation: Mandatory. Calculated.

Table 34–24 (Cont.) Supplementary Sub-Balance Impact Packet Record Fields

Name	Format	Description
PIN_RESOURCE_ID	9(9)	Numeric value of the resource that is impacted; for example, 840 for US dollars. Values: Any configured BRM resource ID. Derivation: Mandatory.
NEXT_BAL	Decimal	None.
DELAYED_BAL	Decimal	None.
GRANTOR	String	The product or discount that granted this resource.
VALID_FROM_DETAILS	Integer	Sub-balance start time mode (such as first-usage or relative) and relative offset and unit.
VALID_TO_DETAILS	Integer	Sub-balance end time mode (such as relative) and relative offset and unit.

Supplementary Sub-Balance Info Packet Record (RECType 607)

Stores validity dates for sub-balances.

Table 34–25 lists the fields in the Supplementary Sub-Balance Info Packet Record.

Table 34–25 Supplementary Sub-Balance Info Packet Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long. Value: 607 Mandatory.
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
PIN_AMOUNT	Decimal	Sub-balance amount.
VALID_FROM	Date	Valid from date for this sub-balance.
VALID_TO	Date	Valid to date for this sub-balance.

Tax Jurisdiction Packet

Table 34–26 lists the fields in the Tax Jurisdiction Packet.

Table 34–26 Tax Jurisdiction Packet Fields

Name	Format
RECORD_TYPE	String
RECORD_NUMBER	Integer

Table 34–26 (Cont.) Tax Jurisdiction Packet Fields

Name	Format
PIN_TAX_TYPE	String
PIN_TAX_VALUE	Decimal
PIN_AMOUNT	Decimal
PIN_TAX_RATE	String
PIN_AMOUNT_GROSS	Decimal

EDR Container Fields for Balance Monitoring

The following fields are used for handling balance monitor information.

MONITOR_LIST (DETAIL.CUST_A.ML)

The MONITOR_LIST packet contains information about the balance monitor.

Table 34–27 lists the fields in the MONITOR_LIST packet.

Table 34–27 MONITOR_LIST Packet Fields

Name	Format	Description
BALANCE_GROUP_ID	String	Balance monitor group ID. Mandatory.
MONITOR_OWNER_ACCT_ID	String	Monitor owner's account ID. Mandatory.
MONITOR_OWNER_ID	String	Monitor owner ID. Mandatory.
MONITOR_OWNER_TYPE	String	Monitor owner type. Mandatory.

MONITOR_PACKET (DETAIL.ASS_PIN.MP)

The MONITOR_PACKET packet stores information about the balance monitor impacts. This information is added to the Associated Billing Record to be loaded into the database.

Table 34–28 lists the fields in the MONITOR_PACKET packet.

Table 34–28 MONITOR_PACKET (DETAIL.ASS_PIN.MP) Fields

Name	Format	Description
RECORD_TYPE	String	Type of record. Extended to be 3 bytes long. Possible value: 800
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. Important: This record number can change if the sequence of records changes; for example, if new record types are inserted.
ACCOUNT_POID	String	POID of the account that the monitor balance impact applies to. Derivation: Mandatory. Calculated.
BAL_GRP_POID	String	Balance monitor group that the monitor balance impact applies to. Derivation: Mandatory. Calculated.
PIN_RESOURCE_ID	9(9)	Numeric value of the resource that is impacted; for example, 840 for US dollars. Possible value: Any configured resource ID. Derivation: Mandatory.
PIN_AMOUNT	9(11)	Amount of impact for one resource to the monitor balance. The value might be either positive or negative. The value is added to the PIN_FLD_CURRENT_BAL field of the PIN_FLD_BALANCES array in the account's monitor object specified by PIN_FLD_ACCOUNT_OBJ field. **Note:** In case of Multiple-RUM rating, this value might be a total value. Possible values: Price (see below for maximum and minimum). If no price given, space; for example, NULL in a database. The format is variable floating point. The floating decimal point must be set if the given value is not in the required format. Example: '00000000125' for 125,00 '00000012.50' for 12,50 '-0012.56780' for -12,5678 Derivation: Mandatory. Derived from the object **/event/PIN_FLD_BAL_IMPACTS.PIN_FLD_AMOUNT**. The post-mapping processor decides what mapping rule applies to this attribute; for example, add multiple charge packet values. **Note:** This value does not include any granted discounts.

MONITOR_SUB_BAL_IMPACT (DETAIL.ASS_PIN.MSBI)

Table 34–29 lists the fields in the MONITOR_SUB_BAL_IMPACT packet.

Table 34–29 MONITOR_SUB_BAL_IMPACT Fields

Name	Format	Description
RECORD_TYPE	String	Type of record. Extended to be 3 bytes long. Possible value: 805
RECORD_NUMBER	9(9)	Sequence number of the record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** This record number can change if the sequence of records changes; for example, if new record types are inserted.
BAL_GRP_POID	String	Balance monitor group that the monitor balance impact applies to. Derivation: Mandatory. Calculated.
PIN_RESOURCE_ID	9(9)	Numeric value of the resource that is impacted; for example, 840 for US dollars. Possible values: Any configured resource ID. Derivation: Mandatory.
MONITOR_SUB_BAL	SB	Sub-balance monitor.

MONITOR_SUB_BAL (DETAIL.ASS_PIN.MSB)

Table 34–30 lists the MONITOR_SUB_BAL Packet fields.

Table 34–30 MONITOR_SUB_BAL Fields

Name	Format	Description
RECORD_TYPE	String	Type of record. Extended to be 3 bytes long. Possible value: 807 Derivation: Mandatory.
RECORD_NUMBER	Integer	Contains the UTC time offset that normalizes the VALID_FROM timestamp to the UTC time zone.
PIN_AMOUNT	Decimal	Sub-balance amount.
VALID_FROM	Date	Contains a timestamp of the event end time, rounded to midnight.
VALID_TO	Date	Contains a timestamp of the VALID_FROM time plus 1 day.
CONTRIBUTOR	String	None
NEXT_BAL	Decimal	None
DELAYED_BAL	Decimal	None
ACCOUNT_POID_STR	String	None
SERVICE_POID_STR	String	None

Table 34–30 (Cont.) MONITOR_SUB_BAL Fields

Name	Format	Description
OFFERING_POID_STR	String	None
START_T	Date	None
UTC_TIME_OFFSET	String	None
DESCRIPTION	String	Optional.
FLAGS	Integer	Optional.

Associated Invoice Data Record (RECType @INTEGRATE)

The Associated Invoice Data Record stores data for displaying on invoices.

Table 34–31 lists the fields in the Associated Invoice Data Record.

Table 34–31 Associated Invoice Data Record Fields

Name	Format	Description
RECORD_TYPE	String	The name of the invoice data template, preceded by the @ symbol. See "Specifying invoice data from Pipeline Manager and custom applications" in *BRM Designing and Generating Invoices*. Values: @INTEGRATE
A_NUMBER	String	See A_NUMBER.
B_NUMBER	String	See B_NUMBER.
BASIC_SERVICE	String	See BASIC_SERVICE.
NUMBER_OF_UNITS	Decimal	See NUMBER_OF_UNITS.
USAGE_CLASS	String	See USAGE_CLASS.
TERMINATING_SWITCH_IDENTIFICATION	String	See TERMINATING_SWITCH_IDENTIFICATION.
BALANCE_IMPACT	N/A	Balance impact data.
INVOICE_DATA_TERMINATOR	String	N/A

Associated Zone Breakdown Record (RECType 960-969)

Stores zoning information. For each evaluated zone type, a single Zone Breakdown Record is generated, following the Basic Detail Record (020, 021, 030, 031, etc.). A new Basic Record or the Trailer Record end the sequence of Zone Breakdown Records. Also for zone values already contained within the Basic Detail Record, these sub-details could be generated.

Table 34–32 lists the fields in the Associated Zone Breakdown Record.

Table 34–32 Associated Zone Breakdown Record Fields

Name	Format	Description
RECORD_LENGTH	Integer	Optional for backward compatibility.
RECORD_TYPE	String	Extended to be 3 bytes long. Values: 960: Standard Zoning (multiple global Zoning per logical EDR Format) 961: Segmentation Zoning (multiple Zoning per customer segment)
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
CONTRACT_CODE	X(20)	External unique contract code as defined within the associated billing system. Uniquely identifies a product-related contract. Could by used by any post-processors to look up and reference contract, subscriber, and customer data (if needed later on within this post processor). Derivation: Optional. As assigned to an ACCOUNT Object and referenced by a primary CLI. Alternatively the A Number could be used as reference.
SEGMENT_CODE	X(5)	External Segmentation ID as defined within the associated billing system or as defined within the rating process. Segments could vertically group multiple subscriber (for example, for quality reasons) or network operator. Could be used by any post-processor to identify the related customer/network segment that was used during the rating processor for this A Number. Derivation: Only mandatory for RECORD_TYPE 981, 984. As assigned to a SUBSCRIBER Object related to the A Number or as assigned to a network operator related to the file stream.
CUSTOMER_CODE	X(20)	External Customer Code as defined within the associated billing system. Could group multiple subscribers. Could be used by any post-processors as an alternative identifier to look up and reference subscriber or customer data (if needed later on within this post-processor). Derivation: Optional. As assigned to a CUSTOMER Object and referenced by a primary CLI.

Table 34–32 (Cont.) Associated Zone Breakdown Record Fields

Name	Format	Description
ACCOUNT_CODE	X(20)	External Customer-Account Code as defined within the associated billing system. Could group multiple products assigned to a customer. A customer might have multiple accounts. Could be used by any post-processors as an alternative identifier to look up and reference subscriber or customer data (if needed later on within this post-processor). Derivation: Mandatory. As assigned to a CUSTOMER Object and referenced by a primary CLI.
SYSTEM_BRAND_CODE	X(5)	External system or brand, specialist system Code as defined within the associated rating or billing. Could be used for vendor-specific reasons (for example, reseller code or target system identification for post processing, NOSP identification, etc.). Derivation: Mandatory, default 0. As defined within the SYSTEM_ BRAND Object and assigned to a PRODUCT Object referenced by a primary CLI.
SERVICE_CODE	X(5)	Internal (mapped, normalized) Service Code used for the zone determination within the associated rating or billing processor, out of the object IFW_SERVICE (.CODE). Could be used by any post-processor to evaluate the service that was really used during the related rating process. Derivation: Mandatory. The external service code is mapped to a unique representation, either: ■ Out of the service code included in the origin record (might be mapped). ■ Out of the service code associated to the SUBSCRIBER's A Number.
CUSTOMER_RATEPLAN_CODE	X(10)	The Original Product related and Customer/Subscriber specific rate plan as defined within the associated billing system. If no customer data is present, the actual, internally used rate plan could be used instead. Could be used by any post-processor to evaluate the rate plan that was really used during the related rating process. Derivation: Only mandatory for RECORD_TYPE 981, 984. As assigned to an ACCOUNT Object and referenced by a primary CLI or as assigned to the associated rating plan.

Table 34–32 (Cont.) Associated Zone Breakdown Record Fields

Name	Format	Description
SLA_CODE	X(5)	The Original product-related and customer-specific Service Level Agreement as defined within the associated billing system. If no customer data is present, the actual, internally default value could be used instead. Could be used by any post-processor to evaluate the rate plan that was really used during the related rating process. Derivation: Only mandatory for RECORD_TYPE 981, 984. As assigned to an ACCOUNT Object and referenced by a primary CLI or as assigned to the associated rating plan.
CUSTOMER_BILLCYCLE	X(2)	The Customers associated Billcycle Code as defined within the associated billing system. Could be used by any post-processor to evaluate the billcycle period that applies to this call. Derivation: Only mandatory for RECORD_TYPE 981, 984. As assigned to a CUSTOMER Object and referenced by a primary CLI.
CUSTOMER_CURRENCY	X(3)	The Customers associated Currency as defined within the associated billing system. Could be used by any post-processor to evaluate the currency and to apply exchange rates that apply to this call. Derivation: Only mandatory for RECORD_TYPE 981, 984. As assigned to a CUSTOMER Object and referenced by a primary CLI.
CUSTOMER_TAX_GROUP	X(5)	The Customers associated Tax Group Code as defined within the associated billing system. Could be used by any post-processor to evaluate the tax rate (together with the rate plan configuration related G/L account's tax code) that applies to this call. Derivation: Only mandatory for RECORD_TYPE 981, 984. As assigned to a CUSTOMER Object and referenced by a primary CLI.
NUMBER_OF_ZONE_PACKET	9(2)	Defines the number of supplementary zone records following these base fields (dynamic structure); for example, a number of '05' means that 5 records are following. Must be used to evaluate how the record structure continues. Values: 01 - 99: A minimum of at least 1 record is required Derivation: Mandatory.

Supplementary Zone Packet Record (RECType 660)

For each zone model (evaluated by any rating processor), one packet is added to this structure.

This applies only to standard and segmentation zoning (where multiple zoning is possible). All other zone values are listed in the charge breakdown records.

Table 34–33 lists the fields in the Supplementary Zone Packet Record.

Table 34–33 Supplementary Zone Packet Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long. Value: 660: Zone Packet Derivation: Mandatory. Set by the first processor and left unchanged.
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
ZONEMODEL_CODE	X(10)	External Zone Model Code as defined in the related ZONEMODEL Object (.CODE) of the rating processor. Could be used by any post-processor to evaluate the zone model that was used during the related rating process. Derivation: Mandatory.
ZONE_RESULT_VALUE_WS	X(5)	Wholesale zone result value as defined for the zone model references by the field ZONEMODEL_CODE. Could be used by any post-processor to evaluate the wholesale zone value that was estimated during the related rating process. Derivation: Optional.
ZONE_RESULT_VALUE_RT	X(5)	Retail zone result value as defined for the zone model references by the field ZONEMODEL_CODE. Could be used by any post-processor to evaluate the retail zone value that was estimated during the related rating process. Derivation: Mandatory.

Table 34–33 (Cont.) Supplementary Zone Packet Record Fields

Name	Format	Description
ZONE_ENTRY_NAME	String	Calculated, will be used by zoning and rating modules.
ZONE_DESCRIPTION	String	Calculated, will be used by zoning and rating modules.
DISTANCE	9(5)	Distance value as calculated by any geographical zone model. Could be used by any post-processor to evaluate the distance value that was estimated during the related rating process. Condition: This applies only if the associated zone model references to a geographical one. Values: The value is given in full and rounded kilometers; for example, 00150 for 150 km. Derivation: Optional. Dependent on the setup of the zone models within the related rating processor. This value represents the internal calculated distance.

Associated Charge Breakdown Record (RECType 970-998)

Stores charge data. For each evaluated charge or partial charge, a single Charge Breakdown Record might be generated, following the Basic Detail Record. A new Detail Record or the Trailer Record end the sequence of Charge Breakdown Records. For charge values already contained within the Basic Detail Record, these sub-details could be generated.

Table 34–34 lists the fields in the Associated Charge Breakdown Record.

Table 34–34 Associated Charge Breakdown Record Fields

Name	Format	Description
RECORD_LENGTH	Integer	Optional for backward compatibility.
RECORD_TYPE	String	Extended to be 3 bytes long. Values: 980: Global Charge (multiple EDR-format-related rate plan) 981: Customer Charge (subscriber-related rate plan) 982: Reseller/SP Charge (specialist-system-related rate plan) 983: Content Provider Charge (content-related rate plan) 984: Multi-Segment Charge (multiple-segment-related rate plan) 990: Carrier Interconnection Charge (trunk-related rate plan) 991: Reseller Interconnection Charge (EDR-format-related rate plan) Derivation: Mandatory. Set by the first processor and left unchanged.
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
CONTRACT_CODE	X(20)	External unique contract code as defined within the associated billing system. Uniquely identifies a product-related contract. Could by used by any post-processors to look up and reference contract, subscriber, and customer data (if needed later on within this post-processor). Derivation: Optional. As assigned to an ACCOUNT Object and referenced by a primary CLI. Alternatively the A Number could be used as reference.

Table 34–34 (Cont.) Associated Charge Breakdown Record Fields

Name	Format	Description
SEGMENT_CODE	X(5)	External Segmentation ID as defined within the associated billing system or as defined within the rating process. Segments could vertically group multiple subscriber (for example, for quality reasons) or network operator. Could be used by any post-processor to identify the related customer/network segment that was used during the rating processor for this A Number. Derivation: Only mandatory for RECORD_TYPE 981, 984. As assigned to a SUBSCRIBER Object related to the A Number or as assigned to a network operator related to the file stream.
CUSTOMER_CODE	X(20)	External Customer Code as defined within the associated billing system. Could group multiple subscribers. Could be used by any post-processors as an alternative identifier to look up and reference subscriber or customer data (if needed later on within this post-processor). Derivation: Optional. As assigned to a CUSTOMER Object and referenced by a primary CLI.
ACCOUNT_CODE	X(20)	External Customer-Account Code as defined within the associated billing system. Could group multiple products assigned to a customer. A customer might have multiple accounts. Could be used by any post-processors as an alternative identifier to look up and reference subscriber or customer data (if needed later on within this post-processor). Derivation: Mandatory. As assigned to a CUSTOMER Object and referenced by a primary CLI.
SYSTEM_BRAND_CODE	X(5)	External system or brand, specialist system Code as defined within the associated rating or billing. Could be used for vendor-specific reasons (for example, reseller code or target system identification for post-processing, NOSP identification, etc.) Derivation: Mandatory, default 0. As defined within the SYSTEM_BRAND Object and assigned to a PRODUCT Object referenced by a primary CLI.

Table 34–34 (Cont.) Associated Charge Breakdown Record Fields

Name	Format	Description
SERVICE_CODE	X(5)	Internal (mapped, normalized) Service Code used for the zone determination within the associated rating or billing processor, out of the object IFW_SERVICE (.CODE). Could be used by any post-processor to evaluate the service that was really used during the related rating process. Derivation: Mandatory. The external service code is mapped to a unique representation, either: ■ Out of the service code included in the origin record (might be mapped). ■ Out of the service code associated to the SUBSCRIBERs A Number.
CUSTOMER_RATEPLAN_CODE	X(10)	The Original Product related and Customer/Subscriber specific rate plan as defined within the associated billing system. If no customer data is present, the actual, internally used rate plan could be used instead. Could be used by any post-processor to evaluate the rate plan that was really used during the related rating process. Derivation: Only mandatory for RECORD_TYPE 981, 984. As assigned to an ACCOUNT Object and referenced by a primary CLI or as assigned to the associated rating plan.
SLA_CODE	X(5)	The Original product-related and customer-specific Service Level Agreement as defined within the associated billing system. If no customer data is present, the actual, internally default value could be used instead. Could be used by any post-processor to evaluate the rate plan that was really used during the related rating process. Derivation: Only mandatory for RECORD_TYPE 981, 984. As assigned to an ACCOUNT Object and referenced by a primary CLI or as assigned to the associated rating plan.
CUSTOMER_BILLCYCLE	X(2)	The Customer's associated Billcycle Code as defined within the associated billing system. Could be used by any post-processor to evaluate the billcycle period that applies to this call. Derivation: Only mandatory for RECORD_TYPE 981, 984. As assigned to a CUSTOMER Object and referenced by a primary CLI

Table 34–34 (Cont.) Associated Charge Breakdown Record Fields

Name	Format	Description
CUSTOMER_CURRENCY	X(3)	The Customer's associated Currency as defined within the associated billing system. Could be used by any post-processor to evaluate the currency and to apply exchange rates that apply to this call. Derivation: Only mandatory for RECORD_TYPE 981, 984. As assigned to a CUSTOMER Object and referenced by a primary CLI.
CUSTOMER_TAXGROUP	X(5)	The Customers associated Tax Group Code as defined within the associated billing system. Could be used by any post-processor to evaluate the tax rate (together with the rate plan configuration-related G/L account's tax code) that applies to this call. Derivation: Only mandatory for RECORD_TYPE 981, 984. As assigned to a CUSTOMER Object and referenced by a primary CLI.
NUMBER_OF_CHARGE_PACKETS	9(2)	Defines the number of CBRs (charge breakdown records); does not reflect the actual number of charge packets per CBR.
NUMBER_OF_TAX_PACKETS	Integer	Mandatory. Calculated. Default = 1.
CUSTOMER_OPENING_BALANCE	Decimal	If prepaid rated call, the opening balance for the subscriber. Optional.
CUSTOMER_CLOSING_BALANCE	Decimal	If prepaid rated call, the closing balance for the subscriber. Optional.
RUM_NAME	String	Optional. Calculated.
FU_DISCOUNT_OBJECTS	String	The account's discounts that have first-usage start times which were used to discount the event. Mandatory. Calculated.

Update Balance Packet

This block contains initialized sub-balances of related resources based on a deal.

Table 34–35 lists the fields in the Update Balance Packet.

Table 34–35 Update Balance Packet Fields

Name	Format	Description
RECORD_TYPE	String	The type of call record. Mandatory.
BALANCE_GROUP_ID	Integer	POID of the account's balance group for which a resource balance starts on first usage. Mandatory.
RESOURCE_ID	Integer	ID of the associated resource. Mandatory.
RECORD_NUMBER	Integer	Sequence number of the record in the file. Mandatory.
VALID_FROM	Date	The resource balance start time. Mandatory.
VALID_TO	Date	The resource balance end time. Mandatory.
VALID_FROM_DETAIL	Integer	The start time mode (such as first-usage or relative), relative offset unit (such as minutes, months, or cycles), and number of offset units. Mandatory.
VALID_TO_DETAIL	Integer	The end time mode (such as relative), relative offset unit (such as minutes, months, or cycles), and number of offset units. Mandatory.
CONTRIBUTOR	String	Balance group contributor.
GRANTOR	String	Balance group grantor.
GRANT_VALID_FROM	Date	Grant validity start time.
GRANT_VALID_TO	Date	Grant validity end time.

RUM Map Block

RUM_MAP block contains all the RUMs that are used in the ACB block.

Table 34–36 lists the fields in the RUM Map Block.

Table 34–36 RUM Map Block Fields

Name	Format	Description
RECORD_TYPE	String	None
RECORD_NUMBER	Integer	None
RUM_NAME	String	None
NET_QUANTITY	Decimal	Contains the summation of BALANCE_ PACKET.PIN_QUANTITY for RUM_NAME.
UNRATED_QUANTITY	Decimal	None

Supplementary Minimum Charge Information

Minimum charge information (the MINIMUM_CHARGE block) prevents charging a customer less than the minimum charge for a call. The values are taken from the price model configuration.

Table 34–37 lists the Supplementary Minimum Charge Information fields.

Table 34–37 Supplementary Minimum Charge Information Fields

Name	Format	Description
RESOURCE	String	Resource used when rating the call.
TOTAL_CHARGE_VALUE	Decimal	Total charge of the call.
MINIMUM_CHARGE_VALUE	Decimal	Minimum charge of the call.

Supplementary Charge Packet Record (RECType 660)

For each rate plan (evaluated by any rating processor), at least one packet is added to this structure. This applies only to rating models as defined in the record type.

Important: If a call had to be split over several time zones, there is a separate packet for each part of the call. The charge sum of all parts (where the related rate plan is equal) represents the total charge of the related basic detail EDR.

Table 34–38 lists the fields in the Supplementary Charge Packet Record.

Table 34–38 Supplementary Charge Packet Record Fields

Name	Format	Description
RECORD_TYPE	String	Value: 660: Charge Packet Record Derivation: Mandatory. Set by the first processor and left unchanged.
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
RATEPLAN_CODE	X(10)	External Rate plan Code as defined in the related RATEPLAN Object (.CODE) used by the rating process. Could be used by any post-processor to evaluate the RATEPLAN that was used during the related rating process. Derivation: Mandatory.

Table 34–38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
RATEPLAN_TYPE	X(1)	A rate plan could be either wholesale or retail. Could be used by any post-processor to determine if the charge calculated throughout the RATEPLAN used is a wholesale or retail one. Values: W: Wholesale R: Retail Derivation: Mandatory. Taken from the setup related to the RATEPLAN_CODE Field
RATETAG_CODE	String	Derivation: Mandatory. Calculated.
ZONEMODEL_CODE	X(10)	External Zone Model Code as defined in the related ZONEMODEL Object (.CODE) defined within the related rate plan used by the rating process. Could be used by any post-processor to evaluate the zone model that was really used during the related rating process. Derivation: Mandatory.
SERVICE_CODE_USED	X(5)	Internal (RATEPLAN related mapped) Service Code used for the zone determination within the associated rating or billing processor, out of the object IFW_ RATESERVICE_MAP (.NEW_SERVICECODE). Could be used by any post-processor to evaluate the service that was really used during the related rating process. Derivation: Mandatory.
SERVICE_CLASS_USED	X(5)	External Service Class Used as defined within the RATEPLAN (for example, for specific QoS) and used by the rating process. Could be used by any post-processor to evaluate the level of service quality that was really used during the related rating process. Derivation: Mandatory. Default = '*'.
IMPACT_CATEGORY	X(10)	Impact Category result value as defined; for example, a zone value references or a usage scenario map result. Could be used by any post-processor to evaluate the zone value that was estimated during the related rating process. Derivation: Mandatory.
ZONE_DESCRIPTION	String	Calculated. Used by zoning and rating modules.
IC_DESCRIPTION	String	Calculated. Used by zoning and rating modules.

Table 34–38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
ZONE_ENTRY_NAME	String	Calculated. Used by zoning and rating modules.
DISTANCE	9(5)	Distance value as calculated by any geographical zone model. Could be used by any post-processor to evaluate the distance value that was estimated during the related rating process. Condition: This applies only if the associated zone model references to a geographical one. Values: The value is given in full and rounded kilometers; for example, 00150 for 150 km. Derivation: Optional. Dependent on the setup of the zone models within the related rating processor. This value represents the internal calculated distance.
TIMEMODEL_CODE	X(10)	External Time Model Code as estimated and used by the rating process. Time model and Time zone define a unique relationship between a day code (special day, weekday, weekend, etc.) and a time interval (time band within a day). Could be used by any post-processor to evaluate the time model that was really used during the related rating process. Derivation: Mandatory. The time model is given by evaluating the RATEPLAN configuration and the starting timestamp of the record.
TIMEZONE_CODE	X(10)	External Time Zone Code as estimated and used by the rating process. Time model and Time zone define a unique relationship between a day code (special day, weekday, weekend, etc.) and a time interval (time band within a day). Could be used by any post-processor to evaluate the time zone that was really used during the related rating process. Derivation: Mandatory. The time zone is given by evaluating the RATEPLAN configuration and the starting timestamp of the record within the related TIMEMODEL.

Table 34–38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
DAY_CODE	X(10)	External Day Code as estimated and used by the rating process. Time model and Time zone define a unique relationship between a day code (special day, weekday, weekend, etc.) and a time interval (time band within a day). This attribute describes the evaluated day code within the above relationship. Could be used by any post-processor to evaluate the day code that was really used during the related rating process, even if single charge packets are being generated in case of time zone splitting. Derivation: Mandatory. DAY_CODE as defined in the related DAYCODE object (.CODE). The day is given by evaluating the RATEPLAN configuration and the starting timestamp of the record within the related TIMEMODEL and TIMEZONE. If Splitting of single charge packets is not performed, the day code of the start of the call is being used.
TIME_INTERVAL_CODE	X(10)	External Time Interval Code as estimated and used by the rating process. Time model and Time zone define a unique relationship between a day code (special day, weekday, weekend, etc.) and a time interval (time band within a day). This attribute describes the evaluated time interval within the above relationship. Could be used by any post-processor to evaluate the time interval that was really used during the related rating process, even if single charge packets are being generated in case of time zone splitting. Derivation: Mandatory. TIME_INTERVAL Code as defined in the related TIME_INTERVAL object (.CODE). The time zone is given by evaluating the RATEPLAN configuration and the starting timestamp of the record within the related TIMEMODEL and TIMEZONE. If Splitting of single charge packets is not performed, the time interval of the start of the call is being used.
PRICEMODEL_CODE	X(10)	External Price model Code as defined in the related PRICEMODEL Object (.CODE) used by the rating process. Could be used by any post-processor to evaluate the price model that was really used during the related rating process. Derivation: Mandatory. Dependent on the setup of the RATEPLAN within the related rating processor. This value represents the external price model code as it had been set up and used.

Table 34–38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
PRICEMODEL_TYPE	X(1)	Defines which type of the price model was used for this charge packet. Could be used by any post-processor to evaluate which price model was really used during the related rating process. Values: S: Standard price model was used A: Alternative price model was used Derivation: Mandatory, default 'S'. Dependent on the setup of the RATEPLAN within the related rating processor. This value represents the external type of price model as it had been set up and used. A packet with a price model type 'S' does always exist. If an alternative price model is configured, a second packet of type 'A' is generated.
RESOURCE	X(10)	Resource, which has been used for rating or discounting purposes. A resource might be a currency or any other type (for example, loyalty points) that should be used to calculate parallel charges. Could be used by any post-processor to classify the different charge items. Values: Any configured values of the IFW_RESOURCE object. Derivation: Mandatory, directly taken out of the PRICEMODEL configuration appropriate to the charge packet.
RESOURCE_ID	Integer	Derivation: Optional. Calculated. If 0, the RESOURCE field is ignored.
RESOURCE_ID_ORIG	Integer	Optional. Used if the exchange rate module is configured.
RUMGROUP	X(10)	Classifies the charging item which was derived from the service. A RUM group defines a list of RUMs that should be used together to define a total charge. For example, 'total' could consist of 'DUR'+'VOL_S'+'VOL_R'. Could be used by any post-processor to classify the different charge items. Values: Any configured values of the IFW_RUMGROUP object. Derivation: Mandatory, directly taken out of the Service object's RUM group definition.

Table 34–38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
RUM	X(10)	Classifies the charging part of a call in a intercarrier relationship, for interconnection or roaming. Values: Dependent on the setup of the IFW_RUM object. Filled by default with '*' if multiple RUMs are used within one Charge Packet.
NETWORK_OPERATOR_CODE	X(10)	Network Operator Code (or Reseller / Content Provider Code) as defined in the NO Objects (.CODE) of the related rating process. Could be used by any post-processor (especially by interconnection billing) to evaluate the network operator to which the calculated charge belongs. Condition: Network operators can be assigned as follows: ■ Directly to an EDR-format ■ Via a trunk identification (carrier/reseller interconnection) ■ A content provider code via a special number, b# Derivation: Only mandatory for RECORD_TYPE 983, 990, 991. The network operator is given by evaluating the relationship during the estimation process which RATEPLAN should be used.
NETWORK_OPERATOR_BILLINGTYPE	X(1)	Classifies the Type of the associated network operator involved within this charge. Could be used by any interconnection processor to distinguish between incoming and outgoing charges. Condition: Only applies to interconnection rating. Values: O: Outgoing NO, charges have to be paid to the related NO I: Incoming NO, charges are received by the related NO Derivation: Only mandatory for RECORD_TYPE 983, 990, 991; else default I. The NO billing type is directly related to the network operator setup.

Table 34–38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
CHARGE_TYPE	X(1)	Classifies the charging part of a call in an intercarrier relationship, for interconnection or roaming. Could be used by any interconnection processor to classify the different charge types. Condition: Only applies to interconnection rating. Values: I: Inroute Charge (only applies for carrier interconnection) O: Outroute Charge (only applies for carrier interconnection) T: Transit Charge (only applies for carrier interconnection) N: Normal Charge (applies for all other charges, default) Derivation: Only mandatory for RECORD_TYPE 983, 988, 990, 991, 995, 996; else default N. The switch/trunk is directly related to the type setup.
TRUNK_USED	X(15)	Trunk ID or MSC which was used to calculate the interconnection charges within this packet. This field contains the internal, virtual or mapped trunk address and not the external one and is related either to the inroute or outroute trunk (see field CHARGE_TYPE as a reference). Could be used by any interconnection processor to classify the different service/charge types. Condition: Only applies to interconnection rating. Derivation: Only mandatory for RECORD_TYPE 990, 991. The trunk ID is directly related to the network model and mapping rules used.
POI_USED	X(10)	POI In which was used to calculate the interconnection charges within this packet. Could be used by any interconnection processor to classify the different service/charge types. Condition: Only applies to interconnection rating. Derivation: Only mandatory for RECORD_TYPE 990, 991. The POI is directly related to the network model and mapping rules used.

Table 34–38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
PRODUCTCODE_USED	X(10)	Internal product which was used to calculate the charges within this packet. This field contains the internal, virtual or mapped network service type and is related either to the inroute or outroute trunk (ICPRODUCT). Could be used by any interconnection processor to classify the different service/charge types. Condition: Only applies to interconnection rating. Derivation: Only mandatory for RECORD_TYPE 990, 991. The Product Code is directly related to the IC Product Code configuration.
PIN_LOGIN_ALIAS	String	Optional. Calculated.
CHARGING_START_TIMESTAMP	YYYYMM DDHH24 MISS	The timestamp used for charging. Format: YYYYMMDDHHMISS; for example, 19990518190357. Optional Field Usage: It is possible to read/write dates in number of seconds since 01.01.1970 00:00:00; for example, 12345. The internal representation is the format YYYYMMDDHHMISS anyway. This is just an optional input/output format conversion.
CHARGEABLE_QUANTITY_VALUE	9(15)	Original chargeable units (beats, duration), as provided by the sender (for example, network element or any other given input format). Contains the original, not-rounded quantity value. Might be useful by some kind of processors analyzing as how many units the call was originally treated by the sender and/or as many the record was treated during rating. Values: Maximum: 999999999999999 Examples: CHARGABLE_QUANTITY_VALUE = 87 sec. a) if RATEPLAN is defined with a 60sec. beat -> ROUNDED_QUANTITY_VALUE will contain 120sec. b) if RATEPLAN is defined with a 30sec. beat -> ROUNDED_QUANTITY_VALUE will contain 90sec. Derivation: Optional, defaulted by ROUNDED_QUANTITY_ VALUE if not provided or present. Set by either the input format or the rating processor generating this packet and left unchanged.

Table 34–38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
ROUNDED_QUANTITY_VALUE	9(15)	Charged units (beats, duration), as calculated via the related RATEPLAN. Contains the rounded quantity value as it has been calculated during rating. Might be useful by some kind of processors analyzing as how many units the call was originally treated by the sender and/or as many the record was treated during rating. Values: Maximum: 999999999999999 Examples: CHARGABLE_QUANTITY_VALUE = 87 sec. a) if RATEPLAN is defined with a 60sec. beat -> ROUNDED_QUANTITY_VALUE will contain 120sec. b) if RATEPLAN is defined with a 30sec. beat -> ROUNDED_QUANTITY_VALUE will contain 90sec. Derivation: Mandatory, defaulted 0. Set by the rating processor generating this packet and left unchanged.
ROUNDED_QUANTITY_FROM	Decimal	Mandatory. Calculated.
ROUNDED_QUANTITY_TO	Decimal	Mandatory. Calculated.
ROUNDED_QUANTITY_UoM	X(3)	The Unit of Measurement associated with the Rounded Chargeable Quantity Value. Can be used to interpret the quantity value, but usually not needed because the quantity itself is sufficient for all rating steps. Values: As specified in the database model, if a UoM conversion had been carried out; else as defined in the related Basic Detail Record. Derivation: Mandatory, default 'SEC'. Set by the rating processor and left unchanged.
QUANTITY_FROM	Decimal	Charge packet start quantity.
QUANTITY_TO	Decimal	Charge packet end quantity.

Table 34–38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
EXCHANGE_RATE	9(11)	Contains the exchange rate which has been used to convert the Incoming currency to the internal currency as indicated in the field CHARGED_ CURRENCY_TYPE. Can be used to convert the virtual currency SDR (which is used in conjunction of TAP) to internal currencies and convert the Charge back to SDR after Rating. This would be a typical usage for Interconnection Rating. Values: Variable floating point format: Given value, might be 0.000. The floating decimal point must be set. Minimum: -9999999999 Maximum: 99999999999 Examples: '00000000125' for 125,00 '00000012.50' for 12,50 '-0012.56780' for -12,5678 Derivation: Optional, defaulted 00000000001 (=1,00).
EXCHANGE_CURRENCY	X(3)	Currency code as defined for the exchange rate; for example, "DEM" or "EUR". Can be used to interpret the exchange rate to distinguish to which currency the exchange rate was used for. For example, for TAP: the charged amount might be given in SDR currency and the exchange rate will define the rate used to convert into local currency. Derivation: Optional. As related to the exchange rate value used for. Use the three-digit ISO currency code.
CHARGED_CURRENCY_TYPE	X(1)	Indicates which of the available currencies was used to generate the charge packet. Could be used by any post-processor to classify the different charge packets. Values: R: Rating Currency (default) B: Billing Currency H: Home Currency **Note:** In case of currency conversion, where in parallel all three currency models are supported (R, B, and H); there is one charge packet for each currency type. Derivation: Depending on the function module which generated the charge packet, the value is set to one of the values given above. The rating modules set the value to "R", while the ExchangeRate module generates two charge packets (one for the home currency and one for the billing currency). This feature is usually required for interconnection purposes.

Table 34–38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
CHARGED_AMOUNT_VALUE	9(11)	The charge for the call (could be any kind of price). A monetary amount assigned to the call by any rating processor. This includes any toll charge but does not include any CAMEL invocation fee. In case of interconnection or roaming charges, this is the advice of charge. Can be used by any post-processor to collect multiple charges related to one record. This opens the possibility to keep more then one charge. With this structure there is the possibility to keep all charges related to the record/call (for example, end-customer, wholesale, content provider, reseller, multi-segment rating, optimized data warehouse etc.). Values: Space: No price given, like NULL in a database Variable floating point format: Given value, might be 0.000. The floating decimal point must be set. Minimum: -9999999999 Maximum: 99999999999 Examples: '00000000125' for 125,00 '00000012.50' for 12,50 '-0012.56780' for -12,5678 Derivation: Mandatory.
CHARGE_REFUND_INDICATOR	Integer	Optional. Charge refund indicator item.
CHARGED_AMOUNT_VALUE_ORIG	Decimal	Optional. Used if the exchange rate module is configured.
CHARGED_AMOUNT_CURRENCY	X(3)	Currency code as defined within the associated RATEPLAN; for example, DEM or EUR. Can be used to interpret the amount value and to distinguish between different currencies (multicurrency support). Any rating or billing processor might convert the different currencies. Derivation: Mandatory. As related to the RATEPLAN used. Use the three-digit ISO currency code.
CHARGED_TAX_TREATMENT	X(1)	Charges might be inclusive or exclusive of tax. Can be used to interpret the amount value and to distinguish between net and gross charges. Values: Y: Tax included in the charge N: Tax not included in the charge (default) Derivation: Mandatory (default N).

Table 34–38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
CHARGED_TAX_RATE	9(4)	Defines the tax rate applicable to the charge. Because some national legal definitions dictate that the tax rate applicable is determined by the invoice date, there is a possibility that the rate on the invoice might differ from the rate on the transfer. However, the likelihood of this happening is extremely low. Can be used to interpret the amount value and to convert between net and gross charges or to represent customer-specific rates. Values: 0000 through 9999 (2 fixed decimals) Example: 16.00% 1600 Derivation: Optional. As related to the taxation module used (refer to IFW_TAX.TAXRATE).
CHARGED_TAX_CODE	X(5)	Defines the tax rate applicable to the charge. Because some national legal definitions dictate that the tax rate applicable is determined by the invoice date, there is a possibility that the rate on the invoice might differ from the rate on the transfer. However, the likelihood of this happening is extremely low. Can be used by any billing processor to interpret the amount value and to convert and calculate a customer-specific tax rate. Values: As defined via the reference IFW_ GLACCOUNT.TAXCODE. Example: M16 for tax code M16. Derivation: Optional. As related to the RATEPLAN configuration's general ledger account used.
USAGE_GL_ACCOUNT_CODE	X(10)	The General Ledger Code defines a reference applicable to the usage revenue account. Can be used by any billing processor to interpret the amount value and to convert and calculate a customer-specific tax rate or to trigger any account balance bookings within a financial accounting system. Values: As defined in the database object IFW_RATEPLAN_ CNF.USG_GLACCOUNT. Example: USG_AIRTEL for account usage revenue airtime. Derivation: Optional. As related to the RATEPLAN used.

Table 34–38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
REVENUE_GROUP_CODE	X(5)	The Revenue Group Code defines a reference applicable to a specific group of usage revenue. Different usage groups can be used to define split billing rules to be distributed to different customers; for example, airtime is paid by a subscriber and monthly periodic fees are paid by the customer above. Usage: Can be used by any billing processor to interpret a split billing based on different revenue groups. Values: As defined in the database object IFW_RATEPLAN_CNF.REVENUEGROUP. Example: AIR for usage revenue group airtime. Derivation: Optional. As related to the RATEPLAN used.
DISCOUNTMODEL_CODE	X(10)	External Discount model Code as defined in the related DISCOUNTMODEL Object (.CODE) used by the rating process. Could be used by any post-processor to evaluate the discount model that was really used during the related rating process. Derivation: Mandatory. Dependent on the setup of the RATEPLAN within the related rating processor. This value represents the external discount model code as it had been set up and used.

Table 34–38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
GRANTED_DISCOUNT_AMOUNT_VALUE	9(11)	The field records the total discount value, which was granted to calculate the correct CHARGED_AMOUNT_VALUE. Can be used by any post-processor analyzing the discounts that had been granted and must be used to calculate the exact charge (with discount included). This will give a good indicator of the discount structure and usage. **Note:** The CHARGED_AMOUNT_VALUE never contains this DISCOUNT_VALUE. Values: Space: No price given, like NULL in a database Variable floating point format: Given value, might be 0.000. The floating decimal point must be set. Minimum: -9999999999 Maximum: 99999999999 Examples: '00000000125' for 125,00 '00000012.50' for 12,50 '-0012.56780' for -12,5678 Derivation: Mandatory, default 0. Note: The currency is always the same currency as given in the field CHARGED_AMOUNT_CURRENCY.
GRANTED_DISCOUNT_QUANTITY_VALUE	9(15)	The total discount quantity value, which was granted to calculate the correct CHARGED_AMOUNT_VALUE; for example, Applied free minutes. Not updated by discounting. Used in interconnect mapping. Can be used by any post-processor analyzing the discounts that had been granted. Could be used to recalculate the original quantity value (without any discount). This will give a good indicator of the discount structure and usage. Condition: Only relevant for quantity (duration) based service discounts. Values: Maximum: 999999999999999 Derivation: Mandatory, default 0. Might be set by any rating or pre-billing processor.

Table 34–38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
GRANTED_DISCOUNT_QUANTITY_UoM	X(3)	The Unit of Measurement associated with the Granted Discount Quantity Value. Not updated by discounting. Used in interconnect mapping. Can be used to interpret the quantity value, but usually not needed because the quantity itself is sufficient for all rating steps. Values: As specified in the database model, if a UoM conversion had been carried out; else as defined in the related Basic Detail Record. Derivation: Mandatory, default 'SEC'. Set by the rating processor and left unchanged.
DEFERRED_AMOUNT	Decimal	Optional Calculated
PIN_PERCENT	Decimal	Optional Calculated
NUMBER_OF_DISCOUNT_PACKETS	9(2)	Defines the number of supplementary discount packet records following these base fields (dynamic structure); for example, 05 means that 5 records are following. Must be used to evaluate how the record structure continues. Values: 00 - 99: either zero or N record(s) are following Derivation: Mandatory.
VALID_FROM	Date	Optional.
VALID_TO	Date	Optional.
CYCLE_OFFSET	Integer	Optional. Identifies a grant's validity period.
CHARGE_INDEX	Integer	The array index of incoming charge packets, used by the discount pipeline to match existing charges, in case a credit limit check changed the original charges.

Split Charge Packet

FCT_Discount splits charge packets if necessary during prepaid authorization. Each split charge packet represents a segment with a single net rate, including discounts.

Table 34–39 lists the fields in the Split Charge Packet.

Table 34–39 Split Charge Packet Fields

Name	Format	Description
RESOURCE_ID	Integer	Numeric ID of the resource. Used for filtering in the discount detail. Copied from the original charge packet.
RUM	String	RUM name. Copied from the original charge packet.
QUANTITY_FROM	Decimal	Split charge packet start quantity. Calculated by the module.
QUANTITY_TO	Decimal	Split charge packet end quantity. Calculated by the module.
CHARGED_AMOUNT_VALUE	Decimal	Amount of the charge for this split charge packet. Calculated by the module.
INTERN_PACKET_INDEX	Integer	The index of the split charge packet.
INTERN_SRC_PACKET_INDEX	Integer	The packet index of the charge packet from which this split charge packet was generated.

Supplementary Last Beat Information

Information about the last beat (the LAST_BEAT_INFO block) is mainly used for abnormal call terminations.

Table 34–40 lists the fields in the Supplementary Last Beat Information block fields.

Table 34–40 Supplementary Last Beat Information Fields

Name	Format	Description
LAST_BEAT_QUANTITY	Decimal	The length of a beat. The value of a beat (for example, clicks or bytes) is defined in the price model and determined by *FCT_MainRating*.
LAST_BEAT_CHARGE	Decimal	The charge for a beat.

Charge Breakdown Record Tax Packet (RECType 660)

Add code to the OutGrammar to store tax information from the EDR container into the output TAP blocks.

Block. *n* times. Optional.

Table 34–41 lists the fields in the Charge Breakdown Record Tax Packet.

Table 34–41 Charge Breakdown Record Tax Packet Fields

Name	Format	Description
RECORD_TYPE	String	Mandatory. Must be set to 660.
RECORD_NUMBER	Integer	Mandatory. Auto-generated.
TAX_CODE	Integer	None
TAX_RATE	String	None
TAX_VALUE	Decimal	None
TAX_PERCENT	Decimal	None
TAX_VALUE_ORIG	Decimal	Optional. Used when exchange rate is configured.
TAX_TYPE	String	None

Table 34–41 (Cont.) Charge Breakdown Record Tax Packet Fields

Name	Format	Description
CHARGE_TYPE	String	None
TAXABLE_AMOUNT	Decimal	None
TAX_QUANTITY	Decimal	None
RELATED_RECORD_NUMBER	Integer	None
RELATED_CHARGE_INFO_ID	Integer	None
CHARGE_INFORMATION_COUNTER	Integer	None
CHARGE_INFORMATION_COUNTER	Integer	None

Associated Message Description Record (RECType 999)

Stores errors that occur in preprocessing. For each error a single Message Description Record is generated, following the Basic Record. A new Basic Record or the Trailer Record end the sequence of Message Description Records.

Table 34–42 lists the fields in the Associated Message Description Record.

Table 34–42 Associated Message Description Record Fields

Name	Format	Description
RECORD_TYPE	String	Value: 999: Message Description Record Derivation: Mandatory. Set by the first processor and left unchanged. Usage: Determination of the different record types.
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. Important: Following modules might change this record number; for example, if new record types are inserted.
SYSTEM	X(8)	Description or Code of the system which produced this record; for example, Host name or process name.

Table 34–42 (Cont.) Associated Message Description Record Fields

Name	Format	Description
MESSAGE_SEVERITY	X(1)	Severity code for this message. Values: N: Normal (Hint) W: Warning E: Error (could be either a minor, major, or critical error) Derivation: Mandatory. Set by the first processor and left unchanged.
MESSAGE_ID	N(7)	An error code used to cross-reference the call to the relevant description. Values: 0000000 through 9999999 Derivation: Mandatory. Set by the first processor and left unchanged.
MESSAGE_ DESCRIPTION	X(50)	Description of the error. It is mandatory but the content is entirely at the discretion of the pre-processor. Derivation: Mandatory. Set by the first processor and left unchanged.

Associated TAP Error Record

Table 34–43 lists the Associated TAP Error Record fields.

Table 34–43 Associated TAP Error Record Fields

Name	Format
RECORD_TYPE	String
RECORD_NUMBER	Integer
ERROR_NAME	String
ERROR_SEVERITY	Integer
TAP3_ERROR_CODE	String
TAP3_ERROR_APPLICATION_TAG	String
TAP3_ERROR_DEPTH	String

Associated SMS Record (RECType 580)

This optional record is used to store SMS call data.

Table 34–44 lists the fields in the Associated SMS Record.

Table 34–44 Associated SMS Record Fields

Name	Format	Description
RECORD_TYPE	String	Record type for Associated SMS Record. Value: 580 Derivation: Mandatory. Set by the first processor.
RECORD_NUMBER	9(9)	Sequence number of record in file. Used to ensure a linear sequence order for all records; for example, as a sorting criteria. Values: Minimum: 000000002 Maximum: 999999998 Derivation: Mandatory. Auto-generated. Set by the first processor. Important: Record number may change; for example, if new record types are inserted.
CONTENT_ INDICATOR	X(1)	Indicator as to the contents of the message sent or received. Derivation: Optional.
ORIGINATING SWITCH IDENTIFICATION	X(10)	SMS-C from which the SMS was issued by the A party. Derivation: Optional.
DESTINATION SWITCH IDENTIFICATION	X(10)	SMS-C from which the SMS was issued to the B party. Derivation: Optional.
PROVIDER ID	X(2)	Unique Service Provider Identifier. Derivation: Optional.
SERVICE ID	X(2)	Unique Service ID. Derivation: Optional.
DEVICE NUMBER	X(24)	Identifies the equipment used by the subscriber during the call; for example, the International Mobile Equipment Identity number (IMEI). Derivation: Optional.
PORT NUMBER	X(24)	Identifies the unique subscriber ID; for example, the IMSI number. Derivation: Optional.
DIALED DIGITS	X(40)	The number dialed by the customer when establishing a call or the number to which the call is forwarded or transferred. Derivation: Optional.

Associated MMS Record (RECType 590)

This optional record is used to store MMS call data.

Table 34–45 lists the Associated MMS Record fields.

Table 34–45 Associated MMS Record Fields

Field Name	Data Format	Description
RECORD TYPE	String	Record type for Associated MMS Record. Value: 590 Derivation: Mandatory. Set by the first processor.
RECORD NUMBER	9(9)	Sequence number of record in file. Used to ensure a linear sequence order for all records; for example, as a sorting criteria. Values: Minimum: 000000002 Maximum: 999999998 Derivation: Mandatory. Auto-generated. Set by the first processor. **Important:** Record number may change; for example, if new record types are inserted.
ACCOUNT STATUS TYPE	X(2)	Indicator of the account type from which the message was sent. Derivation: Optional.
PRIORITY	X(2)	Indicator as to the priority of the message; for example, high, medium or low. Derivation: Optional.
MESSAGE CONTENT	X(255)	Content type; for example, image or plain text. Derivation: Optional.
MESSAGE ID	X(16)	Unique message group ID. If the message was sent as part of a group, an indicator as to which group it was sent from. Derivation: Optional.
STATION IDENTIFIER	X(255)	Station from which message was sent. Value: MMS identifier. Derivation: Optional.
FC INDICATOR	X(9)	Indicator as to whether the message was forwarded or copied. Derivation: Optional.

Table 34–45 (Cont.) Associated MMS Record Fields

Field Name	Data Format	Description
CORRELATION ID	X(16)	Correlation ID for the message if part of a group. Derivation: Optional.
DEVICE NUMBER	X(24)	Identifies the equipment used by the subscriber during the call; for example, the International Mobile Equipment Identity number (IMEI). Derivation: Optional.
PORT NUMBER	X(24)	Identifies the unique subscriber ID; for example, the IMSI number. Derivation: Optional.
DIALED DIGITS	X(40)	The number dialed by the customer when establishing a call or the number to which the call is forwarded or transferred. Derivation: Optional.
CELL ID	X(10)	Cell ID of the A party, or the cell from which the call originated. Derivation: Optional.
B CELL ID	X(10)	Cell ID of the B party, or the cell receiving the call. Derivation: Optional.
A TERM CELL ID	X(10)	Cell ID of the A party when the call terminated. Derivation: Optional.

Trailer Record (RECType 090)

Table 34–46 lists the fields in the Trailer Record.

Table 34–46 Trailer Record Fields

Name	Format	Description
RECORD_LENGTH	Integer	Optional for backward compatibility.
RECORD_TYPE	String	Extended to be 3 bytes long, first byte denotes market like GSM, ISDN. Value: 090: Trailer Record Derivation: Mandatory. Set by the first processor and left unchanged. Usage: Determination of the different record types.
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor.
SENDER	X(10)	Identifies the PLMN or physical (network) operator, which is sending the file, used to determine the network, which is the sender of the data. The full list of mobile codes in use is given in MoU TADIG PRD TD. 13: PLMN Naming Conventions. Can be used to determine a unique NOSP_ID together with the RECIPIENT. Can also be used to determine the network operator responsible for the EDR. Derivation: Optional, but should be defaulted if not present on the input side, for example, by own NO-Id, for example, 'DTAG'. Set by the first processor and left unchanged.
RECIPIENT	X(10)	Identifies the PLMN or physical (network) operator to whom the file is being sent, used to determine the network, which is the recipient of the data. The full list of mobile codes in use is given in MoU TADIG PRD TD. 13: PLMN Naming Conventions. Can be used to determine a unique NOSP_ID together with the SENDER. Can also be used to determine the reseller or service provider who is responsible for billing these CDRs. Derivation: Optional, but should be defaulted if not present on the input side, for example, by own NO-Id, for example, 'DTAG'. Set by the first processor and left unchanged.

Table 34–46 (Cont.) Trailer Record Fields

Name	Format	Description
SEQUENCE_NUMBER	9(6)	Identifies each file sent by the VPLMN or logical sender to a particular HPLMN or logical recipient. It indicates the file number of the specific file type, starting at 1 and increments by one for each new file of that type sent. Separate sequence numbering must be used for Test- and Chargeable-Data. Having reached the maximum value (99999), the number must recycle to 1. **Note:** In the case of retransmission for any reason, this number does not increment. Range: 000001 - 999999 for Test Data 000001 - 999999 for Chargeable Data Derivation: Mandatory. Set by the first processor and could be changed by any following processor in case of recycling to assure a unique and linear sequence order to all following processors.
ORIGIN_SEQUENCE_NUMBER	9(6)	Original file sequence number as generated the first time. Identical content as for the SEQUENCE_ NUMBER, but will never be changed. Used as a reference to the original file/stream, if any processor has changed the actual file sequence number. Derivation: Mandatory, defaulted by SEQUENCE_NUMBER. Set by the first processor and left unchanged.
TOTAL_NUMBER_OF_RECORDS	9(9)	The total number of Basic Record in the file, excluding header and trailer. Should be used as a check value, to determine that all records have been correctly transmitted and/or used. Condition: Not Present in a Notification File or if no Detail records are present. Maximum Number: 999999999 Derivation: Mandatory. Might be recalculated by any processor.
TAP_TOTAL_NUMBER_OF_RECORDS	Integer	Mandatory. Set by TAP input grammar.

Table 34–46 (Cont.) Trailer Record Fields

Name	Format	Description
FIRST_START_TIMESTAMP	YYYYMMDD HHMISS	The earliest start of charging timestamp on any Basic Detail Record. It is not necessarily the start of charging timestamp of the first charge record on the file. Should be used as a check value, to determine that all records have been correctly transmitted and/or used. Condition: Not present in a Notification File or if no Detail records are present. Format: YYYYMMDD HHMISS (see also 'Time-stamp') local time, and not UTC time, is used to determine the earliest call. Optional Field Usage: It is possible to read/write dates in number of seconds since 01.01.1970 00:00:00; for example, 12345. The internal representation is the format YYYYMMDDHHMISS anyway. This is just an optional input/output format conversion. Derivation: Mandatory.
FIRST_CHARGING_UTC_TIME_ OFFSET	X(5)+/-HHMI	All timestamps are sender (VPLMN) local time. So that the time can be equated to time in the recipient (HPLMN) local time, the sender shall give the difference between local time and UTC time. UTC Time Offset = Local Time minus UTC Time. Can be used to translate the TRANSFER_CUTOFF_ TIMESTAMP into a unified UTC time. This might be useful if a centralized rating and billing will take place. Example: Washington DC, USA 1000hrs 10/10/97 UTC Time 1500hrs 10/10/97 UTC Time Offset = 10 - 15 = -0500 Madrid, Spain 1600hrs 10/10/97 UTC Time 1500hrs 10/10/97 UTC Time Offset = 16 - 15 = +0100 Sydney, Australia 0100hrs 11/10/97 UTC Time 1500hrs 10/10/97 UTC Time Offset = (01 + 24) - 15 = +1000 **Note:** Where dates are different, 24 is added to the time of the greater date Derivation: Mandatory. Set by the first processor and left unchanged.

Table 34–46 (Cont.) Trailer Record Fields

Name	Format	Description
LAST_START_TIMESTAMP	YYYYMMDD HHMISS	The latest start of charging timestamp on any Basic Detail Record. It is not necessarily the start of charging timestamp of the last charge record on the file. Should be used as a check value, to determine that all records have been correctly transmitted and/or used. Might also be used to validate that all records are earlier then the given transfer cutoff timestamp (see header record). Condition: Not present in a Notification File or if no Detail records are present. Format: YYYYMMDD HHMISS (see also 'Time-stamp') local time, and not UTC time, is used to determine the earliest call. Optional Field Usage: It is possible to read/write dates in number of seconds since 01.01.1970 00:00:00; for example, 12345. The internal representation is the format YYYYMMDDHHMISS anyway. This is just an optional input/output format conversion. Derivation: Mandatory.
LAST_CHARGING_UTC_TIME_ OFFSET	X(5)+/-HHMI	All timestamps are sender (VPLMN) local time. So that the time can be equated to time in the recipient (HPLMN) local time, the sender gives the difference between local time and UTC time. UTC Time Offset = Local Time minus UTC Time. Can be used to translate the TRANSFER_CUTOFF_ TIMESTAMP into a unified UTC time. This might be useful if a centralized rating and billing will take place. Example: Washington DC, USA 1000hrs 10/10/97 UTC Time 1500hrs 10/10/97 UTC Time Offset = 10 - 15 = -0500 Madrid, Spain 1600hrs 10/10/97 UTC Time 1500hrs 10/10/97 UTC Time Offset = 16 - 15 = +0100 Sydney, Australia 0100hrs 11/10/97 UTC Time 1500hrs 10/10/97 UTC Time Offset = (01 + 24) - 15 = +1000 **Note:** Where dates are different, 24 is added to the time of the greater date. Derivation: Mandatory. Set by the first processor and left unchanged.

Table 34–46 (Cont.) Trailer Record Fields

Name	Format	Description
TOTAL_RETAIL_CHARGED_VALUE	9(15)	The sum of the Retail Charged Amount Value of any Basic Detail Record. The toll element of a charge is that portion related to the carrier and Destination. There will only be one toll charge present for all chained records. Should be used as a check value, to determine that all records have been correctly transmitted and/or used. Values: Space: No price given, like NULL in a database. Variable floating point format: Given value, might be 0.000. The floating decimal point must be set. Minimum: -99999999999999 Maximum: 999999999999999 Examples: '00000000125' for 125,00 '00000012.50' for 12,50 '-0012.56780' for -12,5678 Derivation: Mandatory.
TOTAL_WHOLESALE_CHARGED_VALUE	9(15)	The sum of the Wholesale Charged Amount Value of any Basic Detail Record contained in the batch. This is present for audit purposes only. Should be used as a check value, to determine that all records have been correctly transmitted and/or used. Values: Variable floating point format: Given value, might be 0.000. The floating decimal point must be set. Minimum: -99999999999999 Maximum: 999999999999999 Examples: '00000000125' for 125,00 '00000012.50' for 12,50 '-0012.56780' for -12,5678 Derivation: Mandatory.
TAP_TOTAL_CHARGE_VALUE	Decimal	Mandatory. Set by TAP input grammar.
TOTAL_TAX_VALUE	Decimal	Calculated. Auto-generated.
TAP_TOTAL_TAX_VALUE	Decimal	Mandatory. Set by TAP input grammar.
TAP_TOTAL_DISCOUNT_VALUE	Decimal	Mandatory. Set by TAP input grammar.

Table 34–46 (Cont.) Trailer Record Fields

Name	Format	Description
OPERATOR_SPECIFIC_INFO	String	Stores a key to identify the CDR used to generate a specific EDR. Useful for RAP or CIBER return. Optional. Default = ' ' Must be set by an iScript.
CIBER_FILLER	String	Optional.
CREATION_TIMESTAMP	Date	Optional. See the CIBER specification for usage.
CIBER_RECORD_TYPE	String	Optional. See the CIBER specification for usage.
SETTLEMENT_PERIOD	String	Optional. See the CIBER specification for usage.
CLEARINGHOUSE_ID	String	Optional. See the CIBER specification for usage.
CURRENCY	String	Optional. See the CIBER specification for usage.
SENDING_CLEARINGHOUSE_BID	String	Optional. See the CIBER specification for usage.
CIBER_R70 BATCH_TOTALS_SIGN	String	Optional. See the CIBER specification for usage.
CIBER_R70 ORIGINAL_TOTALS_SIGN	String	Optional. See the CIBER specification for usage.
ORIGINAL_SEQUENCE_NUMBER	Integer	Optional. See the CIBER specification for usage.
ORIGINAL_CREATION_TIMESTAMP	Date	Optional. See the CIBER specification for usage.
ORIGINAL_TOTAL_NUMBER_OF_ RECORDS	Integer	Optional. See the CIBER specification for usage.
ORIGINAL_TOTAL_WHOLESALE_ CHARGED_VALUE	Decimal	Optional. See the CIBER specification for usage.
NOTIFICATION_END_INDEX	Integer	Notification block end index.
AUDIT_CONTROL_INFO_START_ INDEX	Integer	AuditControlInfo block start index.
AUDIT_CONTROL_INFO_END_INDEX	Integer	AuditControlInfo block end index.
DELAYED_ERROR_BLOCK	String	Stores the block name that has the fatal error.
TOTAL_CHARGE_VALUE_LIST	Block	*n* times. Mandatory. The TAP record is used by GSM operators and data clearinghouses to exchange roaming information.
TOTAL_CHARGE_VALUE	Decimal	Mandatory. Set by TAP grammar.
CHARGE_TYPE	String	Mandatory. Set by TAP grammar.
TOTAL_CHARGE_REFUND	String	Optional.
TOTAL_CHARGE_REFUND	Decimal	Optional.

Associated UTCOffset Record

Table 34–47 lists the fields in the Associated UTCOffset Record.

Table 34–47 Associated UTCOffset Record Fields

Name	Format	Description
UTCTIMEOFFSETCODE	Integer	Stores the UTC time offset code from TAP header.
UTCTIMEOFFSET	String	Stores the UTC time offset value from TAP header.

Associated Recentity Record

Table 34–48 lists the fields in the Associated Recentity Record.

Table 34–48 Associated Recentity Record Fields

Name	Format	Description
RECENTITYCODE	Integer	Stores the REC entity code from TAP header.
RECENTITYTYPE	Integer	Stores the REC entity type from TAP header.
RECENTITYID	String	Stores the REC entity ID from TAP header.

TAP Total Charge Value List

Mandatory. 0 .. *N* times.

Table 34–49 lists the TAP Total Charge Value List fields.

Table 34–49 TAP Total Charge Value List Fields

Name	Format	Description
TOTAL_CHARGE_VALUE	Decimal	Set by TAP grammar. Mandatory.
CHARGE_TYPE	String	Set by TAP grammar. Mandatory.
TOTAL_CHARGE_REFUND	Decimal	Optional.

Internal Service Control Container

This record is used internally by the framework.

Table 34–50 lists the fields in the Internal Service Control Container.

Table 34–50 Internal Service Controller Container Fields

Name	Format	Description
STREAM_NAME	String	Calculated.
OFFSET_ GENERATION	Integer	Calculated.
SEQ_CHECK	Integer	Calculated.
SEQ_GENERATION	Integer	Calculated.
TRANSACTION_ID	Decimal	Calculated.
PROCESS_STATUS	Integer	Values: 0: Normal (default) 1: Recycling 2: Recycling test Mandatory. Calculated.

Customer Data Record

This record is used internally to save all customer-related attributes along with an event.

Table 34–51 lists the fields in the Customer Data Record.

Table 34–51 Customer Data Record Fields

Name	Format	Description
ACCOUNT_ID	String	Mandatory.
ACCOUNT_PARENT_ID	String	Optional.
ACCOUNT_NO	String	Mandatory.
CREATION_DATE	Date	Mandatory.
CURRENCY	String	Mandatory.
CUST_SEG_LIST	String	Mandatory.
RESIDENCE_TYPE	String	Optional.
SYSTEM_BRAND	String	Optional.
BILL_CYCLE	String	Values: 00-28 Mandatory.
BILL_FREQUENCY	Integer	Values: 1-12 Mandatory.
PAYMENT_TYPE	String	Optional.
BILL_STATE	Integer	Mandatory.
ACTG_LAST_DATE	Date	Mandatory.
ACTG_NEXT_DATE	Date	Mandatory.
ACTG_FUTURE_DATE	Date	Mandatory.
ACTG_USED_DATE	Date	Mandatory.
BILL_LAST_DATE	Date	Mandatory.
BILL_NEXT_DATE	Date	Mandatory.
BILL_FUTURE_DATE	Date	Mandatory.
RESOURCE_LIST	String	Optional.
LEAST_COST	Integer	Optional.
PROMOTIONAL_SAVING	Integer	Optional.
PROMOTION	Integer	Optional.

Purchased Products

Table 34–52 lists the fields in the Purchase Products block.

Table 34–52 Purchased Products Fields

Name	Format	Description
PRODUCT_NAME	String	Mandatory.
USAGE_START	Date	Optional.
USAGE_END	Date	Optional.
QUANTITY	Decimal	Optional.

Table 34–52 (Cont.) Purchased Products Fields

Name	Format	Description
OFFERING_POID	String	Optional.
OVERRIDDEN_OFFERING_POID	String	Optional.
NODE_LOCATION	String	Mandatory.
DEAL_NAME	String	Mandatory. **Important:** The DEAL_NAME value is not stored in memory nor retained in the EDR. Therefore, this value will not appear in the EDR dump.
PLAN_NAME	String	Mandatory.
PRODUCT_TYPE	Integer	Defines system or normal product; for example, 603 or 602.
RATEPLAN_NAME	String	Mandatory.
PRIORITY	Integer	Mandatory.
PRODUCT_ID	String	Optional.
SERVICE_TYPE	String	For example, **/service/telco/gsm/data**. Mandatory.
SERVICE_ID	String	Optional.
SERVICE_PROMO_CODE	String	Optional.
SERVICE_VENDOR	String	Optional.
SERVICE_SOURCE	String	Optional.
SERVICE_LOGIN	String	Mandatory.
SERVICE_MSISDN	String	Optional.
SERVICE_IMSI	String	Optional.
SERVICE_STATUS	String	Mandatory.
SERVICE_USED_ITEM_POID	String	Mandatory.
NETWORK_IDENT	String	Optional.
FIRST_USAGE_INDICATOR	Integer	Specifies whether the product is configured to start when first used and the first-usage validity period status. Optional.

Extended Rating Attributes List

Table 34–53 lists the fields in the Extended Rating Attributes List.

Table 34–53 Extended Rating Attributes List Fields

Name	Format	Description
PROFILE	String	Mandatory.
LABEL	String	Optional.

Profile Attributes

Table 34–54 lists the Profile Attributes fields.

Table 34–54 Profile Attributes Fields

Name	Format	Description
KEY	String	Mandatory.
VALUE	String	Mandatory.

Alias List

Table 34–55 lists the Alias list field.

Table 34–55 Alias List Field

Name	Format	Description
ALIAS_NAME	String	Mandatory.

Discount List

Table 34–56 lists the Discount List fields.

Table 34–56 Discount List Fields

Name	Format	Description
BALANCE_GROUP_ID	String	Mandatory.
DISCOUNT_OWNER_ACCT_ID	String	Mandatory.
DISCOUNT_OWNER_ID	String	Mandatory.
DISCOUNT_OWNER_TYPE	String	Mandatory.

Purchased Discounts

Table 34–57 lists the Purchased Discounts fields.

Table 34–57 Purchased Discounts Fields

Name	Format	Description
DISCOUNT_ID	String	Mandatory.
DISCOUNT_MODEL	String	Mandatory.
PURCHASE_START	Date	Mandatory.
PURCHASE_END	Date	Mandatory.
USAGE_START	Date	Optional.
USAGE_END	Date	Optional.
PRIORITY	Integer	Mandatory.
MODE	Integer	Mandatory.
VALID_FLAG	Integer	Mandatory.
TYPE	Integer	Mandatory.
OFFERING_POID	String	Mandatory.
NODE_LOCATION	String	Mandatory.
STATUS	Integer	Mandatory.
QUANTITY	Integer	Mandatory.

Table 34–57 (Cont.) Purchased Discounts Fields

Name	Format	Description
FLAGS	Integer	Mandatory.
SCALE	Decimal	Mandatory.
FIRST_USAGE_ INDICATOR	Integer	Specifies whether the product is configured to start when first used and the first-usage validity period status. Optional.

Sponsor List

Table 34–58 shows the Sponsor List fields.

Table 34–58 Sponsor List Fields

Name	Format	Description
BALANCE_GROUP_ID	String	Mandatory.
SPONSOR_OWNER_ACCT_ID	String	Mandatory.
SPONSOR_OWNER_ID	String	Mandatory.
SPONSOR_OWNER_TYPE	String	Mandatory.

Sponsorship Details

Table 34–59 lists the Sponsorship Details fields.

Table 34–59 Sponsorship Details Fields

Name	Format	Description
SPONSORSHIP_ID	String	Mandatory.
DISCOUNT_MODEL	String	Mandatory.
VALID_FLAG	Integer	Mandatory.

Plan List

Table 34–60 lists the Plan List field.

Table 34–60 Plan List Field

Name	Format	Description
PLAN_ID	String	Mandatory.

Balance Group

Table 34–61 lists the Balance Group field.

Table 34–61 Balance Group Field

Name	Format	Description
BALANCE_GROUP_ID	String	Mandatory.

Balance Element

Table 34–62 lists the Balance Element fields.

Table 34–62 Balance Element Fields

Name	Format	Description
RESOURCE_ID	Integer	Mandatory.
CURR_BAL	Decimal	Mandatory.
CREDIT_FLOOR	Decimal	Mandatory.
CREDIT_LIMIT	Decimal	Mandatory.
RESERVED_AMOUNT	Decimal	Mandatory.
Name	Format	Description

Associated CIBER Extension Record

See the CIBER 2.5 specification for explanations of the fields listed in Table 34–63.

Table 34–63 Associated CIBER Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	Default = 701 (this is not yet the final value). Mandatory.
RECORD_NUMBER	Integer	Auto-generated. Mandatory.
FILLER	String	The filler for CIBER optional fields at the end of a record. Optional.
NO_OCC	Integer	Flag to suppress OCC (type 50 or 52) record-creation process. Optional.
INTERN_MOBILE_ID_NO	String	None
INTERN_CALLED_NO	String	None
INTERN_MSISDN_MDN	String	None
INTERN_CALLER_ID	String	None
INTERN_ROUTING_NO	String	None
INTERN_TLDN_NO	String	None
CIBER_RECORD_TYPE	String	Optional.
RETURN_CODE	String	Optional.
RETURN_REASON_CODE	String	Optional.
INVALID_FIELD_ID	String	Optional.
HOME_CARRIER_SID	String	Optional.
MOBILE_ID_NO_LENGTH	Integer	Optional.
MOBILE_ID_NO	String	Optional.
MOBILE_ID_NO_OVERFLOW	String	Optional.
ELECTRONIC_SERIAL_NO	String	Optional.
CALL_DATE	Date	Optional.
SERVING_CARRIER_SID	String	Optional.

Table 34–63 (Cont.) Associated CIBER Extension Record Fields

Name	Format	Description
TOTAL_CHARGE_AND_TAX	Decimal	Optional.
TOTAL_STATE_TAX	Decimal	Optional.
TOTAL_LOCAL_TAX	Decimal	Optional.
CALL_DIRECTION	String	Optional.
CALL_COMPLETION_INDICATOR	String	Optional.
CALL_TERMINATION_INDICATOR	String	Optional.
CALLED_NO_LENGTH	Integer	Optional.
CALLED_NO	String	Optional.
CALLED_NO_OVERFLOW	String	Optional.
TEMP_LOCAL_DIRECTORY_NO	String	Optional.
CURRENCY_TYPE	String	Optional.
ORIG_BATCH_SEQ_NO	Integer	Optional.
INITIAL_CELL_SITE	String	Optional.
TIME_ZONE_INDICATOR	String	Optional.
DAYLIGHT_SAVINGS_INDICATOR	String	Optional.
MSG_ACCOUNTING_DIGITS	String	Optional.
SSU_CONNECT_TIME	Date	Optional.
SERVING_STATE	String	Optional.
RECV_CARRIER_SID	String	Optional.
TRANS_CODE1	String	Optional.
TRANS_CODE2	String	Optional.
SENDING_CARRIER_SID	String	Optional.
CHARGE_NO_1_IND	String	Optional.
CHARGE_NO_1_CONNECT_TIME	Date	Optional.
CHARGE_NO_1_CHARGEABLE_TIME	String	Optional.
CHARGE_NO_1_ELAPSED_TIME	String	Optional.
CHARGE_NO_1_RATE_PERIOD	String	Optional.
CHARGE_NO_1_MULTIRATE_PERIOD	String	Optional.
CHARGE_NO_1	Decimal	Optional.
CHARGE_NO_2_IND	String	Optional.
CHARGE_NO_2_CONNECT_TIME	Date	Optional.
CHARGE_NO_2_CHARGEABLE_TIME	String	Optional.
CHARGE_NO_2_ELAPSED_TIME	String	Optional.
CHARGE_NO_2_RATE_PERIOD	String	Optional.
CHARGE_NO_2_MULTIRATE_PERIOD	String	Optional.
CHARGE_NO_2	Decimal	Optional.
CHARGE_NO_3_IND	String	Optional.

Table 34–63 (Cont.) Associated CIBER Extension Record Fields

Name	Format	Description
CHARGE_NO_3_CONNECT_TIME	Date	Optional.
CHARGE_NO_3_CHARGEABLE_TIME	String	Optional.
CHARGE_NO_3_ELAPSED_TIME	String	Optional.
CHARGE_NO_3_RATE_PERIOD	String	Optional.
CHARGE_NO_3_MULTIRATE_PERIOD	String	Optional.
CHARGE_NO_3	Decimal	Optional.
CHARGE_NO_4_IND	String	Optional.
CHARGE_NO_4_CONNECT_TIME	Date	Optional.
CHARGE_NO_4_CHARGEABLE_TIME	String	Optional.
CHARGE_NO_4_ELAPSED_TIME	String	Optional.
CHARGE_NO_4_RATE_PERIOD	String	Optional.
CHARGE_NO_4_MULTIRATE_PERIOD	String	Optional.
CHARGE_NO_4	Decimal	Optional.
CHARGE_NO_1_SURCHARGE_IND	String	Optional.
CHARGE_NO_2_SURCHARGE_IND	String	Optional.
CHARGE_NO_3_SURCHARGE_IND	String	Optional.
CHARGE_NO_4_SURCHARGE_IND	String	Optional.
TOLL_CONNECT_TIME	Date	Optional.
TOLL_CHARGEABLE_TIME	String	Optional.
TOLL_ELAPSED_TIME	String	Optional.
TOLL_TARIFF_DESC	String	Optional.
TOLL_RATE_PERIOD	String	Optional.
TOLL_MULTIRATE_PERIOD	String	Optional.
TOLL_RATE_CLASS	String	Optional.
TOLL_FROM_RATING_NPA_NXX	String	Optional.
TOLL_CHARGE	Decimal	Optional.
TOLL_STATE_TAX	Decimal	Optional.
TOLL_LOCAL_TAX	Decimal	Optional.
TOLL_NETWORK_CARRIER_ID	String	Optional.
MSID_INDICATOR	String	Optional.
MSID	String	Optional.
MSISDN_MDN_LENGTH	Integer	Optional.
MSISDN_MDN	String	Optional.
ESN_IMEI_INDICATOR	String	Optional.
ESN_IMEI	String	Optional.
CALLER_ID_LENGTH	Integer	Optional.
CALLER_ID	String	Optional.

Table 34–63 (Cont.) Associated CIBER Extension Record Fields

Name	Format	Description
ROUTING_NO_LENGTH	Integer	Optional.
ROUTING_NO	String	Optional.
TLDN_NO_LENGTH	Integer	Optional.
TLDN_NO	String	Optional.
AIR_CONNECT_TIME	Date	Optional.
AIR_CHARGEABLE_TIME	String	Optional.
AIR_ELAPSED_TIME	String	Optional.
AIR_RATE_PERIOD	String	Optional.
AIR_MULTIRATE_PERIOD	String	Optional.
AIR_CHARGE	Decimal	Optional.
OTHER_CHARGE_1_INDICATOR	String	Optional.
OTHER_CHARGE_1	Decimal	Optional.
CALLED_COUNTRY	String	Optional.
SERVING_COUNTRY	String	Optional.
TOLL_RATING_POINT_LENGTH	Integer	Optional.
TOLL_RATING_POINT	String	Optional.
FEATURE_USED_AFTER_HO_IND	String	Optional.
OCC_START_DATE	Date	Optional.
OCC_CHARGE	Decimal	Optional.
FET_EXEMPT_INDICATOR	String	Optional.
PASS_THROUGH_CHARGE_IND	String	Optional.
CONNECT_TIME	Date	Optional.
RECORD_USE_INDICATOR	String	Optional.
OCC_DESCRIPTION	String	Optional.
OCC_END_DATE	Date	Optional.
RECORD_CREATE_DATE	Date	Optional.
SEQ_INDICATOR	String	Optional.
OCC_INTERVAL_INDICATOR	String	Optional.
EVENT_DATE	Date	Optional.
MIN_ESN_APP_INDICATOR	String	Optional.
R70_RECORD_USE_INDICATOR	String	Optional.
EVENT_TIME	Date	Optional.

Discount Balance Packet

Table 34–64 lists the Discount Balance Packet fields.

Table 34–64 Discount Balance Packet Fields

Name	Format	Description
DISCOUNT_KEY	String	Discount key (normally the account ID).
ACCOUNT_ID	Integer	Related account ID.
RESOURCE_ID	Integer	BRM mapped resource ID.
DISCOUNT_STEP	Integer	Alternative key if resource ID 0.
DISCOUNT_MASTER	Integer	Alternative key if resource ID 0.
UPDATE_LEVEL	String	Always empty. Supports backward compatibility.
SERVICE_ID	Integer	Supports backward compatibility.

Aggregation Period

Table 34–65 lists the Aggregation Period fields.

Table 34–65 Aggregation Period Fields

Name	Format	Description
PERIOD	String	The accounting cycle. *YYYYMMDD*.
ITEM_POID	String	POID of the item that identifies the accounting cycle.
CREATED	String	Creation date.
TOTAL_CHARGE	Decimal	Total charge.
TOTAL_QUANTITY	Decimal	Total quantity based on RUMs in the discount filter.
TOTAL_EVENT	Decimal	Sum of events based on charge packets.
GRANTED_CHARGE	Decimal	The discounted charge.
GRANTED_QUANTITY	Decimal	The discounted quantity.
FRAME_CHARGE	Decimal	Total charge of the discount frame, based on the frame.
FRAME_QUANTITY	Decimal	Total quantity in the discount frame based on RUMs in the discount filter.
FRAME_EVENT	Decimal	Total events of the discount frame, based on charge packets.

Discount Packet

Table 34–66 lists the Discount Packet fields.

Table 34–66 Discount Packet Fields

Name	Format	Description
RECORD_TYPE	String	Mandatory.
CREATED	String	Creation date.
OBJECT_ID	String	Discount/sponsor object ID.
OBJECT_TYPE	String	Discount/sponsor object that generated the event.
OBJECT_ACCOUNT	Integer	POID of the discount owner.
OBJECT_OWNER_ID	Integer	POID type of the discount owner.
OBJECT_OWNER_TYPE	String	POID type of the discount owner.

Table 34–66 (Cont.) Discount Packet Fields

Name	Format	Description
DISCOUNTMODEL	String	Discount model.
DISCOUNTRULE	String	Discount rule.
DISCOUNTSTEPID	Integer	Discount step ID.
DISCOUNTBALIMPACTID	Integer	Discount balance impact ID.
TAX_CODE	String	Tax code.
GRANTED_AMOUNT	Decimal	Granted discount/sponsorship amount. Can be currency or non-currency.
GRANTED_AMOUNT_ORIG	Decimal	Original granted discount/sponsorship amount. Used when exchange rate is configured.
GRANTED_QUANTITY	Decimal	The discount base value used to compute the granted amount.
AMOUNT	Decimal	Discounted currency amount.
PIN_PERCENT	Decimal	Percent value filled from charge packet
QUANTITY	Decimal	Discounted non-currency amount.
QUANTITY_FROM	Decimal	Discounted quantity start value.
QUANTITY_TO	Decimal	Discounted quantity end value.
VALID_FROM	Date	Grant case, valid-from date.
VALID_TO	Date	Grant case, valid-to date.
VALID_FROM_DETAIL	Integer	First Usage Offset and Unit for valid-from date.
VALID_TO_DETAIL	Integer	First Usage Offset and Unit for valid-to date.
CYCLE_OFFSET	Integer	Product cycle that identifies a grant's validity period.
BALANCE_GROUP_ID	Integer	Balance group ID.
SERVICE_CODE	String	Service code.
RESOURCE_ID	Integer	Resource ID.
RESOURCE_ID_ORIG	Integer	Original resource ID. Used when exchange rate is configured.
ZONEMODEL_CODE	String	Zone model.
IMPACT_CATEGORY	String	Impact category.
TIMEZONE_CODE	String	Time-zone code.
TIMEMODEL_CODE	String	Time model code.
SERVICE_CLASS	String	Service class.
PRICEMODEL_CODE	String	Price model code.
RUM	String	RUM.
RATETAG	String	Rate tag.
RATEPLAN	String	Rate plan.
GLID	String	G/L ID.
OFFERING_POID	String	Purchased discount POID.
NODE_LOCATION	String	Node location.
INTERN_PACKET_INDEX	Integer	Packet ID.
INTERN_SRC_PACKET_INDEX	Integer	Source packet ID.

Table 34–66 (Cont.) Discount Packet Fields

Name	Format	Description
INTERN_RUM_ID	Integer	RUM ID passed in and out, used in real-time pipeline only.
INTERN_DISC_MATCH_ FACTOR	Decimal	Discount match factor (the percentage of usage discounted).
INTERN_TOTAL_MATCH_ FACTOR	Decimal	Total discounted match factor (the total percentage of usage discounted).
DEFERRED_AMOUNT	Decimal	Deferred amount.

Discount Sub-Balance Packet

Table 34–67 lists the Discount Sub-Balance Packet fields.

Table 34–67 Discount Sub-Balance Packet Fields

Name	Format	Description
REC_ID	Integer	Record ID.
VALID_FROM	Date	Validity start time.
VALID_TO	Date	Validity end time.
AMOUNT	Decimal	Amount.
CONTRIBUTOR	String	Contributor.
NEXT_BAL	Date	Next balance.
DELAYED_BAL	Decimal	Delayed balance.
GRANTOR	String	The product or discount that granted this resource.
VALID_FROM_DETAILS	Date	Sub-balance start time mode (such as first-usage or relative) and relative offset and unit.
VALID_TO_DETAILS	Date	Sub-balance end time mode (such as relative) and relative offset and unit.
GRANT_VALID_FROM	Date	Grant validity start time.
GRANT_VALID_TO	Date	Grant validity end time.

Associated SMS Extension Record

Table 34–68 lists the Associated SMS Extension Record fields.

Table 34–68 Associated SMS Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	Mandatory. Default = 580.
RECORD_NUMBER	Integer	Mandatory. Auto-generated.
CONTENT_INDICATOR	String	Optional.
ORIGINATING_SWITCH_ IDENTIFICATION	String	Optional.
DESTINATION_SWITCH_ IDENTIFICATION	String	Optional.

Table 34–68 (Cont.) Associated SMS Extension Record Fields

Name	Format	Description
PROVIDER_ID	String	Optional.
SERVICE_ID	String	Optional.
DEVICE_NUMBER	String	Optional.
PORT_NUMBER	String	Optional.
DIALED_DIGITS	String	Optional.

Associated MMS Extension Record

Table 34–69 lists the Associated MMS Extension Record fields.

Table 34–69 Associated MMS Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	Mandatory. Default = 590.
RECORD_NUMBER	Integer	Mandatory. Auto-generated.
ACCOUNT_STATUS_TYPE	String	Optional.
PRIORITY	String	Optional.
MESSAGE_CONTENT	String	Optional.
MESSAGE_ID	String	Optional.
STATION_IDENTIFIER	String	Optional.
FC_INDICATOR	String	Optional.
CORRELATION_ID	String	Optional.
CELL_ID	String	Optional.
B_CELL_ID	String	Optional.
A_TERM_CELL_ID	String	Optional.
DEVICE_NUMBER	String	Optional.
PORT_NUMBER	String	Optional.

SGSN Information

Table 34–70 lists the SGSN Information field.

Table 34–70 SGSN Information Field

Name	Format	Description
SGSN_ADDRESS	String	Mandatory.

Profile Event Ordering

Table 34–71 lists the Profile Event Ordering fields.

Table 34–71 Profile Event Ordering Fields

Name	Format	Description
RECORD_TYPE	String	Mandatory. Must be set to 850.
RECORD_NUMBER	Integer	Mandatory.
BAL_GRP_POID	String	Mandatory.
CRITERIA_NAME	String	Mandatory.
PROFILE_POID	String	Mandatory.
BILLING_CYCLE_TIMESTAMP	Date.	Mandatory.

Associated Roaming Extension Record

Table 34–72 lists the Associated Roaming Extension Record fields.

Table 34–72 Associated Roaming Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	None
RECORD_NUMBER	Integer	None
TAP_FILE_SEQ_NO	Integer	None
RAP_FILE_SEQ_NO	Integer	None
RAP_RECORD_TYPE	String	None
SENDER	String	None
RECIPIENT	String	None
TAP_FILE_PATH	String	None
START_MISSING_SEQ_NUM	Integer	None
END_MISSING_SEQ_NUM	Integer	None
SUSPENSION_TIME	Date	None
PORT_NUMBER	String	None
TOTAL_TAX_REFUND	Decimal	None
TOTAL_DISCOUNT_REFUND	Decimal	None
GUARANTEED_BIT_RATE	String	None
MAXIMUM_BIT_RATE	String	None
HSCSD_INDICATO	String	None
SMS_ORIGINATOR	String	None
SMS_DESTINATION_NUMBER	String	None
DISCOUNTABLE_AMOUNT	Decimal	None
DISCOUNT_CODE	Integer	None
NETWORKACCESS_IDENTIFIE	String	None
ISM_SIGNALLING_CONTEXT	Integer	None
IMSI	String	None

Table 34–72 (Cont.) Associated Roaming Extension Record Fields

Name	Format	Description
HOME_BID	String	None
HOMELOCATION_DESCRIPTION	String	None
MOBILE_ID_NUMBER	String	None
MOBILE_DIR_NUMBER	String	None
TOTAL_ADVISEDCHARGE	Decimal	None
TOTAL_ADVISEDCHARGE_ REFUND	Decimal	None
TOTAL_COMMISSION	Decimal	None
TOTAL_COMMISSION_REFUND	Decimal	None
ITEM_OFFSET	Integer	None
ERROR_CODE	Integer	None
TOTAL_SEVERE_RETURN_VALUE	Decimal	None
RETURN_DETAILS_COUNT	Integer	None
CLIR_INDICATOR	String	None
TAP_CURRENCY	String	Currency used for TAP3 and TAP 311.

Associated RAP Extension

Table 34–73 lists the Associated RAP Extension fields.

Table 34–73 Associated RAP Extension Fields

Name	Format	Description
PATH_ITEMID	Integer	None
ITEM_OCCURRENCE	Integer	None
ITEM_LEVEL	Integer	None

Total Advised Charge Value List

Table 34–74 lists the Total Advised Charge Value List fields.

Table 34–74 Total Advised Charge Value List Fields

Name	Format	Description
TOTAL_ADVISEDCHARGE	Decimal	None
TOTAL_ADVISEDCHARGE_REFUND	Decimal	None
ADVISED_CHARGE_CURRENCY	String	Optional. AdvisedChargeCurrency item.
TOTAL_COMMISSION;	Decimal	None
TOTAL_COMMISSION_REFUND	Decimal	None

Field Usage

Roaming

The following conditions are checked to determine if the usage record is for roaming:

```
if [DETAIL.USAGE_DIRECTION] = 2
       then Roaming = TRUE; Roaming-Type = MOC
elseif [DETAIL.USAGE_DIRECTION] = 3
       then Roaming = TRUE; Roaming-Type = MTC
else   Roaming = FALSE
```

International-Call

The following conditions are checked to determine if the usage record is for an international call:

```
if [DETAIL.CONNECT_SUB_TYPE] = '04'
       then International-Call=TRUE
```

CLI Normalization

The result of the mapping operation (performed either within an input module or within an iScript) must be written/copied to the following internal container fields:

DETAIL.A_NUMBER -> normalize -> DETAIL.INTERN_A_NUMBER_ZONE

DETAIL.B_NUMBER -> normalize -> DETAIL.INTERN_B_NUMBER_ZONE

DETAIL.C_NUMBER -> normalize -> DETAIL.INTERN_C_NUMBER_ZONE

The original values within the Basic Detail Record are kept unchanged.

The following fields determine how and which a normalization function should be carried out:

DETAIL.A_NUMBERING_PLAN -> normalize DETAIL.A_NUMBER

DETAIL.B_NUMBERING_PLAN -> normalize DETAIL.B_NUMBER

DETAIL.C_NUMBERING_PLAN -> normalize DETAIL.C_NUMBER

if DETAIL.x_NUMBERING_PLAN between 1 and 9 -> use "ISDN, MSISDN"

if DETAIL.x_NUMBERING_PLAN between A and B -> use "Ipv4, IPv6"

if DETAIL.x_NUMBERING_PLAN = 0 -> no normalization

ISDN, MSISDN

The following rules normalize the A number and B number. All CLIs are normalized to match the international format: *International_access_codeCountry_codeNational_destination_codeSubscriber_number*; for example, '00491711234567' or '004980012345'.

To handle a flexible international format, the following parameters must be set prior to the normalization:

International_access_code (iac): international access code; for example, '00'

> **Note:** Multiple iacs might be defined.

International_access_code_sign (iacs): international access code sign; for example, '+'

Country_code (cc): country code of the home country; for example, '49'

National_destination_access_code (ndac): national destination access code for long distance; for example, '0'

National_destination_code (ndc): default national destination code; for example, '172' (only for special mobile calls)

Normal-Call

Table 34–75 lists the Normal-Call fields.

Table 34–75 Normal-Call Fields

Item	Description
A#:	[DETAIL.A_NUMBER, X(40)]
B#:	[DETAIL.B_NUMBER, X(40)]

1. if <empty>-> replace with '<iac><cc>' (break)
2. if Prefix = '<iacs>'-> replace with '<iac>' (continue)
3. if Prefix = '<iac>'-> do nothing (break)
4. if Prefix = '<ndac>'-> replace with '<iac><cc>' (break)
5. if [TYPE_OF_NUMBER] = 1-> prefixing '<iac>' (break)

Roaming-Call (Mapping only for Zone- and PrefixDesc.-Determination)

An Associated GSM/Wireline Extension Record must exist.

Table 34–76 lists the Roaming-Call fields.

Table 34–76 Roaming-Call Fields

Item	Type	Description
A#:	MOC	-> '0000' + Left([ASS_GSMW_SE.ORIGINATING_SWITCH_ID], 5)
A#:	MTC	-> Normalization as for normal calls
B#:	MOC	-> Normalization as for normal calls
B#:	MTC	-> '0000' + Left([ASS_GSMW_SE.TERMINATING_SWITCH_ID], 5)

If on the input side there is only one single MSC_ID or PLMN_ID available, the related input module has to map this single value into both fields (Originating and Terminating).

Special-Mobile-Call:

A/B-MODIFICATION_INDICATOR = '04'

Normalize to: <iac><cc>'0'<ndc><number>; for example, '0049017222255'

Table 34–77 lists the Special-Mobile-Call fields.

Table 34–77 Special-Mobile-Call Fields

Item	Description
A#:	[DETAIL.A_NUMBER, X(40)]
B#:	[DETAIL.B_NUMBER, X(40)]

1. if <empty>-> replace with '<iac><cc>' (break)

2. if Prefix = '<iacs>'-> replace with '<iac>' (continue)
3. if Prefix = '<iac><cc>'-> replace with '<iac><cc>0' (break)
4. if Prefix = '<ndac>'-> prefixing '<iac><cc>' (break)
5. if Prefix != '<cc>'-> prefixing '<iac><cc>0<ndc>' (break)
6. if [TYPE_OF_NUMBER] = 1-> prefixing '<iac>' (break)

IPv4, IPv6

If the A# (source ip) and B# (destin ip) are carrying ip-addresses, they are normalized to zero-leading-tokens without the dots or colon; for example, '192.168.10.1' is normalized to '192168010001'.

Notations are listed in Table 34–78:

Table 34–78 IPv4 and IPv6 Attributes

Item	Type	Description
IPv4:	nnn.nnn.nnn.nnn	n = (0..9)
IPv6:	hhhh:hhhh:hhhh:hhhh:hhhh:hhhh:hhhh:hhhh	h = (0..F)
IPv4 as v6:	0000:0000:0000:0000:0000:0000:nnn.nnn.nnn.nnn	None
or	0000:0000:0000:0000:0000:FFFF:nnn.nnn.nnn.nnn	None

IPv4 Record

Table 34–79 lists the IPv4 Record fields.

Table 34–79 IPv4 Record Fields

Item	Type	Description
A#:	[DETAIL.A_NUMBER, X(40)]	(source ip-address)
B#:	[DETAIL.B_NUMBER, X(40)]	(destination ip-address)

1. determine the four decimal ip-tokens
2. fill each token with leading zeros (up to 3-digits)
3. remove all dots '.'

IPv6 Record

Table 34–80 lists the IPv6 Record fields.

Table 34–80 IPv6 Record Fields

Item	Type	Description
A#:	[DETAIL.A_NUMBER, X(40)]	(source ip-address)
B#:	[DETAIL.B_NUMBER, X(40)]	(destination ip-address)

1. determine the eight hexadecimal ip-tokens
2. fill each token with leading zeros (up to 4-digits)
3. remove all colons ':'

IPv4 as IPv6 Record

Table 34–81 lists the IPv4 vs IPv6 Record fields.

Table 34–81 IPv4 as IPv6 Record Fields

Item	Type	Description
A#:	[DETAIL.A_NUMBER, X(40)]	(source ip-address)
B#:	[DETAIL.B_NUMBER, X(40)]	(destination ip-address)

1. determine the six fix hexadecimal v6 ip-tokens
2. fill each token with leading zeros (up to 4-digits)
3. remove all colons ':'
4. determine the four decimal ip-tokens at the end of the address
5. take the first two v4 tokens and convert them to hexadecimal (for example, '192.168' = 192*256^1 + 168*256^0 = 49320 = 'C0A8')
6. take the last two v4 tokens and convert them to hexadecimal
7. replace the origin v4 address by the two calculated v6 equivalents filled with leading zeros

UsageClass (CallClass)

The result of the mapping operation (performed either within an input module or within an early iScript) must be written/copied to the following internal container fields:

DETAIL.USAGE_CLASS -> mapping -> DETAIL.INTERN_USAGE_CLASS

The original values within the Basic Detail Record are kept unchanged.

The following rules apply to determine the external UsageClass value:

Always: Value of the field [DETAIL.USAGE_CLASS]

ServiceCode / ServiceClass

The result of the mapping operation (performed either within an input module or within an early iScript) must be written/copied to the following internal container fields:

DETAIL.BASIC_SERVICE -> mapping -> DETAIL.INTERN_SERVICE_CODE

mapping -> DETAIL.INTERN_SERVICE_CLASS

The original values within the Basic Detail Record are kept unchanged.

The following rules apply to determine the external ServiceCode value:

Always: Value of the field [DETAIL.BASIC_SERVICE];

containing [SERVICE_TYPE, X(1)]+ [SERVICE_CODE, X(2)]

35

List of Pipeline Manager Modules, iScripts, and iRules

This chapter lists the Pipeline Manager modules.

For information about pipeline rating, see "About Pipeline Rating".

For information about placement of modules in a pipeline, see "Function Module Dependencies".

Pipeline Manager Modules

Table 35–1 lists the Pipeline Manager modules with descriptions.

Table 35–1 Pipeline Manager Modules

Module	Description
Controller	Controls and monitors the pipeline framework. See "About the Controller" in *BRM Concepts*.
Database Connect (DBC)	Provides database connections for other modules.
DAT_AccountBatch	Provides customer data from the BRM database. See: ■ "Adding Customer Balance Impact Data to EDRs" in *BRM Setting Up Pricing and Rating* ■ Using Pipeline Manager with Multiple Database Schemas
DAT_AccountRealtime	Provides data to a real-time discounting pipeline. See "Configuring a Real-Time Discounting Pipeline".
DAT_BalanceBatch	Maintains balance information in the Pipeline Manager memory. See "Configuring Discounting Modules and Components".
DAT_BalanceRealtime	Retrieves current balance information from the BRM database and supplies the data to the real-time discounting pipeline. See "Configuring a Real-Time Discounting Pipeline".
DAT_Calendar	Provides holiday calendar data for the FCT_MainRating module. See "Rating by Date and Time with Pipeline Manager" in *BRM Setting Up Pricing and Rating*.
DAT_Currency	Converts currency symbols to numeric values. See "Setting up Pipeline Manager Resources" in *BRM Setting Up Pricing and Rating*.
DAT_Dayrate	Provides special day rate data for the FCT_Dayrate module. See "About Special Day Rates" in *BRM Setting Up Pricing and Rating*.

Table 35–1 (Cont.) Pipeline Manager Modules

Module	Description
DAT_Discount	Provides data for the FCT_Discount module and the FCT_DiscountAnalysis module. See "Configuring Discounting Modules and Components".
DAT_ExchangeRate	Provides currency exchange rate data for the FCT_ExchangeRate module. See "Defining Currency Exchange Rates" in *BRM Setting Up Pricing and Rating*.
DAT_InterConnect	Provides network configuration data for the FCT_CarrierIcRating module. See "Configuring DAT_InterConnect" in *BRM Configuring Roaming in Pipeline Manager*.
DAT_ItemAssign	Returns the item POID for an item tag to the FCT_ItemAssign and FCT_Billing Record modules. See "Creating Custom Bill Items" in *BRM Configuring and Running Billing*.
DAT_Listener	Listens to business events from BRM and provides data to the DAT_AccountBatch and DAT_Discount modules. See "Installing and Configuring the Account Synchronization DM" in *BRM Installation Guide*.
DAT_ModelSelector	Provides model selector rules to other modules. See "Configuring Pipeline Rating" and "Configuring Discounting Modules and Components".
DAT_NOSP	Provides data for mapping network source and destinations to new values for the FCT_NOSP module, used for multi-segment rating. See the following documents: ■ Identifying the Network Operator/Service Provider ■ About Multi-Segment Rating
DAT_NumberPortability	Provides number portability data to the FCT_NumberPortability module. See "Setting Up Number Portability".
DAT_PortalConfig	Provides data for mapping phone number prefixes to descriptions, used by the FCT_PrefixDesc module. See "Creating Call Destination Descriptions".
DAT_PriceModel	Provides price model data for the FCT_MainRating module. See "About Pipeline Rating".
DAT_Rateplan	Provides rate plan data for the FCT_MainRating module. See "Configuring Pipeline Rating".
DAT_Recycle	Used by standard recycling and Suspense Manager EDR to recycle EDRS. See "Configuring Standard Recycling".
DAT_ScenarioReader	Provides aggregation scenario data for the FCT_AggreGate module. See "Setting Up Pipeline Aggregation".

Table 35–1 (Cont.) Pipeline Manager Modules

Module	Description
DAT_TimeModel	Provides time model, time zone, and day code data for the FCT_ Mainrating module. See "Rating by Date and Time with Pipeline Manager" in *BRM Setting Up Pricing and Rating*.
DAT_USC_Map	Provides usage scenario (USC) mapping data. It retrieves USC mapping data from the Pipeline Manager database or an ASCII file for the FCT_USC_Map module. See "Setting Up Usage Scenario Mapping" in *BRM Setting Up Pricing and Rating*.
DAT_Zone	Provides zone data for the FCT_MainRating module. See "Setting Up Zones for Batch Pipeline Rating" in *BRM Setting Up Pricing and Rating*.
EDR Factory	Generates and allocates memory to EDR Containers. See "About the EDR Factory" in *BRM Concepts*.
Event Handler	Starts external programs. See "Using Events to Start External Programs" in *BRM System Administrator's Guide*.
EXT_InEasyDB	Handles pipeline input from a database. See "Configuring EDR Input Processing". Configure this module as a submodule of the INP_GenericStream module. See INP_GenericStream.
EXT_InFileManager	Performs file handling for pipeline input from files. See "Configuring EDR Input Processing". Configure this module as a submodule of the INP_GenericStream module. See INP_GenericStream.
EXT_OutFileManager	Handles files for the OUT_Generic_Stream and OUT_Reject modules. See "Configuring EDR Output Processing".
Pipeline Dispatcher	Parses CDR files from a single input directory to multiple pipelines. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.
FCT_Account	Adds customer data to an EDR. See "Adding Customer Balance Impact Data to EDRs" in *BRM Setting Up Pricing and Rating*.
FCT_AccountRouter	For a multischema system, finds the database schema for the customer and routes the EDRs to the appropriate pipeline. See "Using Pipeline Manager with Multiple Database Schemas".
FCT_AggreGate	Performs aggregation of data in EDR containers. See "Setting Up Pipeline Aggregation".
FCT_APN_Map	**Before zoning**: Maps the access point name (APN) to a physical PDP address. **After zoning**: Enhances zone values to support enhanced zoning functionality. See "Setting Up APN Mapping" in *BRM Setting Up Pricing and Rating*.

Table 35–1 (Cont.) Pipeline Manager Modules

Module	Description
FCT_ApplyBalance	Reads the discount packets added by DAT_Discount, adds the discounting sub-balance impact to the EDR, and updates the in-memory balance. See "About Discounts".
FCT_BillingRecord	Consolidates balance impact data into an associated BRM billing record and one or more balance impact packets. This data is loaded into the BRM database by RE Loader. See "About Consolidation for BRM Billing".
FCT_CallAssembling	Assembles EDRs that have been split into multiple EDRs. See "Assembling EDRs".
FCT_CarrierIcRating	Adds roaming data to EDRs for rating by the FCT_PreRating and FCT_MainRating modules. See "About Linking Rate Plans to Network Operators and IC Products" in *BRM Configuring Roaming in Pipeline Manager*.
FCT_CiberOcc	The FCT_CiberOcc module creates a CIBER record for other charges and credits (OCC record), type 50 or 52. See "About Settling Roaming Charges" in *BRM Configuring Roaming in Pipeline Manager*.
FCT_CliMapping	Maps multiple numbers to a single number for billing. See "Mapping Multiple Phone Numbers to a Single Number".
FCT_CreditLimitCheck	Performs credit limit checking to determine whether the event owner has enough resources for the requested service. See "About Credit Limit Checks in the Real-Time Discounting Pipeline" in *BRM Telco Integration*.
FCT_CustomerRating	Supplies the rate plan for the FCT_MainRating module. See "About Customer Rating".
FCT_Dayrate	Calculates charges for special day rates, for example, a discount for calls made on January 1. See "About Special Day Rates" in *BRM Setting Up Pricing and Rating*.
FCT_Discard	Discards or skips EDRs based on configurable EDR properties. ■ Skipping an EDR removes it from the pipeline. ■ Discarding an EDR sends it to a different output stream. In both the cases the state of the EDR becomes invalid. See "Discarding and Skipping EDRs".
FCT_Discount	Performs discounting functions. See "Configuring Discounting Modules and Components".
FCT_DiscountAnalysis	Performs discounting analysis functions. See "Configuring Discounting Modules and Components".
FCT_DroppedCall	Identifies dropped calls and continuation calls. See "About Finding Dropped Calls and Continuation Calls" in *BRM Telco Integration*.
FCT_DuplicateCheck	Checks for duplicate EDRs. See "Handling Duplicate EDRs".

Table 35–1 (Cont.) Pipeline Manager Modules

Module	Description
FCT_EnhancedSplitting	Specifies different output streams for EDRs based on rules. For example: ■ You can split EDRs for different service types to different output streams. ■ You can split EDRs from roaming outcollects and incollects into different streams. See "Using Rules to Send EDRs to Different Output Streams".
FCT_ExchangeRate	Converts the currency used for rating to the home (system) currency, and the customer's billing currency. See "Defining Currency Exchange Rates" in *BRM Setting Up Pricing and Rating*.
FCT_Filter_Set	Determines whether an EDR is eligible for the system products and system discounts contained in a filter set, and if it is, adds those system products and discounts to a customer's list of purchased products. See "About Using Filter Sets to Apply System Products and Discounts".
FCT_GlobalRating	Rates all EDRs with a default set of rate plans. See "About Global Rating".
FCT_IRules	Evaluates iRules. Those rules can be used for mapping functions for EDR data fields, splitting EDR containers to different output streams, and so forth. See "About Configuring iRules" in *BRM System Administrator's Guide*.
FCT_IScript	Runs iScripts. The scripts are run in the order specified in the registry. See "About Configuring iScripts" in *BRM System Administrator's Guide*.
FCT_Reject	Retrieves an item POID for an item tag from the DAT_ItemAssign module and populates the EDR container with the item POID. See DAT_ItemAssign.
FCT_MainRating	Performs the main Pipeline Manager rating functionality. See "About Main Rating".
FCT_MainZoning	Performs zoning for multi-segment zoning. See "Setting Up Multi-Segment Zoning" in *BRM Setting Up Pricing and Rating*.
FCT_NOSP	Maps network source and destination to new values. See "Identifying the Network Operator/Service Provider".
FCT_NumberPortability	Specifies the new network operator for an existing phone number. See "Setting Up Number Portability".
FCT_PrefixDesc	Maps phone number prefixes to destination descriptions. See "Creating Call Destination Descriptions".
FCT_PreRating	Calculates zones and creates impact categories. See "Setting Up Prerating" in *BRM Setting Up Pricing and Rating*.

Table 35–1 (Cont.) Pipeline Manager Modules

Module	Description
FCT_PreRecycle	Used for pipeline-only implementations. Gets the file of rejected EDRs from the reject stream output directory. The module puts the reject EDR file into the pipeline input directory for recycling. It uses the same input folder as the incoming CDR files. See "Recycling EDRs in Pipeline-Only Systems".
FCT_PreSuspense	When used as part of BRM standard recycling, this module adds suspense-related information to EDRs. When used with Suspense Manager, this module also configures the queryable fields for EDRs suspended in a specific pipeline.
FCT_RateAdjust	Adjusts the charge for an EDR after rating has been performed. See "About Rate Adjustment" in *BRM Setting Up Pricing and Rating*.
FCT_Recycle	Used for pipeline-only implementations. Runs at the end of the pipeline It does either of the following: ■ When the FCT_PreRecycle module runs in test mode, the FCT_Recycle module creates a report about the processing, but does not send the EDRs to an output file. ■ When the FCT_PreRecycle module runs in recycle mode, the FCT_Recycle module sends the results to an output file, and attaches a sequence number to the output file. See "Recycling EDRs in Pipeline-Only Systems".
FCT_Reject	The FCT_Reject module analyzes the errors in an EDR and, if necessary, moves the EDR to a reject file. See "About Standard Recycling".
FCT_Rounding	Performs rounding for rating and discounting. See "About Resource Rounding" in *BRM Setting Up Pricing and Rating*.
FCT_RSC_Map	Performs rate service class (RSC) mapping. See "About Rate-Service Class Mapping".
FCT_SegRateNoCust	Assigns a segment to an EDR based on the source network instead of customer information. See "About Multi-Segment Rating".
FCT_SegZoneNoCust	Finds the segment using the source network information instead of using the customer information. See "Setting Up Multi-Segment Zoning" in *BRM Setting Up Pricing and Rating*.
FCT_ServiceCodeMap	Maps external service codes to internal service codes. See "Mapping Service Codes and Service Classes" in *BRM Setting Up Pricing and Rating*.
FCT_SocialNo	Flags social numbers for special processing. See "Setting Up Social Numbers".

Table 35–1 (Cont.) Pipeline Manager Modules

Module	Description
FCT_Suspense	When used as part of BRM standard recycling, routes failed EDRs to appropriate output streams depending on their processing status (normal, recycling, or test recycling) and suspense status (succeeded or suspended). When used with Brand Manager, also determines the brand for each suspended call. When used with Suspense Manager, also adds the suspense reason and subreason codes to EDRs.
FCT_TriggerBill	Sends EDRs to the billing-trigger output stream to trigger immediate billing for the associated accounts. It also sets a billing-trigger error code used to route the EDRs to the suspense output stream, and the **Trigger_Billing** recycle key used to retrieve the suspended EDRs for recycling. See "Setting Up Pipeline-Triggered Billing" in *BRM Configuring and Running Billing*.
FCT_UoM_Map	Converts the unit of measurement (UoM) of an incoming EDR to a UoM needed for rating a particular service. See "Converting Units of Measurement" in *BRM Setting Up Pricing and Rating*.
FCT_UsageClassMap	The FCT_UsageClassMap module maps external codes for secondary services, such as call forwarding, to internal usage classes. See "Mapping Usage Classes" in *BRM Setting Up Pricing and Rating*.
FCT_USC_Map	The FCT_USC_Map module performs usage scenario mapping. See "Setting Up Usage Scenario Mapping" in *BRM Setting Up Pricing and Rating*.
FCT_Zone	The FCT_Zone module computes zones when you use Pipeline Manager only for zoning. See "About Setting Up Zones" in *BRM Setting Up Pricing and Rating*.
INP_GenericStream	Provides the input interface to the pipeline. See "Configuring EDR Input Processing".
INP_Realtime	Converts data in an flist to the EDR container format. See "Configuring a Real-Time Discounting Pipeline".
INP_Recycle	Used by standard recycling and Suspense Manager in the pre-recycling pipeline. It reads suspended usage records from the BRM database, restores original EDRs, applies edits to them, and pushes EDRs into the pre-recycling pipeline.
IRL_EventTypeSplitting	Sends EDRs to separate output streams based on service codes. See "Sending EDRs to Pipeline Output Streams".
IRL_LeastCostPerEDR	Flags all EDRs that satisfy the criteria for least cost rating. see "About Least Cost Rating".
IRL_PipelineSplitting	Used in the pre-recycling pipeline to send EDRs to different output streams depending on their original pipeline names. The EDRs are then routed to their original pipelines for recycling.
IRL_LeastCostPerEDR	Flags all EDRs that satisfy the criteria for a promotional savings calculation. See "About Least Cost Rating".

Table 35–1 (Cont.) Pipeline Manager Modules

Module	Description
IRL_UsageType	Assigns usage types to EDRs. See "Mapping Usage Types" in *BRM Setting Up Pricing and Rating*.
ISC_AddCBD	Prepares EDRs for rerating in the back-out pipeline. **Important:** This is a deprecated module but remains in BRM for backward compatibility. See "About Rerating Pipeline-Rated Events" in *BRM Setting Up Pricing and Rating*.
ISC_BACKOUTTypeSplitting	Used by the backout pipeline for back-out-only rerating. It determines if the EDRs are flagged for back-out-only rerating and sends the EDRs to different output streams based on the event types. See "About Configuring the Backout Pipeline for Back-Out-Only Rerating" in *BRM Setting Up Pricing and Rating*.
ISC_CiberInputValidation	Performs record-level validations of CIBER records. See "About Validating Roaming Usage Data" in *BRM Configuring Roaming in Pipeline Manager*.
ISC_CiberOutputMapping	Adds charge data to the ASSOCIATED_CIBER_EXTENSION block of the EDR. If the EDR does not contain an ASSOCIATED_CIBER_EXTENSION block, this iScript adds one. See "About Settling Roaming Charges" in *BRM Configuring Roaming in Pipeline Manager*.
ISC_CiberRejectReason	Sets a reason code in the CIBER extension block for records that are rejected.
ISC_EDRToTAPOUTMap	Populates standard values to fields in output TAP file based on its corresponding value in the EDR container.
ISC_GetCamelFlag	Retrieves the CAMEL flag information for a roaming partner. This iScript is used by roaming outcollect processing.
ISC_LeastCost	Performs one of the following: ■ Calculates and finds the lowest charge for an EDR. ■ Calculates the total savings when using an overlay promotion. See "About Least Cost Rating" and "About Calculating the Promotional Savings".
ISC_MapNetworkOperatorInfo	Maps the DETAIL.SOURCE_NETWORK field to the PIN_FLD_ORIGIN_NETWORK field and the DETAIL.DESTINATION_NETWORK field to the PIN_FLD_DESTINATION_NETWORK field of the opcode input block for the corresponding event. See "Setting Up Number Portability".
ISC_NRTRDE_ErrorReport	Collects the validation errors in the EDRs and creates error records in the Pipeline Manager database. This iScript is used during roaming incollect processing by the NRTRDE (Near Real-Time Roaming Data Exchange) processing pipeline. See the description for detecting roaming fraud using NRTRDE in *BRM Configuring Roaming in Pipeline Manager*.
ISC_NRTRDE_EventSplit	Duplicates and routes EDRs to the corresponding roaming partner NRTRDE output streams based on the roaming partner's NRTRDE flag. This iScript is used by roaming outcollect processing. See the description for detecting roaming fraud using NRTRDE in *BRM Configuring Roaming in Pipeline Manager*.

Table 35–1 (Cont.) Pipeline Manager Modules

Module	Description
ISC_NrtrdeHeaderValidation_v2_01	Validates the information in the header record of the TD35 file based on the TD35 specifications. This iScript is used during roaming incollect processing by the NRTRDE processing pipeline. See the description for detecting roaming fraud using NRTRDE in *BRM Configuring Roaming in Pipeline Manager*.
ISC_ObjectCacheTypeOutputSplitter	Creates two output CDRs from a single input EDR.
ISC_OverrideRateTag	Populates the RATE_TAG field with the value of the NRTRDE flag in the balance impact. This iScript is used by the outcollect settlement pipelines.
ISC_ProfileAnalyzer	Analyzes friends and family extended rating attributes (ERAs) during pipeline rating. See "Pipeline Rating for Friends and Family ERAs" in *BRM Setting Up Pricing and Rating*.
ISC_ProfileLabel	Analyzes ERAs during pipeline rating to determine whether the ERA profiles specified in the **ProfileName** registry entry match the EDR field value.
ISC_PostRating	Adds all the retail and wholesale charges and puts them in DETAIL.RETAIL_CHARGED_AMOUNT_VALUE and DETAIL.WHOLESALE_CHARGED_AMOUNT_VALUE fields. See "Billing Consolidation with CIBER Roaming and Revenue Assurance".
ISC_SetAndValidateBatchInfo	Populates and validates the batch related fields for the EDR container.
ISC_SetEDRStatus	Sets the EDR status to **Success**, **Suspense**, **Duplicate**, **Discard**, or **Skipped** for each EDR.
ISC_SetOutputStream	Sets the Output Stream to **TelOut**, **SMSOut**, **GPRSOut**, **RejectOut**, or **DuplicateOut** for each EDR.
ISC_SetRevenueFigures	Collects the previous and current charged and discount amount for a configured Resource ID.
ISC_SetRevenueStream	Sets the Revenue Stream to **Retail**, **Wholesale**, **Roaming**, or **Unknown** for each EDR.
ISC_SetSvcCodeRTZoning	Finds the service type and updates the **DETAIL.INTERN_SERVICE_CODE** EDR field with the customized service code value for each EDR.
ISC_TapDetailValidation_v3_12	Validates that the fields present in the detail record of the EDR container contain valid data.
ISC_TapSplitting	Splits mobile originating and terminating EDRs when the CDR contains more than one basic service. ISC_TapSplitting creates a new EDR for each additional basic service. See "Generating Multiple TAP MOC and MTC Records".
ISC_TaxCalc	Applies a flat tax to pipeline-rated events. See "About Pipeline Taxation" in *BRM Calculating Taxes*.
LOG	Logs error messages. See "About Pipeline Manager Transactions" in *BRM System Administrator's Guide*.

Table 35–1 (Cont.) Pipeline Manager Modules

Module	Description
Memory Monitor	Monitors Pipeline Manager system memory during startup and while it is processing files. See "Monitoring Pipeline Manager Memory Usage" in *BRM System Administrator's Guide*.
NET_EM	The NET_EM module hosts a BRM External Module (EM). This allows the NET_EM module to use the BRM API opcodes to transfer data between real-time rating and Pipeline Manager. See "Configuring a Real-Time Discounting Pipeline".
OUT_DB	Sends output to the database. See "Sending Output to a Database".
OUT_DevNull	Removes EDRs that are not needed by Pipeline Manager. See "Configuring Output of Discarded EDRs". All registry entries and error messages are handled by the Output Collection module. See Output Collection. For more information, see "Discarding and Skipping EDRs".
OUT_GenericStream	Handles the output stream for rated EDRs. See "Configuring EDR Output Processing". When you configure the OUT_GenericStream module, you configure the EXT_OutFileManager module to specify file management options. See EXT_OutFileManager.
OUT_Realtime	The OUT_Realtime module converts data in the pipeline EDR output to flist format. See "Configuring a Real-Time Discounting Pipeline".
OUT_Reject	Writes rejected EDRs to an output stream. The written record is exactly the same as the original input record. See "Configuring Output for Rejected or Duplicate EDRs". All registry entries and error messages are handled by the Output Collection module. See Output Collection. For more information, see the following documents: ■ About Standard Recycling ■ Handling Duplicate EDRs
Sequencer	Checks for duplicate CDR input files and adds tracking numbers to output streams. See "Configuring Sequence Checking" in *BRM System Administrator's Guide*.
Input Controller	Manages incoming input streams for its associated pipeline. See "Configuring EDR Input Processing".
Output Controller	Manages the output streams for its associated pipeline. See "Configuring EDR Output Processing".
Output Collection	Handles output streams. See "Configuring EDR Output Processing".
Pipeline Controller	Manages all processes in its associated pipeline. See Pipeline Controller.

Table 35–1 (Cont.) Pipeline Manager Modules

Module	Description
Transaction Manager	Coordinates the state of all transactional modules and components in a pipeline. See "About Pipeline Manager Transactions" in *BRM System Administrator's Guide*.
Transaction ID Controller	Generates transaction IDs for all pipelines. See "Configuring the Transaction ID Controller" in *BRM System Administrator's Guide*.
Transaction ID Database Generator	Stores transaction IDs in a Pipeline Manager database table. See "Configuring the Transaction ID Controller" in *BRM System Administrator's Guide*.
Transaction ID File Generator	Stores transaction IDs in a file. See "Configuring the Transaction ID Controller" in *BRM System Administrator's Guide*.

36

Pipeline Manager Function Modules

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager function modules.

FCT_Account

The FCT_Account module adds customer data to an EDR. See "Adding Customer Balance Impact Data to EDRs" in *BRM Setting Up Pricing and Rating*.

It also does the following:

- Flags incoming CDRs to be suspended when the account is being rerated by **pin_rerate**. See "About comprehensive rerating using pin_rerate" in *BRM Setting Up Pricing and Rating*.
- For exclusion rules for usage discounts, retrieves plan list information from the DAT_AccountBatch module and adds this information to the PLAN_LIST block in the CustomerData block in the EDR. The plan list includes all plans for the subscription service, including plans owned by any member services in the subscription group. See "About Exclusion Rules for Usage Discounts".
- For Balance Monitoring, retrieves the balance monitor information for an event owner from the DAT_AccountBatch module and fills the CustomerData block in the EDR with the monitor list. See "About Balance Monitoring and Pipeline Rating" in *BRM Managing Accounts Receivable*.

Dependencies

Requires a connection to the DAT_AccountBatch module.

This module must run before the zoning and rating modules.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–1lists the FCT_Account registry entries.

Table 36–1 FCT_Account Registry Entries

Entry	Description	Mandatory
Active	Specifies whether the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataModule	Specifies the connection to the DAT_AccountBatch module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes
DisableRatingProductCheck	Specifies whether the module rejects any CDRs with no rating products. **True** = FCT_Account does not reject CDRs, if the configured event type for batch rating is not found in any of the products owned by the service or account. **False** = FCT_Account rejects CDRs if the configured event type for batch rating is not found in any of the products owned by the service or account.	No

Sample Registry

```
Account
{
  ModuleName = FCT_Account
  Module
  {
    Active         = True
    DataModule     = ifw.DataPool.CustomerData
    Offset         = 5
  }
}
```

Semaphore File Entries

Table 36–2 lists the FCT_Account Semaphore file entry.

Table 36–2 FCT_Account Semaphore File Entries

Entry	Description
Active	Specifies whether the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.CustomerSearch.Module.Active = False
```

EDR Container Fields

The FCT_Account module uses the EDR container fields listed in Table 36–3:

Table 36–3 FCT_Account EDR Container Fields

Alias field name Default field name	Type	Access	Description
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code that determines the EDR container fields that are used for the customer lookup. For example, a telephone service uses the A number to find the customer account.
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the start timestamp of the event. The time zone information for this timestamp is stored in the field BDR_UTC_TIME_OFFSET.
BDR_UTC_TIME_OFFSET DETAIL.UTC_TIME_OFFSET	String	Read	Contains the UTC time offset that normalizes the charging start timestamp to the UTC time zone. All validity timestamps in the BRM customer data are stored in normalized UTC time. The format is +/- *HHMM*.
DETAIL.CUST_A.ACTG_LAST_DATE	Date	Read	Contains the date that the current monthly cycle began.
DETAIL.CUST_A.ACTG_NEXT_DATE	Date	Read	Contains the date that the current monthly cycle ends.
DETAIL.CUST_A.ACTG_USED_DATE	Date	Write	Contains the date used for this EDR.
RESOURCE_LIST DETAIL.CUST_A.RESOURCE_LIST	String	Write	Contains a list of the resources included in the A-number customer's balance.
RESOURCE_LIST DETAIL.CUST_B.RESOURCE_LIST	String	Write	Contains a list of the resources included in the B-number customer's balance.
DETAIL.CUST_A.INTERN_RATING_PRODUCTS	String	Create	Contains the product rating indexes. This is a comma-separated list of all rating products' indexes associated with the same service and event, and their priorities.
DETAIL.CUST_A.PRODUCT_PRIORITY	Integer	Read	Contains the priority for a product.
DETAIL.CUST_A.PRODUCT.USAGE_START	Date	Read	Contains the start time for a product.
DETAIL.CUST_A.PRODUCT.FIRST_USAGE_INDICATOR	Integer	Write	Specifies whether the account's product is configured to start when first used and the state of the validity period.

Table 36–3 (Cont.) FCT_Account EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.CUST_A.DL.PD.FIRST_USAGE_INDICATOR	Integer	Write	Specifies whether the account's discount is configured to start when first used and the state of the validity period.
BALANCE_GROUP_ID DETAIL.CUST_A.ML.BALANCE_GROUP_ID	String	Create	Contains the balance monitor group ID.
MONITOR_OWNER_ACCT_ID DETAIL.CUST_A.ML.MONITOR_OWNER_ACCT_ID	String	Create	Contains the monitor owner's account ID.
MONITOR_OWNER_ID DETAIL.CUST_A.ML.MONITOR_OWNER_ID	String	Create	Contains the monitor owner ID.
MONITOR_OWNER_TYPE DETAIL.CUST_A.ML.MONITOR_OWNER_TYPE	String	Create	Contains the monitor owner type.
DETAIL.CUST_A.SHARED_PROFILE_LIST. ERA.LABEL	String	Write	Contains the label associated with a shared service profile. For example, MYFAMILY.
DETAIL.CUST_A.PRODUCT.ERA. LABEL	String	Write	Contains the label associated with an owned service profile. For example. MYFAMILY.
DETAIL.CUST_A. SHARED_PROFILE_LIST	Block	Write	Contains all the shared profiles, which the service shares as a member of one or more profile sharing groups.
DETAIL.CUST_A. SHARED_PROFILE_LIST.ERA	Block	Write	Contains shared ERA information.

Database Interface for the FCT_Account Module

The FCT_Account modules uses the following database tables:

- The FCT_Account and FCT_AccountRouter modules use the data in the IFW_ALIAS_MAP table to link an internal service code the EDR container field used for identifying an account. See "Mapping Events and Services" in *BRM Setting Up Pricing and Rating*.
- The FCT_Account and FCT_AccountRouter module use the data in the IFW_EDRC_DESC table to look up the alias mapping data. This table contains all valid EDR container fields for different format descriptions.
- The FCT_Account module uses the data in the IFW_EDRC_FIELD table to look up the alias mapping data. This table contains all valid EDR container fields for different format descriptions.
- The FCT_Account and FCT_AccountRouter use the data in the IFW_PIPELINE table to look up the alias mapping data. The IFW_PIPELINE database table defines the EDR formats that can be used for each pipeline.

For information about the fields in database tables, see the documentation in *Pipeline_Home*/**database**. *Pipeline_Home* is the directory where you installed Pipeline Manager.

FCT_AccountLPRouter

The FCT_AccountLPRouter module looks up the logical partition in which the account resides and writes it to the LOGICAL_PARTITION_ID EDR field.

Dependencies

Requires a connection to the DAT_AccountBatch module.

Registry Entries

Table 36–4 lists the FCT_AccountPRouter registry entries.

Table 36–4 FCT_AccountLPRouter Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataModule	Specifies the connection to the DAT_AccountBatch module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes

Sample Registry

```
#------------------------------------------------
# AccountLPRouter
#------------------------------------------------
AccountLPRouterModule
{
  ModuleName = FCT_AccountLPRouter
  Module
  {
    Active       = True
    DataModule   = ifw.DataPool.CustomerData
  } # end Module
} # end LPRouter
#------------------------------------------------
```

Semaphore File Entries

Table 36–5 lists the FCT_AccountPRouter Semaphore file entry.

Table 36–5 FCT_AccountPRouter Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.AccountLPRouter.Module.Active = False
```

EDR Container Fields

The FCT_AccountLPRouter module uses the following EDR container fields:

Events

Table 36–6 lists the events for FCT_AccountPRouter.

Table 36–6 FCT_AccountPRouter Events

Alias Field Name Default Field Name	Type	Access	Description
LOGICAL_PARTITION_ID	String	Read	Contains the logical partition ID for the event.

FCT_AccountRouter

The FCT_AccountRouter module routes EDRs to the appropriate database schema in multischema systems. This module finds the customer's schema and routes the EDRs to the appropriate pipeline. See "Using Pipeline Manager with Multiple Database Schemas".

When used with standard recycling or Suspense Manager, this module routes call records from the pre-recycling pipeline to the appropriate rating pipeline.

Important: FCT_AccountRouter runs in its own instance of Pipeline Manager and should be configured with its own registry file. Create a registry file that includes entries for the Input, FCT_AccountRouter, and Output modules.

Dependencies

Requires a connection to the DAT_AccountBatch module.

For general use, this module must run after the FCT_ServiceCodeMap module and before the rating modules.

For use with standard recycling or Suspense Manager using multiple database schemas, this module must be run before OUT_GenericStream in a pre-recycling pipeline.

This module sends output to a separate pipeline for each BRM database schema.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–7 lists the FCT_AccountRouter registry entries.

Table 36–7 FCT_AccountRouter Registry Entries

Entry	Description	Mandatory
Active	Specifies whether the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataModule	Specifies the connection to the DAT_AccountBatch module. See "Connecting a Pipeline Manager Module to Another Module" in *BRM System Administrator's Guide*.	Yes
Mode	Specifies how this module routes EDRs to the appropriate pipeline. **ROUTER** = Routes EDRs based on the database schema ID. **RECYCLE** = Routes EDRs based on the pipeline name and schema ID. See "Configuring a Pre-Recycling Pipeline".	Yes
Streams	Lists the target output streams. The syntax for this section depends on whether the module is operating in **ROUTER** mode or **RECYCLE** mode. See "Configuring EDR Output Processing".	Yes

Sample Registries

Sample registry entries for the **ROUTER** mode:

```
AccountRouter
{
   ModuleName = FCT_AccountRouter
   Module
   {
      Active = True
      DataModule = ifw.DataPool.Account
      Mode=ROUTER
      Streams
      {
         1 = DB0001Stream
         2 = DB0002Stream
         3 = DB0003Stream
      }
   }
}
```

Sample registry entries for the **RECYCLE** mode:

```
AccountRouter
{
    ModuleName = FCT_AccountRouter
    Module
    {
        Active = True
        DataModule = ifw.DataPool.CustomerData
        Mode = RECYCLE

        Streams
        {
        # This section maps pipelines to their original stream
            1_Pipeline_A = StreamA_for_DB1
            1_Pipeline_B = StreamB_for_DB1
            2_Pipeline_C = StreamC_for_DB2
```

```
            2_Pipeline_D = StreamD_for_DB2

        # This section maps pipelines to their alternate stream
            1_Pipeline_C = StreamA_for_DB1
            1_Pipeline_D = StreamB_for_DB1
            2_Pipeline_A = StreamC_for_DB2
            2_Pipeline_B = StreamD_for_DB2

        # This section sends EDRs to the suspense stream
            0_* = SuspenseStream
        }
    }
}
```

Semaphore File Entries

Table 36–8 list the FCT_AccountRouter Semaphore file entry.

Table 36–8 FCT_AccountRouter Semaphore File Entry

Entry	Description
Active	Specifies whether the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.Rating.Module.AccountRouter.Module.Active = False
```

EDR Container Fields

The FCT_AccountRouter module uses the EDR container fields listed in Table 36–9:

Table 36–9 FCT_AccountRouter EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code that determines the EDR container fields used for the customer lookup.
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the start timestamp of the event. The time zone information for this timestamp is stored in the BDR_UTC_TIME_OFFSET field.
BDR_UTC_TIME_OFFSET DETAIL.UTC_TIME_OFFSET	String	Read	Contains the UTC time offset that normalizes the charging start timestamp to the UTC time zone. All validity timestamps in the BRM customer data are stored in normalized UTC time. The format is +/- *HHMM*.

Database Tables

The FCT_Account module uses the following database tables:

- IFW_ALIAS_MAP. The FCT_Account and FCT_AccountRouter modules use the data in the IFW_ALIAS_MAP database table to link an internal service code the

EDR container field used for identifying an account. See "Specifying Which Data Is Used for Identifying Accounts" in *BRM Setting Up Pricing and Rating*.

You enter data into this table by using Pricing Center.

- IFW_EDRC_DESC. The FCT_Account and FCT_AccountRouter module use the data in the IFW_EDRC_DESC table to look up the alias mapping data. This table contains all valid EDR container fields for different format descriptions. See "Using Pipeline Manager with Multiple Database Schemas".
- IFW_PIPELINE. The FCT_Account and FCT_AccountRouter use the data in this table to look up the alias mapping data. The IFW_PIPELINE database table defines the EDR formats that can be used for each pipeline. See "Using Pipeline Manager with Multiple Database Schemas".

For information about the fields in database tables, see the documentation in *Pipeline_Home***database**.

Note: For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

FCT_AggreGate

The FCT_AggreGate module performs aggregation of data in EDR containers. See "Setting Up Pipeline Aggregation".

Dependencies

The FCT_AggreGate function module requires a connection to the DAT_ScenarioReader module.

This module runs after rating modules and can run in its own pipeline.

When you configure the FCT_CallAssembling function module to not drop EDRs from the pipeline, ensure that the FCT_AggreGate function module that counts them runs before the FCT_Reject function module. See "FCT_CallAssembling".

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–10 lists the FCT_AggreGate registry entries.

Table 36–10 FCT_AggreGate Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
BackOut	Specifies if the module is working in back-out mode, which is used for rerating. See "Configuring rerating in Pipeline Manager" in *BRM Setting Up Pricing and Rating*. Possible values are **True** and **False**. Default = **False**	No
ControlFile	Specifies the definitions for the control file. The control files are used by the Database Loader utility to load the results into the database.	No
ControlFile.DataFilePath	Specifies whether the path to the data file is included in the control file. Possible values are **True** and **False**. Default = **True**	No
ControlFile.Suffix	Specifies the file name suffix for the control file. You can specify any suffix. Default = **.ctl**	No
IncludeCtlFile	Specifies whether to create control files. If **True**, control and data files are created. If **False**, only data files are created. Default = **True**	No
IncludeErrorEDRs	Specifies whether EDRs that include errors are included in the aggregation scenario. Possible values are **True** and **False**. Default = **False**	No
IncludeInvalidDetailEDRs	Specifies whether EDRs that are invalid are included in the aggregation scenario. Possible values are **True** and **False**. Default = **False**	No
ResultFile	Sub-entries define characteristics of the result file.	No
ResultFile.DoneSuffix	Specifies the file name suffix for processed files. You can specify any suffix. Default = **.dat**	No
ResultFile.TempSuffix	Specifies the file name suffix for temporary files created during processing. You can specify any suffix. Default = **.tmp**	No
ResultFile.WriteEmptyFile	Indicates whether to create an empty processed file if there are no processing results. Possible values are **True** and **False**. Default = **True**	No
ScenarioReaderDataModule	Specifies a connection to the DAT_ScenarioReader module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes

Table 36–10 (Cont.) FCT_AggreGate Registry Entries

Entry	Description	Mandatory
Scenarios	Specifies the scenario that is processed and configured. See "Specifying Scenario Attributes". If nothing is entered, no processing occurs.	No
Scenarios	No value. Subentries define the scenarios to be processed and how they are configured. See "Specifying Scenario Attributes". If nothing is entered, no processing occurs.	No
Scenarios.*name*	Specifies the name of a scenario to be configured. Names and codes are set when scenarios are defined in Pricing Center. The scenarios included with Revenue Assurance Manager have names such as **RA_01**, **RA_02** and so on. See "Preconfigured Aggregation Scenario Details" in *BRM Collecting Revenue Assurance Data*.	Yes
Scenarios.*name***.ControlPointId**	Defines the Revenue Assurance control point that uses this scenario. Control point names must be unique system-wide. If a value is specified, the control point ID is included in the result file.	No
Scenarios.*name***.CtlDir**	Specifies the directory from which to read the control file. Use this entry when you want to override the default directory defined for the scenario in Pricing Center.	No
Scenarios.*name***.DoneDir**	Specifies a path and name for processed files. Use this entry when you want to override the default directory defined for the scenario in Pricing Center.	No
Scenarios.*name***.FieldDelimiter**	Specifies the delimiter of result fields. The default is a semicolon (;), which is the value defined in the IFW_SCENARIO table.	Yes
Scenarios.*name***.IncludeErrorEDRs**	Speechifies whether EDRs that have errors are included in the aggregation processing. Possible values are **True** and **False**. Default value is **False**. **Note:** For Revenue Assurance, this parameter must be present and set to **True**.	No
Scenarios.*name***.IncludeInvalidDetailEDRs**	Specifies whether invalid EDRs are included in aggregation processing. Possible values are **True** and **False**. Default value is **False**. **Note:** For Revenue Assurance, this parameter must be present and set to **True**.	No
Scenarios.*name***.IncludeProcessingTimestamps**	Specifies whether transaction time range data is included in result files. Possible values are **True** and **False**. Default value is **False**.	No

Table 36–10 (Cont.) FCT_AggreGate Registry Entries

Entry	Description	Mandatory
Scenarios.*name***.TableName**	Specifies the table used when DB Loader stores aggregation results in the database. The table name is also used for files created by this scenario. Use this entry when you want to override the default table name defined for the scenario in Pricing Center.	No
Scenarios.*name***.TempDir**	Specifies a path and name for a directory to use for temporary files created during processing. Use this entry when you want to override the default directory defined for the scenario in Pricing Center.	No
Scenarios.*name***.Threshold**	Specifies the maximum number of aggregations stored in memory before writing to disk. In most cases, there aren't enough aggregations to make the threshold meaningful. A typical value is a relatively large number, such as **99999**. Use this entry when you want to override the default directory defined for the scenario in Pricing Center.	No

Sample Registry

```
Aggregate
{
  ModuleName = FCT_AggreGate
  Module
  {
    Active = TRUE

    ScenarioReaderDataModule = ifw.DataPool.ScenarioReader

    Scenarios
      {
      PURCHASE
      {
        TableName = PURCHASE_RESULT
        Threshold = 100000
        TempDir = result/temp
        DoneDir = result/done
        CtlDir = result/ctl
        FieldDelimiter = |
      }
      STAT
      {
        TempDir = result/temp
        DoneDir = result/done
        CtlDir = result/ctl
        }
        DAY
        {
      }
   }

    ResultFile
    {
      TempSuffix = .tmp
      DoneSuffix = .dat
      WriteEmptyFile = FALSE
```

```
    }

    ControlFile
      {
      IncludeCtlFile = FALSE
      Suffix = .ctl
      DataFilePath = TRUE
    }
  }
}
```

Semaphore File Entries

Table 36–11 lists the FCT_AggreGate Semaphore file entry.

Table 36–11 FCT_AggreGate Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.Aggregate.Module.Active = False
```

EDR Container Fields

The module reads EDR container fields defined by the DAT_ScenarioReader module. See "DAT_ScenarioReader".

Events

Table 36–12 lists the FCT_AggreGate events.

Table 36–12 FCT_AggreGate Events

Event name	Trigger	Sender	Parameter
EVT_CTL_FILE_CREATED	Control file for the DB Loader utility was created.	DAT_ScenarioReader	File name

FCT_APN_Map

Before zoning: The FCT_APN_Map module maps the access point name (APN) to a physical PDP address.

After zoning: The FCT_APN_Map module enhances zone values to support enhanced zoning functionality.

For more information, see "Setting up APN Mapping" in *BRM Setting Up Pricing and Rating*.

Dependencies

Requires a connection to the Pipeline Manager database.

This module can be run before or after the zoning modules (FCT_Zone and FCT_PreRating).

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–13 lists the FCT_APN_Map registry entries.

Table 36–13 FCT_APN_Map Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
APNGroup	Specifies the Access Point Name (APN) group value for the mapping. If you enter a group name, run the module before zoning. Otherwise, run it after zoning. You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes

Sample Registry

```
APN_Map
{
  ModuleName = FCT_APN_Map
  Module
  {
    Active = True
    APNGroup = apn_group
    DataConnection = integrate.DataPool.Login
  }
}
```

Semaphore File Entries

Table 36–14 lists the FCT_APN_Map Semaphore file entries.

Table 36–14 FCT_APN_Map Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
APNGroup	If you enter a group name, run the module before zoning. Otherwise, run it after zoning.
Reload	Reloads data from the database into memory. See "Reloading data into a Pipeline Manager module" in *BRM System Administrator's Guide*.

Sample Semaphore File Entries

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.APNMap.Module.Active = True
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.APNMap.Module.APNGroup = apn_group
```

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.APNMap.Module.Reload{}
```

EDR Container Fields

The FCT_APN_Map module uses the EDR container fields listed in Table 36–15:

Table 36–15 FCT_APN_Map EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code.
APN_ADDRESS DETAIL.ASS_GPRS_EXT.APN_ADDRESS	String	Read	Contains the access point name address.
ACTION_CODE DETAIL.ASS_GSMW_EXT.SS_PACKET.ACTION_CODE	String	Read	Contains the action code in supplementary service packet.
SS_EVENT DETAIL.ASS_GSMW_EXT.SS_PACKET.SS_EVENT	String	Read	Contains the supplementary service event.
INTERN_B_NUMBER_ZONE DETAIL.INTERN_B_NUMBER_ZONE	String	Write	Contains the destination zone.
BDR_INTERN_APN_GROUP DETAIL.INTERN_APN_GROUP	String	Read	Contains the APN group.
WHOLESALE_IMPACT_CATEGORY DETAIL.WHOLESALE_IMPACT_CATEGORY	String	Read/Write	Contains the wholesale impact category.
RETAIL.IMPACT_CATEGORY DETAIL.RETAIL.IMPACT_CATEGORY	String	Read/Write	Contains the retail impact category.
ASS_CBD_IMPACT_CATEGORY DETAIL.ASS_CBD.CP.IMPACT_CATEGORY	String	Write	Contains the impact category.
ASS_CBD_INTERN_APN_GROUP DETAIL.ASS_CBD.CP.INTERN_APN_GROUP	String	Read	Contains the APN group in the charge packet.
RATEPLAN_TYPE DETAIL.ASS_CBD.CP.RATEPLAN_TYPE	String	Read	Contains the wholesale or retail rate plan type. The default is retail.
ASS_ZBD_INTERN_APN_GROUP DETAIL.ASS_ZBD.ZP.INTERN_APN_GROUP	String	Read	Contains the APN group in zone breakdown records.
ASS_ZBD_ZONE_RESULT_VALUE_WS DETAIL.ASS_ZBD.ZP.ZONE_RESULT_VALUE_WS	String	Write	Contains the wholesale zone result.
ASS_ZBD_ZONE_RESULT_VALUE_RT DETAIL.ASS_ZBD.ZP.ZONE_RESULT_VALUE_RT	String	Write	Contains the retail zone result.

Database Tables

The FCT_APN_Map module uses the following database tables:

- IFW_APN_MAP. This table stores the APN mapping rules. To enter data in this table, use Pricing Center.

- IFW_APN_GROUP. The IFW_APN_GROUP table stores the APN groups used for APN mapping.

For information about the fields in database tables, see the documentation in *Pipeline_Home***database**.

Note: For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

FCT_ApplyBalance

The FCT_ApplyBalance module reads the discount packets added by DAT_Discount, adds the discounting sub-balance impact to the EDR, and updates the in-memory balance.

For more information, see "About Discounts".

When the discount impacts a non-currency resource balance that starts on first usage, this module adds the validity period information to the EDR. See "About settIng the Validity of Resources Impacted by Discounts".

See "About Credit Limit and Threshold Checking during Batch Rating" in *BRM Managing Customers*.

This module is mandatory when you configure batch discounting in Pipeline Manager. Add this module to the pipeline after the FCT_Rounding module.

Dependencies

Requires a connection to the DAT_BalanceBatch module.

This module should be in the same function pool as the FCT_Discount module for performance reasons and must run after that module.

This module also must run after FCT_Rounding.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–16 lists the FCT_ApplyBalance registry entries.

Table 36–16 FCT_ApplyBalance Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive: **True** = Active **False** = Inactive	Yes
BalanceDataModule	Specifies the connection to the DAT_BalanceBatch module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes
DiscountDataModule	Specifies the connection to the DAT_Discount module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes

Table 36–16 (Cont.) FCT_ApplyBalance Registry Entries

Entry	Description	Mandatory
FirstUsageCreateStream	Specifies the output stream for resource balance impacts that whose validity periods start on first usage. See "About settIng the Validity of Resources Impacted by Discounts". Default = **FirstUsageResourceOutput**	No
IgnoreEDROnDeadlock	Specifies what the module should do when it encounters a deadlock. **True** = Ignore the EDRs and continue processing the EDR file. **False** = Roll back already processed EDRs and start reprocessing the same file.	No
NumberOfNotificationLimit	Specifies the maximum number of notification events that can be written into the output XML file. Once the maximum number of notification events is reached, the module creates another XML file.	No
NotificationOutputDirectory	Specifies the directory in which to write the output XML files.	No
NotificationOutputPrefix	Specifies the prefix of the output XML file.	No
PortalConfigDataModule	Specifies the connection to the DAT_PortalConfig module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes

Sample Registry

```
ApplyBalance
{
   ModuleName = FCT_ApplyBalance
   Module
   {
      Active                  = TRUE
      DiscountDataModule      = ifw.DataPool.DiscountModelDataModule
      BalanceDataModule       = ifw.DataPool.BalanceDataModule
      PortalConfigDataModule  = ifw.DataPool.PortalConfigDataModule
      IgnoreEDROnDeadlock     = False

     NumberOfNotificationLimit = ifw.DataPool.PortalConfigDataModule
     NotificationOutputDirectory          = ./data/out/notifybalancebreach
     NotificationOutputPrefix             = balancebreach_
   }
}
```

Semaphore File Entries

Table 36–17 lists the FCT_ApplyBalance Semaphore file entries.

Table 36–17 FCT_ApplyBalance Semaphore File Entries

Entry	Description
ReloadCreditThresholdParam	Reloads the value from the **CreditThresholdChecking** business parameter. See "Reloading Data into a Pipeline Manager Module" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_
RATE.Functions.Processing.FunctionPool.ApplyBalanceModule.Module.ReloadCreditThres
holdParam{}
```

For more information, see "Semaphore File Syntax" in *BRM System Administrator's Guide*.

EDR Container Fields

Table 36–18 lists the FCT_ApplyBalance EDR container fields.

Table 36–18 FCT_ApplyBalance EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
DETAIL.EVENT_TYPE	String	Read	Specifies the BRM event type.
INTERNAL.TRANSACTION_ID	String	Read	Specifies the transaction ID. This is used for queuing.
DETAIL.INTERN_PROCESS_STATUS	Integer	Read	Specifies the process status. If set to 2, a recycle test is in progress, and this container is skipped.
DETAIL.ASS_CBD.DP.OBJECT_ACCOUNT	Integer	Read	Specifies the POID of the discount owner.
DETAIL.ASS_CBD.DP.OFFERING_POID	String	Read	Specifies the POID of the account's purchased discount.
DETAIL.ASS_CBD.DP.RESOURCE_ID	Integer	Read	Specifies the ID of the resource impacted by the discount.
DETAIL.ASS_CBD.DP.GRANTED_AMOUNT	Decimal	Read	Specifies the discount grant amount. This can be currency or non-currency.
DETAIL.ASS_CBD.DP.BALANCE_GROUP_ID	String	Read	Specifies the POID of the account's balance group that is impacted by the discount.
DETAIL.ASS_CBD.DP.VALID_FROM	Date	Read	Specifies the date when the discount becomes valid.
DETAIL.ASS_CBD.DP.VALID_FROM_DETAIL	Integer	Read	Specifies the mode of the discounts' start time (such as first-usage or relative), the relative offset unit (such as minutes, months, or cycles) and the number of offset units.
DETAIL.ASS_CBD.DP.VALID_TO	Date	Read	Specifies the date when the discount is no longer valid.
DETAIL.ASS_CBD.DP.VALID_TO_DETAIL	Integer	Read	Specifies the mode of the discounts' end time (such as relative), the relative offset unit (such as minutes, months, or cycles) and the number of offset units.
DETAIL.ASS_CBD.DP.SUB_BALANCE	Packet	Write	Specifies the sub-balance that is impacted by the discount.

Table 36–18 (Cont.) FCT_ApplyBalance EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
DETAIL.ASS_CBD.DP.SUB_BALANCE.REC_ID	Integer	Write	Specifies the ID of the sub-balance impacted by this discount.
DETAIL.ASS_CBD.DP.SUB_BALANCE.AMOUNT	Decimal	Write	Specifies the amount of the sub-balance impacted by this discount packet.
DETAIL.ASS_CBD.DP.SUB_BALANCE.GRANTOR	String	Write	Specifies the product or discount that granted this resource.
DETAIL.ASS_CBD.DP.SUB_BALANCE.VALID_FROM	Date	Write	Specifies the date when this sub-balance becomes valid.
DETAIL.ASS_CBD.DP.SUB_BALANCE.VALID_TO	Date	Write	Specifies the date when this sub-balance is no longer valid.
DETAIL.ASS_CBD.DP.SUB_BALANCE.VALID_FROM_DETAILS	Integer	Write	Specifies the mode of the sub-balance validity period and the relative offset start time details.
DETAIL.ASS_CBD.DP.SUB_BALANCE.VALID_TO_DETAILS	Integer	Write	Specifies the mode of the sub-balance validity period and the relative offset end time details.
DETAIL.ASS_CBD.DP.SUB_BALANCE.CONTRIBUTOR	String	Write	Specifies the service or account that contributes to the sub-balance amount.
DETAIL.ASS_CBD.UBP	Packet	Write	The update balance packet. This packet contains validity period information for all the account's sub-balances that start on first usage. This packet is added when a sub-balance with a first-usage start time is consumed for the first time.
DETAIL.ASS_CBD.UBP.BALANCE_GROUP_ID	Integer	Write	Specifies the POID of the account's balance group associated with a resource balance that starts on first usage.
DETAIL.ASS_CBD.UBP.RESOURCE_ID	Integer	Write	Specifies the ID of the associated resource.
DETAIL.ASS_CBD.UBP.RECORD_NUMBER	Integer	Write	Specifies the sequence number of the record in the file.
DETAIL.ASS_CBD.UBP.VALID_FROM	Date	Write	Specifies the start time of the resource balance.

Table 36–18 (Cont.) FCT_ApplyBalance EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
DETAIL.ASS_CBD.UBP.VALID_TO	Date	Write	Specifies the end time of the resource balance.
DETAIL.ASS_CBD.UBP.VALID_FROM_DETAIL	Integer	Write	Specifies the resource balance start time details: the mode of the validity period (such as first-usage or relative), the relative offset unit (such as minutes, months, or cycles) and the number of offset units.
DETAIL.ASS_CBD.UBP.VALID_TO_DETAIL	Integer	Write	Specifies the resource balance end time details: the mode of the validity period (such as relative), the relative offset unit (such as minutes, months, or cycles) and the number of offset units.

FCT_BatchSuspense

This module is used by the Suspense Manager service to handle file-level suspense operations. It generates the suspended batch create and update streams to be loaded into the BRM database for suspense batch files.

Important: This module must be placed before all the validation modules/iScripts in a pipeline.

This module also adds suspense reason and suspense subreason codes to batches. See:

- Setting Up Suspended Batch (SB) Loader for Suspense Manager
- About the FCT_BatchSuspense Module
- About Suspense Manager

Dependencies

Requires a connection to the BRM database.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–19 lists the FCT_BatchSuspense registry entries.

Table 36–19 FCT_BatchSuspense Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
ResubmitDataModule	Specifies a connection to the DAT_Resubmit module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes
DataConnection	Specifies the connection to the BRM database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
PipelineCategory	Specifies the pipeline category. Default = **CDRPipeline**	Yes
StorableClass	Specifies the storable class used to store suspended batch records. Default =/suspended_batch/cdr See "Suspending CDR Files".	Yes (for Batch file processing)
SuspenseFile	Specifies the batch suspense file this module generates: ■ **Path** Specifies the path where the data file will be written to. ■ **Prefix** Specifies the prefix for the resulting filename. ■ **Suffix** Specifies the suffix for the resulting filename. ■ **TempDataPrefix** Specifies the prefix for the temporary filename that is used while the file is being built.	Yes

Sample Registry

```
#---------------------------
# Batch Suspense FCT
#---------------------------
BatchSuspense
{
  ModuleName                = FCT_BatchSuspense
  Module
  {
    Active                  = TRUE
    ResubmitDataModule      = ifw.DataPool.ResubmitBatch
    DataConnection          = ifw.DataPool.LoginInfranet
    PipelineCategory        = CDRPipeline
    StorableClass           = /suspended_batch/cdr
    SuspenseFile
    {
       Path = ..
       Prefix = Framework
       Suffix = msg
       TempDataPrefix = rej_
          }
```

```
    }
}
```

EDR Container Fields

Table 36–20 lists the FCT_BatchSuspense EDR container fields.

Table 36–20 FCT_BatchSuspense EDR Container Fields

Alias Field Name Default Field Name	Type	Description
BATCH_ID HEADER.BATCH_ID	String	The unique identifier for the batch file. The batch file's BATCH_ID field is set in the HEADER block when it is received by the pipeline and this field receives its value from that field.
SEQUENCE_NUMBER HEADER.SEQUENCE_NUMBER	String	The suspense sequence number.
SENDER HEADER.SENDER	String	The sender.
TAP_PROCESSING_INFO HEADER.TAP_PROCESSING_INFO	String	Tap specific information (e.g TAP file name for a specific RAP etc). The format of this field is specific to TAP/RAP processing modules and this information created and interpreted by these modules
OVERRIDE_REASONS HEADER.OVERRIDE_REASONS	String	Contains the batch suspense reasons that are overridden from SMC while resubmitting the batch. Mapped from the error code. Used by Suspense Manager.

FCT_BillingRecord

The FCT_BillingRecord module consolidates balance impact data into an associated BRM billing record and one or more balance impact packets. See "About Consolidation for BRM Billing".

Note: The FCT_ItemAssign module handles items assigned for usage events.

Important: Don't use the FCT_BillingRecord module in a CIBER roaming revenue assurance environments. For more information, see "Billing Consolidation with CIBER Roaming and Revenue Assurance".

Dependencies

Requires a connection to the following modules:

- DAT_AccountBatch
- DAT_BalanceBatch
- DAT_Currency
- DAT_ItemAssign

This module must run after the FCT_MainRating and FCT_Discount modules.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–21 lists the FCT_BillingRecord registry entries.

Table 36–21 FCT_BillingRecord Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **TRUE** = Active **FALSE** = Inactive You can use this entry in a semaphore file.	Yes
AccountDataModule	Specifies a connection to the DAT_AccountBatch module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes
BalanceDataModule	Specifies a connection to the DAT_BalanceBatch module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes
ChargeBreakDownRecordType	By default, FCT_BillingRecord module processes Charge Breakdown records whose record type is 981. Use this entry to specify additional Charge Breakdown records. **Note:** You can specify multiple Charge Breakdown record types. Separate each record type by using a comma.	No
CurrencyDataModule	Specifies a connection to the DAT_Currency module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes
CurrencyType	Specifies the CHARGED_CURRENCY_TYPE value that the charge packets should contain. The possible values are: ■ **H** = Home currency ■ **B** = Billing currency ■ **R** = Rating currency Default = **H**. See "Defining Currency Exchange Rates" in *BRM Setting Up Pricing and Rating*	No
ItemAssignDataModule	Specifies a connection to the DAT_ItemAssign module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes

Table 36–21 (Cont.) FCT_BillingRecord Registry Entries

Entry	Description	Mandatory
PortalConfigDataModule	Specifies the connection to the DAT_PortalConfig module. See the discussion about connecting pipeline manager modules to DAT_PortalConfig in *BRM System Administrator's Guide*.	Yes
RatingPipeline	Specifies whether the module is running in the rating or rerating pipeline: **FALSE** = Rerating **TRUE** = Rating	No
RatePlanType	By default, FCT_BillingRecord processes charge packets whose rateplan type is R. Use this entry to specify additional rateplan types.	No

Sample Registry Entry

```
AddInfranetBillingRecord
{
  ModuleName = FCT_BillingRecord
  Module
  {
    Active = TRUE
    AccountDataModule  = ifw.DataPool.CustomerData
    PortalConfigDataModule   = ifw.DataPool.PortalConfigDataModule
    BalanceDataModule = ifw.DataPool.BalanceDataModule
    ChargeBreakDownRecordType = 981
    CurrencyType = R
    CurrencyDataModule = ifw.DataPool.CurrencyDataModule
    ItemAssignDataModule = ifw.DataPool.ItemAssignDataModule
    RatingPipeline = TRUE
    RatePlanType = W
  }
}
```

Semaphore File Entries

Table 36–22 lists the FCT_BillingRecord Sempahore file entries.

Table 36–22 FCT_BillingRecord Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.AddInfranetBillingRecord.
Module.Active = false
```

EDR Container Fields

Table 36–23 lists the FCT_BillingRecord EDR container fields.

Table 36–23 FCT_BillingRecord EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
A_NUMBER DETAIL.A_NUMBER	String	Read	Contains the normalized A number.
ACCOUNT_POID DETAIL.ASS_PIN.ACCOUNT_POID	String	Write	Contains the resulting account POID.
ACCOUNT_POID DETAIL.ASS_PIN.MP.ACCOUNT_POID	String	Create	POID of the account that the balance impact applies to. **Derivation:** Mandatory. Calculated.
ASS_CBD_IMPACT_CATEGORY DETAIL.ASS_CBD.CP.IMPACT_CATEGORY	String	Read	Contains the impact category.
ASS_CBD_RECORD_TYPE DETAIL.ASS_CBD.RECORD_TYPE	String	Read	Contains the record type of the associated charge record.
ASS_CBD_RUM_NAME DETAIL.ASS_CBD.RUM_NAME	String	Read	Contains the RUM name.
ASS_PIN_BALANCE_PACKET DETAIL.ASS_PIN.NUMBER_OF_BALANCE_PACKETS	Integer	Write	Contains the resulting number of balance packets.
B_NUMBER DETAIL.B_NUMBER	String	Read	Contains the normalized B number.
BAL_GRP_ID DETAIL.INTERN_BALANCE_GROUP_ID	String	Read	Contains the balance group POID of the service.
BAL_GRP_POID DETAIL.ASS_PIN.MP.BAL_GRP_POID	String	Create	Balance monitor group that the balance impact applies to. **Derivation:** Mandatory. Calculated.
BALANCE_GROUP_ID DETAIL.CUST_A.ML.BALANCE_GROUP_ID	String	Read	Balance monitor group ID.
BDR_UTC_TIME_OFFSET DETAIL.UTC_TIME_OFFSET	String	Read	Contains the UTC offset.
BP_ACCOUNT_POID DETAIL.ASS_PIN.BP.ACCOUNT_POID	String	Write	Contains account POID.
BP_RUM_ID DETAIL.ASS_PIN.BP.RUM_ID	Long	Read/Write	Contains RUM id of the balance packet.
CHARGED_AMOUNT_CURRENCY DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_CURRENCY	String	Read	Contains the currency.
CHARGED_AMOUNT_VALUE_ORIG DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE_ORIG	Decimal	Read	Contains the charged amount value.

Table 36–23 (Cont.) FCT_BillingRecord EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
CHARGED_AMOUNT_VALUE DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE	Decimal	Read	Contains the charged amount value.
CHARGED_CURRENCY_TYPE DETAIL.ASS_CBD.CP.CHARGED_CURRENCY_TYPE	String	Read	Contains the currency type: ■ **H** = Home ■ **R** = Rating ■ **B** = Billing
CHARGED_TAX_CODE DETAIL.ASS_CBD.CP.CHARGED_TAX_CODE	String	Read	Contains the charged tax code.
CHARGING_END_TIMESTAMP DETAIL.CHARGING_END_TIMESTAMP	Date	Read	Contains charging end time.
CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains charging start time.
CONNECT_SUBTYPE DETAIL.CONNECT_SUBTYPE	String	Read	Contains the connect subtype.
DETAIL.ASS_CBD.CP.ITEM_TAG	String	Read	Contains the name of the charge item for balance impacts.
DETAIL.ASS_CBD.CP.ITEM_POID	String	Read	Contains the POID of the associated charge item for balance impacts.
DETAIL.ASS_CBD.DP.ITEM_TAG	String	Read	Contains the name of the discount item for balance impacts.
DETAIL.ASS_CBD.DP.ITEM_POID	String	Read	Contains the POID of the associated discount item for balance impacts.
DETAIL.ASS_CBD.TP.ITEM_TAG	String	Read	Contains the name of the tax item for balance impacts.
DETAIL.ASS_CBD.TP.ITEM_POID	String	Read	Contains the POID of the associated tax item for balance impacts.
CP_RUM_ID DETAIL.ASS_CBD.CP.RUM_ID	Long	Read	Contains RUM id.
DETAIL.ASS_CBD.CP.RECORD_TYPE	String	Read	Contains charge packet record type.
DETAIL.ASS_CBD.DP	Record	Read	The discount packet record.
DETAIL.ASS_CBD.DP.BALANCE_GROUP_ID	Integer	Read	Contains the ID of the balance group that the discount applies to.
DETAIL.ASS_CBD.DP.GLID	String	Read	Contains the G/L ID.
DETAIL.ASS_CBD.DP.GRANTED_AMOUNT	Decimal	Read	Contains the discount amount granted. This can be currency or non-currency.

Table 36–23 (Cont.) FCT_BillingRecord EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
DETAIL.ASS_CBD.DP.GRANTED_QUANTITY	Decimal	Read	Contains the discount base values used to calculate the amount granted.
DETAIL.ASS_CBD.DP.IMPACT_CATEGORY	String	Read	Contains the discount impact category.
DETAIL.ASS_CBD.DP.NODE_LOCATION	String	Read	Contains the node location of the discount.
DETAIL.ASS_CBD.DP.OBJECT_ACCOUNT	String	Read	Contains the POID of the discount owner.
DETAIL.ASS_CBD.DP.OBJECT_ID	String	Read	Contains the ID of the discount or sponsor object.
DETAIL.ASS_CBD.DP.OBJECT_OWNER_ID	String	Read	Contains the POID type of the discount owner.
DETAIL.ASS_CBD.DP.OBJECT_OWNER_TYPE	String	Read	Contains the POID type of the discount owner.
DETAIL.ASS_CBD.DP.OBJECT_TYPE	String	Read	Contains the discount or sponsor object that generated the discount.
DETAIL.ASS_CBD.DP.RATETAG	String	Read	Contains the rate tag for the discount.
DETAIL.ASS_CBD.DP.RESOURCE_ID	String	Read	Contains the ID of the resource impacted.
DETAIL.ASS_CBD.DP.SUB_BALANCE.AMOUNT	Decimal	Read	Contains the amount applied to this sub-balance.
DETAIL.ASS_CBD.DP.SUB_BALANCE.CONTRIBUTOR	String	Read	Contains the sub-balance contributor field.
DETAIL.ASS_CBD.DP.SUB_BALANCE.REC_ID	Integer	Read	Contains the record ID of the sub-balance record.
DETAIL.ASS_CBD.DP.SUB_BALANCE.VALID_FROM	Date	Read	Contains the date the resource is valid from.
DETAIL.ASS_CBD.DP.SUB_BALANCE.VALID_TO	Date	Read	Contains the date the resource is valid to.
DETAIL.ASS_CBD.DP.TAX_CODE	String	Read	Contains the tax code for the discount.
DETAIL.ASS_PIN.BI.PIN_RATE_TAG	String	Write	Contains the name of the rate plan that provided a promotional savings.
DETAIL.ASS_PIN.BP.BAL_GRP_POID	String	Write	Contains the POID of the balance group that is impacted.
DETAIL.ASS_PIN.BP.ITEM_POID	String	Write	Contains the POID of the associated item.
DETAIL.ASS_PIN.SBI	Record	Write	The sub-balance impact record.
DETAIL.ASS_PIN.SBI.BAL_GRP_POID	String	Write	Contains the POID of the balance group that is impacted.

Table 36–23 (Cont.) FCT_BillingRecord EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
DETAIL.ASS_PIN.SBI.PIN_RESOURCE_ID	Integer	Write	Contains the resource ID.
DETAIL.ASS_PIN.SBI.RECORD_NUMBER	Integer	Write	Contains the record number of the sub-balance impact record.
DETAIL.ASS_PIN.SBI.RECORD_TYPE	String	Write	Contains the record type of the sub-balance impact record.
DETAIL.ASS_PIN.SBI.SB	Record	Write	The sub-balance record.
DETAIL.ASS_PIN.SBI.SB.CONTRIBUTOR	String	Write	Contains the sub-balance contributor field.
DETAIL.ASS_PIN.SBI.SB.PIN_AMOUNT	Decimal	Write	Contains the amount of the sub-balance impact.
DETAIL.ASS_PIN.SBI.SB.RECORD_NUMBER	Integer	Write	Contains the record number of the sub-balance record.
DETAIL.ASS_PIN.SBI.SB.RECORD_TYPE	String	Write	Contains the record type of the sub-balance record.
DETAIL.ASS_PIN.SBI.SB.VALID_FROM	Date	Write	Contains the date the sub-balance resource is valid.
DETAIL.ASS_PIN.SBI.SB.VALID_TO	Date	Write	Contains the date the sub-balance resource is no longer valid.
DETAIL.CUST_A.ACCOUNT_PARENT_ID	String	Read	Contains the BRM account ID.
DETAIL.CUST_A.INTERN_FOUND_PP_INDEX	Long	Read	Contains the index of the purchased product.
DETAIL.CUST_A.PRODUCT.OFFERING_POID	String	Read	Contains the product's node location.
DETAIL.CUST_A.PRODUCT.PRODUCT_ID	String	Read	Contains the BRM product ID.
DETAIL.CUST_A.PRODUCT.SERVICE_ID	String	Read	Contains the ID of the product.
DETAIL.CUST_A.PRODUCT.SERVICE_TYPE	String	Read	Contains the service type of the product.
DETAIL.CUST_A.PRODUCT.SERVICE_USED_ITEM_POID	String	Read	Contains the item POID.
DETAIL_OBJECT_CACHE_TYPE DETAIL.OBJECT_CACHE_TYPE	Long	Write	Contains the cache type of the associated object.
DP_AMOUNT_ORIG DETAIL.ASS_CBD.DP.GRANTED_AMOUNT_ORIG	Decimal	Read	Contains original discount amount
DP_INTERN_RUM_ID DETAIL.ASS_CBD.DP.INTERN_RUM_ID	Long	Read	Contains discount RUM id.
DP_OFFERING_POID DETAIL.ASS_CBD.DP.OFFERING_POID	String	Read	Contains discount offering POID.

Table 36–23 (Cont.) FCT_BillingRecord EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
DP_RESOURCE_ID_ORIG DETAIL.ASS_CBD.DP.RESOURCE_ID_ORIG	Long	Read	Contains original resource id.
DP_SB_GRANTOR DETAIL.ASS_CBD.DP.SUB_BALANCE.GRANTOR	String	Read	Contains discount sub-balance grantor.
DP_SB_VALID_FROM_DETAILS DETAIL.ASS_CBD.DP.SUB_BALANCE.VALID_FROM_DETAILS	Long	Read	Contains discount sub-balance valid from details.
DP_SB_VALID_TO_DETAILS DETAIL.ASS_CBD.DP.SUB_BALANCE.VALID_TO_DETAILS	Long	Read	Contains discount sub-balance valid to details.
EXCHANGE_CURRENCY DETAIL.ASS_CBD.CP.EXCHANGE_CURRENCY	String	Read	Contains currency for exchange.
EXPIRED_UNITS DETAIL.ASS_CBD.CP.DP. EXPIRED_UNITS	String	Write	Contains the number of units that exceed the maximum.
GRANTED_DISCOUNT_AMOUNT_VALUE DETAIL.ASS_CBD.CP.GRANTED_DISCOUNT_AMOUNT_VALUE	Decimal	Read	Contains the granted discount amount value.
GRANTOR DETAIL.ASS_PIN.SBI.SB.GRANTOR	String	Write	Contains grantor of sub-balance impact.
ITEM_POID DETAIL.ASS_PIN.ITEM_POID	String	Write	Contains the resulting item POID.
ITEM_TAG DETAIL.ITEM_TAG	String	Read	Contains the item tag.
MONITOR_OWNER_ACCT_ID DETAIL.CUST_A.ML.MONITOR_OWNER_ACCT_ID	String	Read	Monitor owner's account ID.
MONITOR_OWNER_ID DETAIL.CUST_A.ML.MONITOR_OWNER_ID	String	Read	Monitor owner ID.
MONITOR_OWNER_TYPE DETAIL.CUST_A.ML.MONITOR_OWNER_TYPE	String	Read	Monitor owner type.
MONITOR_SUB_BAL DETAIL.ASS_PIN.MSBI.MONITOR_SUB_BAL	SB	Create	Sub-balance monitored.
MSBI_BAL_GRP_POID DETAIL.ASS_PIN.MSBI.BAL_GRP_POID	String	create	Contains balance group POID of monitor sub-balance impact.
MSBI_MSB_PIN_AMOUNT DETAIL.ASS_PIN.MSBI.MSB.PIN_AMOUNT	Decimal	Write	Contains amount of monitor sub-balance.
MSBI_MSB_RECORD_NUMBER DETAIL.ASS_PIN.MSBI.MSB.RECORD_NUMBER	Integer	Write	Contains record number of monitor sub-balance.
MSBI_MSB_RECORD_TYPE DETAIL.ASS_PIN.MSBI.MSB.RECORD_TYPE	String	Write	Contains record type of monitor sub-balance.

Table 36–23 (Cont.) FCT_BillingRecord EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
MSBI_MSB_VALID_FROM DETAIL.ASS_PIN.MSBI.MSB.VALID_FROM	Date	Write	Contains valid from of monitor sub-balance.
MSBI_MSB_VALID_TO DETAIL.ASS_PIN.MSBI.MSB.VALID_TO	Date	Write	Contains valid to of monitor sub-balance.
MSBI_PIN_RESOURCE_ID DETAIL.ASS_PIN.MSBI.PIN_RESOURCE_ID	Long	Write	Contains resource id of monitor sub-balance impact.
MSBI_RECORD_TYPE DETAIL.ASS_PIN.MSBI.RECORD_TYPE	String	Write	Contains record type of monitor sub-balance impact.
NET_QUANTITY DETAIL.NET_QUANTITY	Decimal	Write	Contains net quantity.
OBJECT_CACHE_TYPE DETAIL.ASS_PIN.BP.OBJECT_CACHE_TYPE"	Integer	Write	Contains the cache type of the associated object. **Possible values:** ■ **2** (POSTPAID) ■ **0** (CONVERGENT).
ORIGINAL_EVENT_POID DETAIL.ASS_PIN.ORIGINAL_EVENT_POID	String	Read	Contains original event POID.
PIN_AMOUNT_DEFERRED DETAIL.ASS_PIN.BP.PIN_AMOUNT_DEFERRED	Integer	Write	Specifies whether an EDR contains tax data. This field is set to **0** when an EDR contains a tax packet and a PIN_AMOUNT when an EDR doesn't contain a tax packet.
PIN_AMOUNT_ORIG DETAIL.ASS_PIN.BP.PIN_AMOUNT_ORIG	Decimal	Read/Write	Contains original amount.
PIN_AMOUNT DETAIL.ASS_PIN.BP.PIN_AMOUNT	Decimal	Write	Contains the resulting amount used to update BRM balances.

Table 36–23 (Cont.) FCT_BillingRecord EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
PIN_AMOUNT DETAIL.ASS_PIN.MP.PIN_AMOUNT	9(11)	Create	Amount of one resource impact for the account balance. The value can be either positive or negative. The value is added to the PIN_FLD_CURRENT_BAL field of the PIN_FLD_BALANCES array in the account object specified by PIN_FLD_ACCOUNT_OBJ. **Note:** In case of multiple-RUM rating, this value might be a total value. Possible values: Price. See below for minimum and maximum values.If no price is given, this is a space, for example, NULL in a database. The format is variable floating point. The floating decimal point must be set if the given value is not in the required format. Maximum value: 99999999999 Minimum value: -9999999999 Examples: ■ '00000000125' for 125,00 ■ '00000012.50' for 12,50 ■ '-0012.56780' for -12,5678 Derivation: Mandatory. Derived from the object, **/event/PIN_FLD_BAL_IMPACTS.PIN_FLD_AMOUNT**. The post-mapping processor decides what mapping rule applies to this attribute; for example, add multiple charge packet values. **Note:** This value does not include any granted discounts.
PIN_DISCOUNT DETAIL.ASS_PIN.BP.PIN_DISCOUNT	Decimal	Write	Contains the resulting discount values.
PIN_GL_ID DETAIL.ASS_PIN.BP.PIN_GL_ID	String	Write	Contains the resulting G/L ID.

Table 36–23 (Cont.) FCT_BillingRecord EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
PIN_IMPACT_CATEGORY DETAIL.ASS_PIN.BP.PIN_IMPACT_CATEGORY	String	Write	Contains the impact category from the charge packet or the discount packet, depending on which was passed.
PIN_IMPACT_TYP DETAIL.ASS_PIN.BP.PIN_IMPACT_TYPE	Long	Write	Contains impact type.
PIN_INFO_STRING DETAIL.ASS_PIN.BP.PIN_INFO_STRING	String	Read/Write	Contains information of balance packet.
PIN_INVOICE_DATA DETAIL.ASS_PIN.PIN_INVOICE_DATA	String	Write	Contains the resulting BRM invoice data.
PIN_OFFERING_POID DETAIL.ASS_PIN.BP.PIN_OFFERING_POID	String	Write	Uniquely identifies an account product, discount, or sponsor.
PIN_PRODUCT_POID DETAIL.ASS_PIN.BP.PIN_PRODUCT_POID	String	Write	Contains the resulting product object.
PIN_QUANTITY DETAIL.ASS_PIN.BP.PIN_QUANTITY	Decimal	Write	Contains the resulting quantity used to update BRM accounts.
PIN_RATE_TAG DETAIL.ASS_PIN.BP.PIN_RATE_TAG	String	Write	Contains the rate tag from the discount packet.
PIN_RESOURCE_ID_ORIG DETAIL.ASS_PIN.BP.PIN_RESOURCE_ID_ORIG	Long	Write	Contains original resource id.
PIN_RESOURCE_ID DETAIL.ASS_PIN.BP.PIN_RESOURCE_ID	String	Write	Contains the resulting ID of the BRM resource.
PIN_RESOURCE_ID DETAIL.ASS_PIN.MP.PIN_RESOURCE_ID	9(9)	Create	Numeric value of the resource that is impacted; for example, 840 for US dollars. Possible value: Any configured BRM resource ID. **Derivation:** Mandatory.
PIN_TAX_CODE DETAIL.ASS_PIN.BP.PIN_TAX_CODE	String	Write	Contains the resulting BRM tax code.
PRICEMODEL_TYPE DETAIL.ASS_CBD.CP.PRICEMODEL_TYPE	String	Read	Contains the price model type.
RATEPLAN_CODE DETAIL.ASS_CBD.CP.RATEPLAN_CODE	String	Read	Contains rateplan code.
RATEPLAN_TYPE DETAIL.ASS_CBD.CP.RATEPLAN_TYPE	String	Read	Contains the rate plan type.

Table 36–23 (Cont.) FCT_BillingRecord EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
RECORD_TYPE DETAIL.ASS_PIN.MP.RECORD_TYPE	String	Create	Extended to be 3 bytes long. Possible values: ■ 800: monitor packet ■ 805: monitor sub-balance impact ■ 807: monitor sub-balance
RESOURCE_ID DETAIL.ASS_CBD.CP.RESOURCE_ID	Long	Read	Contains charge packet resource ID.
RESOURCE_ORIG DETAIL.ASS_CBD.CP.RESOURCE_ID_ORIG	Long	Read	Contains charge packet original resource ID.
RM_NET_QUANTITY DETAIL.ASS_CBD.RM.NET_QUANTITY	Decimal	Read/Write	Contains net quantity of RUM.
ROUNDED_QUANTITY_VALUE DETAIL.ASS_CBD.CP.ROUNDED_QUANTITY_VALUE	Decimal	Read	Contains the rounded quantity value that was used for the price calculations.
RUM_NAME DETAIL.ASS_PIN.RUM_NAME	String	Write	Contains RUM name.
SERVICE_POID DETAIL.ASS_PIN.SERVICE_POID	String	Write	Contains the resulting service POID.
TAX_LOCALES DETAIL.ASS_PIN.PIN_TAX_LOCALES	String	Write	Contains the resulting tax locales string.
TJ_PIN_AMOUNT DETAIL.ASS_PIN.BP.TJ.PIN_AMOUNT	Decimal	create	Contains amount of tax jurisdiction of balance packet.
TJ_PIN_TAX_RATE DETAIL.ASS_PIN.BP.TJ.PIN_TAX_RATE	String	Create	Contains tax rate of tax jurisdiction of balance packet.
TJ_PIN_TAX_TYPE DETAIL.ASS_PIN.BP.TJ.PIN_TAX_TYPE	String	Read/create	Contains tax type of tax jurisdiction of balance packet.
TJ_PIN_TAX_VALUE DETAIL.ASS_PIN.BP.TJ.PIN_TAX_VALUE	Decimal	Create	Contains tax value of tax jurisdiction of balance packet.
TJ_RECORD_TYPE DETAIL.ASS_PIN.BP.TJ.RECORD_TYPE	String	Create	Contains record type of tax jurisdiction of balance packet.
TP_RELATED_CHARGE_INFO_ID DETAIL.ASS_CBD.TP.RELATED_CHARGE_INFO_ID	Long	Read	Contains related charge info id.
TP_TAX_CODE DETAIL.ASS_CBD.TP.TAX_CODE	String	Read	Contains tax code.

Table 36–23 (Cont.) FCT_BillingRecord EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
TP_TAX_RATE DETAIL.ASS_CBD.TP.TAX_RAT	String	Read	Contains tax rate.
TP_TAX_TYPE DETAIL.ASS_CBD.TP.TAX_TYPE	String	Read	Contains tax type.
TP_TAX_VALUE_ORIG DETAIL.ASS_CBD.TP.TAX_VALUE_ORIG	Decimal	Read	Contains original tax value.
TP_TAX_VALUE DETAIL.ASS_CBD.TP.TAX_VALUE	Decimal	Read	Contains tax value.
TP_TAXABLE_AMOUNT DETAIL.ASS_CBD.TP.TAXABLE_AMOUNT	Decimal	Read	Contains taxable amount.
USAGE_GL_ACCOUNT_CODE DETAIL.ASS_CBD.CP.USAGE_GL_ACCOUNT_CODE	String	Read	Contains G/L account code.
VALID_FROM_DETAILS DETAIL.ASS_PIN.SBI.SB.VALID_FROM_DETAILS	Long	Write	Contains valid from details of sub-balance impact.
VALID_FROM DETAIL.ASS_PIN.MSB.VALID_FROM	Date	Create	Valid from date for this sub-balance.
VALID_TO_DETAILS DETAIL.ASS_PIN.SBI.SB.VALID_TO_DETAILS	Long	Write	Contains valid to details of sub-balance impact.
VALID_TO DETAIL.ASS_PIN.MSB.VALID_TO	Date	Create	Valid-to date for this sub-balance.

FCT_CallAssembling

The FCT_CallAssembling module assembles the multiple CDRs that comprise a single wireless call into a single EDR that Pipeline Manager can process. See "Assembling EDRs".

Dependencies

You must run this module early in a pipeline to assemble EDRs. You must run it before FCT_Discard.

When you configure the FCT_CallAssembling function module to not drop EDRs from the pipeline, ensure that the FCT_AggreGate function module that counts them runs before the FCT_Reject function module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–24 lists the FCT_CallAssembling registry entries.

Table 36–24 FCT_CallAssembling Registry Entries

Entry	Description	Mandatory
Active	Turns FCT_CallAssembling module processing on and off. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
AssembledEDR	Specifies a list of fields that the EDR takes from the **L** call segment and appends it to the last EDR (if the two are different). No default entry. See "Capturing Fields From the Last Call Record".	No
AssembleSGSN	Turns SGSN data capture on and off. If **True**, this entry waits for all CDRs to arrive before rating a call. If your system does not process TAP records, leave this set to **False** to save system resources. **True** = SGSN data recorded **False** = SGSN data not recorded Default = **False** See "Rating Calls by Volume of Data Sent".	No
AssembleVolume	Turns volume rating on and off. If **True**, this entry waits for all CDRs to arrive before rating a call. If your system does not require volume rating, leave this set to **False** to save system resources. **True** = volume rating on **False** = volume rating off Default = **False** See "Rating Calls by Volume of Data Sent".	No
CallDurationTolerance	Specifies an allowable cumulative time error for a single call (in seconds). Used with **SplitAtGaps = True**. Default = **60** See "Specifying a Time Error".	No
DropLateCDRs	Specifies how to handle the output of late EDRs: ▪ **True** = Drop late EDRs from the pipeline. ▪ **False** = Send late EDRs through the pipeline as non-valid. Default = **True** See "Dropping Late Calls".	No
EmitPartialEDROnUpgrade	Specifies the results of the **UpgradeFlushLimit** semaphore. Results are one of the following: ▪ Silently drops EDRs from the in-memory **.dat** file. ▪ Emit partial EDRs for revenue assurance tracking. (Partial EDRs should be sent to the discard stream.) Default = **False** (disabled) See "Discarding Incomplete Calls After Changing The EDR Container Description" in *BRM Setting Up Pricing and Rating*.	No
FileName	Specifies the base file name for the data files. The transaction ID and the suffix are appended. See "Managing the Call Assembling Data Files".	Yes

Table 36–24 (Cont.) FCT_CallAssembling Registry Entries

Entry	Description	Mandatory
MaxDuration	Directs FCT_CallAssembling to rate segments of a wireless call periodically. This entry specifies the maximum amount of time (in seconds) that a call can remain open before FCT_CallAssembling rates the segments that have arrived. This module recalculates the call duration for every call each time a new call segment arrives and compares it to the **MaxDuration** setting. If the new time duration equals or exceeds the setting for **MaxDuration,** FCT_CallAssembling emits an EDR to rate the existing portion of the call. For details and a comparison to **FlushLimit**, see "Rating Calls by Time Duration". No default entry.	No
Mode	Change this entry *only* if you are creating data upgrade pipelines that are used when changing an EDR container description. The possible values are: **Normal**. The default mode, and the most common use of this module. Directs FCT_CallAssembling to assemble CDRs into EDRs so Pipeline Manager can process them. **RestoreEDRs**. Directs FCT_CallAssembling to read serialized EDRs in sequence from data files and inserts them into the pipeline. **UpGradeData**. Directs FCT_CallAssembling to update data files based on the EDRs it receives. Default = **Normal** See "Upgrading Incomplete Calls to the New Container Description" in *BRM Setting Up Pricing and Rating*.	Yes
Path	Specifies the directory for the data files. Default = **.** See "Managing the Call Assembling Data Files".	No
RejectMissingChain	Specifies whether to report an error if a chain reference value is missing. Default = **False**	No
SplitAtGaps	Specifies whether a non-contiguous set of CDRs can be collected into a single EDR. For example, assume that CDRs F, I1, I2, and I4 have arrived. If set to **True**, this entry directs FCT_CallAssembling to emit an EDR for F, I1, and I2, and because I3 is missing, a separate EDR for I4. If set to **False**, all CDRs that have arrived are collected into a single EDR. If set to **True**, FCT_CallAssembling will emit multiple EDRs if a CDR is missing when the call. **True** = Active **False** = Inactive Default = **False** See "Rating Calls by Volume of Data Sent".	No

Startup Registry Interdependencies

The Following Section Explains The Relationships Between Certain Startup Registry Entries.

Sample Registry

```
CallAssembling
{
   ModuleName = FCT_CallAssembling
   Module
    {
      Active = True
      Path = .
      FileName = calls
      RejectMissingChain = False
      AssembleVolume = TRUE
      AssembledEDR  {
          1 = Detail.custom_fields_from_last_edr1
          2 = Detail.custom_field_from_last_edr2...
       }
     }
}
```

Semaphore File Entries

Table 36–25 lists the FCT_CallAssembling Semaphore file entries.

Table 36–25 FCT_CallAssembling Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
ExportDataToXml	Exports the call data in the existing data file to an XML file with the name specified by the **FileName** entry in the startup registry file. See "Migrating Call Assembling Data Between Releases and Pipelines".
ExportDataToXML.CallsPerFile	If the number of calls exported is larger than the resources available in the host system, you can divide the call data into multiple files by using this option and specifying the number of calls per file. See "Migrating Call Assembling Data Between Releases and Pipelines".
FlushLimit	Sets the maximum age (in days) an open (incomplete) EDR can have before being flushed from the work files. For example, a setting of **0** flushes all open calls; a setting of **1** flushes all calls that have been open for a day or more; a setting of **2** flushes all calls that have been open for two days or more, and so on. **Note:** The setting of **0** does not flush future-dated EDRs because the value of CHARGING_START_TIMESTAMP is greater than the system date. No default value. See "Rating Incomplete Time Duration Calls".
FlushServiceCode	Used with **FlushLimit**. Specifies a service. When used, only the calls with the service that match the three-letter service code are flushed. Multiple entries are not allowed. No default value. See "Rating Partial Calls by Service".
ImportDataFromXml	Imports the entire contents of the XML file created by the **ExportDataToXml** entry to the **.dat** file in the new format. **Values:** See "Migrating Call Assembling Data Between Releases and Pipelines".

Table 36–25 (Cont.) FCT_CallAssembling Semaphore File Entries

Entry	Description
ImportDataFromXML.FileName	Specifies the XML file from which to import data. See "Migrating Call Assembling Data Between Releases and Pipelines".
KeepCallOpen	Used with **FlushLimit**. Specifies whether to rate additional EDRs for a call that has already been flushed (**True**). Default = **False** See: ■ Rating Calls by Implied Time Duration ■ Rating Continuous Data Calls by Segment ■ Rating Partial Calls by Service
RemoveLimit	Sets a time limit (in days) for removing EDRs in a **Closed** or **Timeout** state from the work files. No default value. See "Removing Incomplete Time Duration Calls".
RemoveRejectedLimit	Sets a time limit (in days) for removing EDRs in a **Closed_Rejected** or **Timeout_Rejected** state from the work files. No default value. See "Removing Incomplete Time Duration Calls".
UpgradeFlushLimit	Flushes partial EDRs that were closed as a result of a change to the EDR container. No default value (no limit). See "Discarding Incomplete Calls after Changing the EDR Container Description" in *BRM Setting Up Pricing and Rating*.

Sample FlushLimit Semaphore Commands

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.CallAssembling.Module.FlushLimit=1

ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.CallAssembling.
Module.FlushServiceCode = tel

ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.CallAssembling.
Module.KeepCallOpen = True
```

Semaphore Entries for a Call-Assembling Report

For information, see "Tracking the Status of Assembled Calls".

Table 36–26 lists the semaphore entries for Call-Assembling Report.

Table 36–26 Semaphore Entries for Call-Assembling Report

Entry	Description	Mandatory
CreateReport	Command to create the report.	Yes
EndTime	Specifies the end date and time for the report. EDRs created before this date and time are reported. The format is *YYYYMMDDhhmmss*. Default = **0** (Current time)	No
ReportPath	Specifies path of the report file. Default = **.**	No
ReportPrefix	Specifies the file name prefix of the report file. Default = **assembly**	No
StartTime	Specifies the start date and time for the report. EDRs created on or after this date and time are reported. The format is *YYYYMMDDhhmmss*.	No

Sample semaphore command for call assembling reports

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.CallAssembling.Module
{
     CreateReport      = True
     StartTime         = 20111206000000
     Module.StartTime  = 20020315000000
     EndTime           = 20111212090000
     ReportPath        = ./
     ReportPrefix      = call_assembly
}
```

EDR Container Fields

Table 36–27 lists the EDR container fields for Call-Assembling Report.

Table 36–27 EDR Container Fields for Call-Assembling Report

Alias Field Name Default Field Name	Type	Access	Description
CHAIN_REFERENCE DETAIL.CHAIN_REFERENCE	String	Read	Contains the chain reference key.
LONG_DURATION_INDICATOR DETAIL.LONG_DURATION_INDICATOR	String	Read/Write	Contains the long duration indicator. Arriving call segments have one of these: ■ **F** = First ■ **I** = Intermediate ■ **L** = Last Assembled call segments are given one of these: ■ **C** = Complete call. ■ **SL** = *Slice* (portion) of a call. ■ **P** = Partially assembled call. ■ **XC** = Late intermediate call segment. ■ **XO** = Late overlap segment. ■ **XP** =Late segment (any) of a call. See "How FCT_CallAssembling Classifies EDRs".
TRANSACTION_ID	Decimal	Read	Contains the transaction ID.
PROCESS_STATUS	Long	Read	Contains the EDR status.
CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read/Write	Contains the charging time stamp.
DURATION DETAIL.DURATION	Decimal	Read/Write	Contains the duration of the assembled EDR.
VOLUME_SENT DETAIL.VOLUME_SENT	Decimal	Read/Write	Contains the volume sent for the assembled EDR.
VOLUME_RECEIVED DETAIL.VOLUME_RECEIVED	Decimal	Read/Write	Contains the volume received for the assembled EDR.
NUMBER_OF_UNITS DETAIL.NUMBER_OF_UNITS	Decimal	Read/Write	Contains the number of units for the assembled EDR.

Table 36–27 (Cont.) EDR Container Fields for Call-Assembling Report

Alias Field Name Default Field Name	Type	Access	Description
RETAIL_CHARGED_AMOUNT_VALUE DETAIL.RETAIL_CHARGED_AMOUNT_VALUE	Decimal	Read/Write	Contains the retail charged amount value for the assembled EDR.
WHOLESALE_CHARGED_AMOUNT_VALUE DETAIL.WHOLESALE_CHARGED_AMOUNT_VALUE	Decimal	Read/Write	Contains the wholesale charged amount value for the assembled EDR.
NUMBER_OF_CDRS DETAIL.NUMBER_OF_CDRS	Integer	Write	The number of CDRs assembled in the EDR.

FCT_CancelTimer

The FCT_CancelTimer module checks the TimerID to identify the EDR and the timeout flag to verify if the EDR is valid or timed out. If the time out flag is set to **False**, FCT_CancelTimer cancels the timeout flag in the EDR so that the EDR can be sent for further processing.

If the timeout flag is set to **True**, it means there is a duplicate EDR and the FCT_CancelTimer discards the EDR.

Dependencies

FCT_CancelTimer depends on the FCT_Timer in the Dispatcher pipeline for the TimerID and the timeout flag values.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–28 lists the FCT_CancelTimer registry entries.

Table 36–28 FCT_CancelTimer Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
StreamName	The output queue to which the timed out EDR is sent.	Yes

Sample Registry

```
CancelTimer
{
  ModuleName = FCT_CancelTimer
  Module
```

```
    {
      Active = TRUE
      StreamName = ExceptionOutput
    }
}
```

EDR Container Fields

FCT_CancelTimer uses the EDR container fields listed in Table 36–29:

Table 36–29 FCT_CancelTimer Container Fields

Alias Field Name Default Field Name	Type	Access	Description
TIMER_ID DETAIL.TIMER_ID	Integer	Read	Contains the timer ID needed to cancel the timer

FCT_CarrierIcRating

The FCT_CarrierIcRating module adds roaming/interconnect data to EDRs for rating by the FCT_PreRating and FCT_MainRating modules.

See "About Linking Rate Plans to Network Operators and IC Products" in *BRM Configuring Roaming in Pipeline Manager*.

Dependencies

Requires a connection to the Pipeline Manager database.

All rating and mapping related modules should be placed in the pipeline.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–30 lists the FCT_CarrierIcRating registry entries.

Table 36–30 FCT_CarrierIcRating Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
EdrNetworkModel	Specifies the network model. This entry identifies your network as the home network. You can specify one home network per pipeline. You can use this entry in a semaphore file. See "About Linking Rate Plans to Network Operators and IC Products" in *BRM Configuring Roaming in Pipeline Manager*.	Yes
IcProductGroup	Specifies the IC product group that contains the IC products. This field is mandatory for all modes except CARRIER_IC mode. See "About Linking Rate Plans to Network Operators and IC Products" in *BRM Configuring Roaming in Pipeline Manager*.	Yes

Table 36–30 (Cont.) FCT_CarrierIcRating Registry Entries

Entry	Description	Mandatory
InterConnectDataModule	Specifies a connection to the DAT_Interconnect module. See "Connecting a Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes
Mode	Specifies the evaluation path for finding the IC-product. ■ If you specify ROAMING, IC products are found using the **IcProductGroup** registry entry. ■ If you specify CARRIER_IC, the module assigns a rate plan by using trunk information from the EDR.	Yes
RecordTypeField	Specifies the EDR field that contains the record type. The record type is used to search the IFW_ICPRODUCT_CNF database table for matching records. When processing CIBER record types, this entry is used to find the IC Products and the corresponding rate plan to use for rating the CIBER records.	No
UseRateplan	Specifies how the price is calculated: ■ **STANDARD**. The price is calculated using the specified rate plan. ■ **ALTERNATIVE**. The price is calculated using the alternative rate plan. If you entered an alternative rate plan when configuring the IC product, you can specify whether to use the alternate rate plan. You can use this entry in a semaphore file.	Yes

Sample Registry

```
Module
{
  Active = TRUE
  InterConnectDataModule = integRate.DataPool.InterConnect
  EdrNetworkModel = OWN
  UseRateplan = STANDARD
  Mode = ROAMING
  IcProductGroup = PG_ROAM
  RecordTypeField = DETAIL.ASS_CIBER_EXT.CIBER_RECORD_TYPE
}
```

Semaphore File Entries

Table 36–31 lists the FCT_CarrierIcRating Semaphore file entries.

Table 36–31 FCT_CarrierIcRating Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
EdrNetworkModel	Specifies the network model to be used (CODE from table IFW_NETWORKMODEL).
UseRateplan	■ **STANDARD:** IC-Price will be calculated using the rate plan from IFW_ICPRODUCT_RATE. ■ **ALTERNATIVE:** IC-Price will be calculated using the alternative rate plan from IFW_ICPRODUCT_RATE.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.
CarrierIcRating.Module.Active = TRUE

ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.
CarrierIcRating.Module.EdrNetworkModel = OTHER

ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.
CarrierIcRating.Module.UseRateplan = ALTERNATIVE
```

EDR Container Fields

Table 36–32 lists the FCT_CarrierIcRating EDR container fields.

Table 36–32 FCT_CarrierIcRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_CBD DETAIL.ASS_CBD	Block	Create	The associated charge breakdown record created to hold the mapping data.
ASS_CBD_CHARGE_PACKET DETAIL.ASS_CBD.CP	Block	Create	The charge packet created to hold the mapping data.
TRUNK_INPUT DETAIL.ASS_GSMW_EXT.TRUNK_INPUT	String	Read	Contains the input trunk search value from the IFW_TRUNK_CNF table.
TRUNK_OUTPUT DETAIL.ASS_GSMW_EXT.TRUNK_OUTPUT	String	Read	Contains the output trunk search value from the IFW_TRUNK_CNF table.
ASS_GSMW_ORIGINATING_SWITCH_IDENTIFICATION DETAIL.ASS_GSMW_EXT.ORIGINATING_SWITCH_IDENTIFICATION	String	Read	Contains the switch search value from the IFW_TRUNK_CNF table.
SOURCE_NETWORK DETAIL.SOURCE_NETWORK	String	Read	Contains the IC product search value from the IFW_ICPRODUCT_CNF table.
DESTINATION_NETWORK DETAIL.DESTINATION_NETWORK	String	Read	Contains the destination network search value from the IFW_ICPRODUCT_CNF table.

Table 36–32 (Cont.) FCT_CarrierIcRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
A_NUMBER DETAIL.A_NUMBER	String	Read	Contains the A number search value from the IFW_ICPRODUCT_CNF table.
B_NUMBER DETAIL.B_NUMBER	String	Read	Contains the B number search value from the IFW_ICPRODUCT_CNF table.
C_NUMBER DETAIL.C_NUMBER	String	Read	Contains the C number search value from the IFW_ICPRODUCT_CNF table.
RECORD_TYPE DETAIL.RECORD_TYPE	String	Read	Contains the record type search value from the IFW_ICPRODUCT_CNF table.
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code search value from the IFW_ICPRODUCT_CNF table.
INTERN_SERVICE_CLASS DETAIL.INTERN_SERVICE_CLASS	String	Read	Contains the internal service class search value from the IFW_ICPRODUCT_CNF table.
INTERN_USAGE_CLASS DETAIL.INTERN_USAGE_CLASS	String	Read	Contains the internal usage class search value from the IFW_ICPRODUCT_CNF table.
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging time stamp.
INTERN_A_NUMBER_ZONE DETAIL.INTERN_A_NUMBER_ZONE	String	Read	Contains the internal A number zone. This value sets the charge packet INTERN_ORIGIN_NUM_ZONE value.
INTERN_B_NUMBER_ZONE DETAIL.INTERN_B_NUMBER_ZONE	String	Read	Contains the internal B number zone. This value sets the charge packet INTERN_DESTIN_NUM_ZONE value.
ASS_CBD_RECORD_TYPE DETAIL.ASS_CBD.RECORD_TYPE	String	Write	Contains the record type: ■ **990** = CarrierIC ■ **991** = Roaming
INTERN_CALC_MODE DETAIL.ASS_CBD.INTERN_CALC_MODE	String	Write	Contains the calculation mode from the CALCMODE field in the IFW_NETWORKMODEL database table.
CHARGE_TYPE DETAIL.ASS_CBD.CP.CHARGE_TYPE	String	Write	Contains the charge type.
NETWORK_OPERATOR DETAIL.ASS_CBD.CP.NETWORK_OPERATOR_CODE	String	Write	Contains the network operator code from the CONNECTED_NO field in the IFW_TRUNK database table.
NETWORK_OPERATOR_BILLINGTYPE DETAIL.ASS_CBD.CP.NETWORK_OPERATOR_BILLINGTYPE	String	Write	Contains the billing type from the BILL_DIRECTION field in the IFW_ICPRODUCT_RATE database table.

Table 36–32 (Cont.) FCT_CarrierIcRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
PRODUCTCODE_USED DETAIL.ASS_CBD.CP.PRODUCTCODE_USED	String	Write	Contains the IC product code from the ICPRODUCT field in the IFW_ICPRODUCT database table.
TRUNK_USED DETAIL.ASS_CBD.CP.TRUNK_USED	String	Write	Contains the trunk from the TRUNK field in the IFW_ TRUNK database table.
POI_USED DETAIL.ASS_CBD.CP.POI_USED	String	Write	Contains the POI from the POI field in the IFW_POI database table.
ASS_CBD_CHARGING_START_TIMESTAMP DETAIL.ASS_CBD.CP.CHARGING_START_ TIMESTAMP	Date	Write	Contains the charging time stamp.
INTERN_FIX_COST DETAIL.ASS_CBD.CP.INTERN_FIX_COST	Decimal	Write	Contains the internal fixed cost. Added to the charge by the FCT_MainRating module.
INTERN_RATEPLAN DETAIL.ASS_CBD.CP.INTERN_RATEPLAN	String	Write	Contains the rate plan. Used by the FCT_PreRating and FCT_ MainRating modules.
INTERN_ORIGIN_NUM_ZONE DETAIL.ASS_CBD.CP.INTERN_ORIGIN_NUM_ZONE	String	Write	Contains the A number zone. Used by the FCT_PreRating module to find the impact category.
INTERN_DESTIN_NUM_ZONE DETAIL.ASS_CBD.CP.INTERN_DESTIN_NUM_ZONE	String	Write	Contains the B number zone. Used by the FCT_PreRating module to find the impact category.
INTERN_BILLING_CURRENCY DETAIL.ASS_CBD.CP.INTERN_BILLING_CURRENCY	String	Write	Contains the billing currency name
INTERN_HOME_CURRENCY DETAIL.ASS_CBD.CP.INTERN_HOME_CURRENCY	String	Write	Contains the home currency name

FCT_CiberOcc

The FCT_CiberOcc module creates a CIBER record for other charges and credits (OCC record), type 50 or 52.

See "About Processing CIBER OCC Records" in *BRM Configuring Roaming in Pipeline Manager*.

Dependencies

This module requires a connection to the DAT_InterConnect module.

Must run after the FCT_DuplicateCheck module and before the FCT_CarrierIcRating module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–33 lists the FCT_CiberOcc registry entries.

Table 36–33 FCT_CiberOcc Registry Entries

Entry	Description	Mandatory	Default
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes	N/A
CallRecordTypeField	Specifies the EDR field that indicates the CIBER record type. When processing CIBER record types, this entry is used to find the IC Products and the corresponding rate plan to use for rating the CIBER records.	No	DETAIL.ASS_CIBER_EXT.CIBER_RECORD_TYPE
EdrNetworkModel	Specifies the network model to use for CIBER_OCC searching. This identifies the home network You can specify one home network per pipeline. You can change this value by using a semaphore. See "About Settling Roaming Charges" in *BRM Configuring Roaming in Pipeline Manager*.	Yes	N/A
InterConnectDataModule	Specifies a connection to the DAT_InterConnect module. See "Connecting a Pipeline Manager Module to Another Module" in *BRM System Administrator's Guide*.	Yes	N/A
NoOCCField	Specifies the field that indicates whether to generate an OCC record. This field must match the field name specified in the **DuplicateIndicatorField** entry of the FCT_DuplicateCheck registry. See "FCT_DuplicateCheck".	Yes	DETAIL.ASS_CIBER_EXT.NO_OCC
OCCDescription	Description of the service associated with the OCC. **Important:** This field must not contain spaces. If you require spaces in the description, write an iScript to populate this field. See "Changing the Default Time Scheme" in *BRM Configuring Roaming in Pipeline Manager*.	No	" " (Empty string)
OCCIntervalIndicator	Specifies the interval at which the associated OCC record is generated. See "Changing the Default Time Scheme" in *BRM Configuring Roaming in Pipeline Manager*.	No	**3**

Sample Registry

```
CiberOcc
{
   ModuleName = FCT_CiberOcc
   Module
   }
      Active = TRUE
      InterConnectDataModule = ifw.DataPool.InterConnect
      EdrNetworkModel = ROAMING
      CallRecordTypeField = DETAIL.ASS_CIBER_EXT.CIBER_RECORD_TYPE
      NoOCCField = DETAIL.ASS_CIBER_EXT.NO_OCC
      OCCIntervalIndicator = 3
      OCCDescription = DAILY_SURCHARGE
    }
}
```

Semaphore File Entries

Table 36–34 lists the FCT_CiberOcc Semaphore file entries.

Table 36–34 FCT_CiberOcc Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
EdrNetworkModel	The network model to use for CIBER_OCC searching. See "About Settling Roaming Charges" in *BRM Configuring Roaming in Pipeline Manager*.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.
CiberOcc.Module.Active = false

ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.
CiberOcc.Module.EdrNetworkModel = ROAMING
```

EDR Container Fields

The FCT_CiberOcc module uses the EDR container fields listed in Table 36–35:

Table 36–35 FCT_CiberOcc EDR Container Fields

Default Field Name	Type	Access	Description
DETAIL.ASS_CIBER_EXT.AIR_CONNECT_TIME	Date	Read	Used to specify the connection time for the OCC record.
DETAIL.CHARGING_START_TIMESTAMP	Date	Read	IFW_CIBER_OCC
DETAIL.ASS_CIBER_EXT.CHARGE_NO_1_ CONNECT_TIME	Date	Read	Used to specify the connection time for the OCC record.

Table 36–35 (Cont.) FCT_CiberOcc EDR Container Fields

Default Field Name	Type	Access	Description
DETAIL.ASS_CIBER_EXT.CIBER_RECORD_TYPE	String	Read	This field is specified in the **CallRecordTypeField** entry in the registry. This is the default field used to determine the current EDR call record type.
DETAIL.ASS_CIBER_EXT.CIBER_RECORD_TYPE	String	Write	Specifies the type of record to create. If the current record type is: ■ 22 or 32, assign **52** to this field. ■ 10, 20, or 30, assign **50** to this field.
DETAIL.ASS_CIBER_EXT.CONNECT_TIME	Date	Write	This field is set to one of the following: ■ **AIR_CONNECT_TIME** if the call record type is 22. ■ **SSU_CONNECT_TIME** if the call record type is 10 or 20. ■ **CHARGE_NO_1_ CONNECT_TIME** if the call record type is 30 or 32.
DETAIL.ASS_CIBER_EXT.NO_OCC	String	Read	This field is specified in the **NoOCCField** entry in the registry.
DETAIL.ASS_CIBER_EXT.OCC_DESCRIPTION	String	Write	The value of this field is specified in the **OCCDescription** entry in the registry. The default value is an empty string: " "
DETAIL.ASS_CIBER_EXT.OCC_END_DATE	Date	Write	The value of **CHARGING_ START_TIMESTAMP** in the current EDR.
DETAIL.ASS_CIBER_EXT.OCC_INTERVAL_ INDICATOR	String	Write	The value of this field is specified in the **OCCIntervalIndicator** entry in the registry. The default value is **3** (daily interval).
DETAIL.ASS_CIBER_EXT.OCC_START_DATE	Date	Write	The value of **CHARGING_ START_TIMESTAMP** in the current EDR.
DETAIL.ASS_CIBER_EXT.RECORD_CREATE_DATE	Date	Write	This field is set to the system date.
DETAIL.ASS_CIBER_EXT.RECORD_USE_INDICATOR	String	Write	This field is set to **1**.

Table 36–35 (Cont.) FCT_CiberOcc EDR Container Fields

Default Field Name	Type	Access	Description
DETAIL.ASS_CIBER_EXT.SEQ_INDICATOR	String	Write	This field is set to **01**.
DETAIL.SOURCE_NETWORK	String	Read	Search value used for searching the IFW_CIBER_OCC database table.
DETAIL.ASS_CIBER_EXT.SSU_CONNECT_TIME	Date	Read	Used to specify the connection time for the OCC record.

Database Interface for the FCT_CiberOcc Module

The FCT_CiberOcc module uses the IFW_CIBER_OCC database table to determine whether OCC records are generated for the network operator. See "About processing CIBER OCC records" in *BRM Configuring Roaming in Pipeline Manager*.

For information about the fields in database tables, see the documentation in *Pipeline_Home***database**.

FCT_CliMapping

The FCT_CliMapping module maps multiple numbers to a single number for billing. See "Mapping Multiple Phone Numbers to a Single Number".

Dependencies

Must run before the rating modules.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–36 lists the FCT_CliMapping registry entries.

Table 36–36 FCT_CliMapping Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
MapFile	Specifies the path to the mapping file. You can use this entry in a semaphore file.	Yes

Sample Registry Entry

```
CliMapping
{
  ModuleName = FCT_CliMapping
  Module
   {
    Active = True
    MapFile = cli_map_1.dat
  }
```

```
}
```

Semaphore File Entries

Table 36–37 lists the FCT_CliMapping Semaphore file entries.

Table 36–37 FCT_CliMapping Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FucntionPool.CliMapping.Module.Active = True
```

EDR Container Fields

Table 36–38 lists the FCT_CliMapping EDR container fields.

Table 36–38 FCT_CliMapping EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
A_NUMBER DETAIL.A_NUMBER	String	Read/Write	Contains the customer A number.
CUST_A_ACCOUNT_ID DETAIL.CUST_A.ACCOUNT_ID	Block	Read	Contains the customer account ID.

FCT_CreditLimitCheck

The FCT_CreditLimitCheck module determines whether event owners have enough resources in their account balance to cover the cost of usage. If the account does not have sufficient resources to authorize the entire request, this module determines how much usage can be authorized with the available resources.

> **Note:** The FCT_CreditLimitCheck module does not check the credit floor.

For more information, see:

- Configuring a Real-Time Discounting Pipeline
- "About Determining whether there Are Sufficient Resources" in *BRM Telco Integration*
- "About Credit Limit Checks in the Real-Time Discounting Pipeline" in *BRM Telco Integration*
- Real-Time Discounting Architecture
- "How BRM Authorizes Users to Access Prepaid Services" in *BRM Telco Integration*

Dependencies

Use this module in a real-time discounting pipeline.

This module must run after all other discounting modules.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–39 lists the FCT_CreditLimitCheck registry entries.

Table 36–39 FCT_CreditLimitCheck Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive	Yes
CLCTrace	Specifies whether to generate a credit limit check trace file. **True** = Generate a trace file. **False** = Do not generate a trace file (Default)	No
CurrencyDataModule	Specifies the connection to the DAT_Currency module. See "Connecting a Pipeline Manager Module to Another Module" in *BRM System Administrator's Guide*.	Yes
RoundUpRequestQuantity	Determines whether authorized quantities are rounded up to remove fractional values. **True** = Round up **False** = No rounding (Default) See "Enabling Rounding for Maximum Quantity Results" in *BRM Telco Integration*.	No
StepValue	Specifies the step value to be considered for the quantity during reverse rating and for rounding the prorated quantity.	No

Sample Registry Entry

```
#------------------------------------------------
# Credit Limit Check
#------------------------------------------------
CreditLimitCheckModule
{
  ModuleName = FCT_CreditLimitCheck
  Module
  {
    Active  = True
    RoundUpRequestQuantity = True
    CLCTrace = True
    CurrencyDataModule = ifw.DataPool.CurrencyDataModule
    StepValue = 0.1
  }
}
```

EDR Container Fields

The FCT_CreditLimitCheck module uses the EDR container fields listed in Table 36–40:

Table 36–40 FCT_CreditLimitCheck EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
CREDIT_LIMIT_CHECK DETAIL.CREDIT_LIMIT_CHECK	Integer	Read	Specifies whether to perform a credit limit check on the EDR: 1 = check; 0 = don't check.
CREDIT_LIMIT_CHECK_RESULT DETAIL.CREDIT_LIMIT_CHECK_RESULT	Integer	Write	Specifies whether the credit limit check passed or failed: 1 = passed; 0 = failed.
INTERN_BALANCE_GROUP_ID DETAIL.INTERN_BALANCE_GROUP_ID	String	Read	Account level balance group of the event owner account.
DETAIL.UNRATED_QUANTITY	Decimal	Write	The quantity that could not be rated.
RESOURCE_ID DETAIL.ASS_CBD.CP.RESOURCE_ID	Integer	Read	Resource ID for the charge packet.
CHARGED_AMOUNT_VALUE DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE	Decimal	Read	The balance impact of the charge packet. This amount was computed by real-time rating.
QUANTITY_FROM DETAIL.ASS_CBD.CP.QUANTITY_FROM	Decimal	Read	Charge packet start quantity. If the charge packet is split by FCT_Discount, this module reads QUANTITY_FROM values from the DETAIL.ASS_CBD.CP.SPLIT_CP block.
QUANTITY_TO DETAIL.ASS_CBD.CP.QUANTITY_TO	Decimal	Read	Charge packet end quantity. If the charge packet is split by FCT_Discount, this module reads QUANTITY_TO values from the DETAIL.ASS_CBD.CP.SPLIT_CP block.
DETAIL.ASS_CBD.CP.ROUNDED_QTY_VALUE	Decimal	Read	The quantity that could be authorized.
DP_DISCOUNT_BALANCE_GROUP_ID DETAIL.ASS_CBD.DP.BALANCE_GROUP_ID	Integer	Read	POID of the balance group impacted by this discount packet.
DP_DISCOUNT_GRANTED_AMOUNT DETAIL.ASS_CBD.DP.GRANTED_AMOUNT	Decimal	Read	Total amount of the discount packet. This discount amount is applied to the balance group.

Table 36–40 (Cont.) FCT_CreditLimitCheck EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
DP_QUANTITY_FROM DETAIL.ASS_CBD.DP.QUANTITY_FROM	Decimal	Write	Discount packet start quantity. Aligns with the QUANTITY_FROM value in a charge packet or a split charge packet.
DP_QUANTITY_TO DETAIL.ASS_CBD.DP.QUANTITY_TO	Decimal	Write	Discount packet end quantity. Aligns with the QUANTITY_TO value in a charge packet or a split charge packet.
DP_DISCOUNT_RESOURCE_ID DETAIL.ASS_CBD.DP.RESOURCE_ID	Integer	Read	Resource ID for the discount packet.
BG_BG_ID DETAIL.CUST_A.BG.BALANCE_GROUP_ID	String	Read	Numeric ID for the balance group.
BG_BELEM_RESOURCE_ID DETAIL.CUST_A.BG.BAL_ELEM.RESOURCE_ID	Integer	Read	The resource ID for the balance group element.
BG_BELEM_CURR_BAL DETAIL.CUST_A.BG.BAL_ELEM.CURR_BAL	Decimal	Read	The event owner's current balance for this balance group element.
BG_BELEM_CREDIT_LIMIT DETAIL.CUST_A.BG.BAL_ELEM.CREDIT_LIMIT	Decimal	Read	The credit limit for this balance group element.
BG_BELEM_RESERVED_AMOUNT DETAIL.CUST_A.BG.BAL_ELEM.RESERVED_AMOUNT	Decimal	Read	The amount already in reserve by the event owner.
DETAIL.RUM_MAP.RUM_NAME	String	Read	Name of the RUM. Used in multi-RUM checks.
DETAIL.RUM_MAP.NET_QUANTITY	Decimal	Write	The total requested quantity for a RUM. Used in multi-RUM checks.
DETAIL.RUM_MAP.UNRATED_QUANTITY	Decimal	Write	The quantity of a RUM that could not be authorized. Used in multi-RUM checks.

FCT_CustomerRating

The FCT_CustomerRating module supplies rate plans for the FCT_MainRating module.

See the following documents:

- About Customer Rating
- About Multi-Segment Rating
- FCT_MainRating

The FCT_CustomerRating module is also used for least-cost rating and promotional overlays. See:

- About Least Cost Rating

- About Calculating the Promotional Savings

Dependencies

Requires a connection to the Pipeline Manager database.

This module must run after FCT_Account.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–41 lists the FCT_CustomerRating registry entries.

Table 36–41 FCT_CustomerRating Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
DefaultRateplan	Specifies the rate plan code used as default if no customer data for the A number can be found. You can use this entry in a semaphore file. See "Assigning a Default Rate Plan and Default Segment for Customer Rating".	No
DefaultSegment	Specifies the segment name used as default if no customer data for the A number can be found. You can use this entry in a semaphore file. See "Assigning a Default Rate Plan and Default Segment for Customer Rating".	No
LeastCostRating	Specifies whether least cost rating is active or inactive. For more information, see "About Least Cost Rating". ■ **True** activates least cost rating. ■ **False** disables least cost rating. See "Configuring Least Cost Rating".	No
Mode	Specifies if the module is run for customer rating (**CUSTOMER**) or segment rating (**SEGMENT**). See: ■ About Customer Rating ■ About Multi-Segment Rating	Yes
PromotionalSaving	Specifies whether to calculate the total savings to customers when rating a usage event with a promotional product rather than a base product. For more information, see "About Calculating the Promotional Savings". ■ **True** specifies to calculate the savings amount. ■ **False** specifies to not calculate the savings amount. See "About Calculating the Promotional Savings".	No

Sample Registry

```
CustomerRating
{
  ModuleName = FCT_CustomerRating
  Module
  {
    Active = True
    Mode = CUSTOMER
    DataConnection = ifw.DataPool.Database
  }
}
```

Semaphore File Entries

Table 36–42 lists the FCT_CustomerRating Semaphore file entries.

Table 36–42 FCT_CustomerRating Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
DefaultRateplan	Specifies the rate plan code used as default if no customer data for the A number can be found. See "Assigning a Default Rate Plan and Default Segment for Customer Rating".
DefaultSegment	Specifies the segment name used as default if no customer data for the A number can be found. See "Assigning a Default Rate Plan and Default Segment for Customer Rating".

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.CustomerRating.Module.Active = False
```

EDR Container Fields

Table 36–43 lists the FCT_CustomerRating EDR container fields.

Table 36–43 FCT_CustomerRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_CBD DETAIL.ASS_CBD	Block	Create	The associated charge breakdown record created to hold the rating data.
ASS_CBD_CHARGE_PACKET DETAIL.ASS_CBD_CHARGE_PACKET	Block	Create	The charge packet created to hold the rating data.
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging time stamp. Written to the DETAIL.ASS_CBD.CP.CHARGING_START_TIMESTAMP field.
INTERN_A_NUMBER_ZONE DETAIL.INTERN_A_NUMBER_ZONE	String	Read	Contains the A number zone. Written to the DETAIL.ASS_CBD.CP.INTERN_ORIGIN_NUM_ZONE field.
INTERN_B_NUMBER_ZONE DETAIL.INTERN_B_NUMBER_ZONE	String	Read	Contains the B number zone. Written to the DETAIL.ASS_CBD.CP.INTERN_DESTIN_NUM_ZONE field.
INTERN_SLA_RSC_GROUP DETAIL.INTERN_SLA_RSC_GROUP	String	Write	Contains the SLA RSC group.
INTERN_SLA_USC_GROUP DETAIL.INTERN_SLA_USC_GROUP	String	Write	Contains the SLA USC group code.
INTERN_SLA_IRULE_SET DETAIL.INTERN_SLA_IRULE_SET	String	Write	Contains the SLA iRule set.
ASS_CBD_RECORD_TYPE DETAIL.ASS_CBD.RECORD_TYPE	String	Write	Contains the record type for the associated charge breakdown record. ■ 981 = Customer rating ■ 984 = Multi-segment rating
CHARGE_TYPE DETAIL.ASS_CBD.CP.CHARGE_TYPE	String	Write	Contains the charge type. This field is always set to **N**.
ASS_CBD_ACCOUNT_CODE DETAIL.ASS_CBD.ACCOUNT_CODE	String	Write	Contains the account code. Set with the value from the DETAIL.CUST_A.ACCOUNT_NO field.

Table 36–43 (Cont.) FCT_CustomerRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_CBD_SYSTEM_BRAND_CODE DETAIL.ASS_CBD.SYSTEM_BRAND_CODE	String	Write	Contains the system brand code. Set with the value from the DETAIL.CUST_A.SYSTEM_BRAND field.
ASS_CBD_CUSTOMER_BILLCYCLE DETAIL.ASS_CBD.CUSTOMER_BILLCYCLE	String	Write	Contains the customer's bill cycle code. Set with data from the DETAIL.CUST_A.BILL_CYCLE field.
ASS_CBD_CUSTOMER_CURRENCY DETAIL.ASS_CBD.CUSTOMER_CURRENCY	String	Write	Contains the customer's currency. Set with the value from the DETAIL.CUST_A.CURRENCY field.
ASS_CBD_CUSTOMER_RATEPLAN_CODE DETAIL.ASS_CBD.CUSTOMER_RATEPLAN_CODE	String	Write	A comma-separated list of rate plan codes for all rating products. The list is arranged by product priority, with highest priority first and lowest priority last.
INTERN_BILLING_CURRENCY DETAIL.ASS_CBD.CP.INTERN_BILLING_CURRENCY	String	Write	Contains the billing currency. Set with the value from the DETAIL.CUST_A.CURRENCY field.
ASS_CBD_SEGMENT_CODE DETAIL.ASS_CBD.SEGMENT_CODE	String	Write	Contains the segment code. Set with the value from the Data Warehouse ERA.
ASS_CBD_SLA_CODE DETAIL.ASS_CBD.SLA_CODE	String	Write	Contains the SLA code. Set with the value from the Service Level Agreement ERA.
INTERN_DISCOUNT_ACCOUNT DETAIL.ASS_CBD.CP.INTERN_DISCOUNT_ACCOUNT	String	Write	Contains the discount account. Set with the value from the Discount Model ERA.
RATEPLAN_CODE DETAIL.ASS_CBD.CP.RATEPLAN_CODE	String	Write	Contains the rate plan code to use for rating.

Table 36–43 (Cont.) FCT_CustomerRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_CBD_CHARGING_START_TIMESTAMP DETAIL.ASS_CBD.CP.CHARGING_START_TIMESTAMP	Date	Write	Contains the charging time stamp. Set with the value from the DETAIL.CHARGING_START_TIMESTAMP field.
INTERN_RATEPLAN DETAIL.ASS_CBD.CP.INTERN_RATEPLAN	String	Write	Contains the internal rate plan code. Set with values from the Data Warehouse ERA.
INTERN_ORIGIN_NUM_ZONE DETAIL.ASS_CBD.CP.INTERN_ORIGIN_NUM_ZONE	String	Write	Contains the zone for the A number. Set with DETAIL.INTERN_A_NUMBER_ZONE
INTERN_DESTIN_NUM_ZONE DETAIL.ASS_CBD.CP.INTERN_DESTIN_NUM_ZONE	String	Write	Contains the zone for the B number. Set with DETAIL.INTERN_B_NUMBER_ZONE
CUST_A_ACCOUNT_ID DETAIL.CUST_A.ACCOUNT_ID	String	Read	Contains the customer account ID. Write to DETAIL.ASS_CBD.ACCOUNT_CODE
CUST_A_ACCOUNT_NO DETAIL.CUST_A.ACCOUNT_NO	String	Read	Contains the customer account number.
CUST_A_SYSTEM_BRAND DETAIL.CUST_A.SYSTEM_BRAND	String	Read	Contains the system brand. Written to the DETAIL.ASS_CBD.SYSTEM_BRAND_CODE field.
CUST_A_BILL_CYCLE DETAIL.CUST_A.BILL_CYCLE	String	Read	Contains the customer's bill cycle. Written to the DETAIL.ASS_CBD.CUSTOMER_BILLCYCLE field.
CUST_A_CURRENCY DETAIL.CUST_A.CURRENCY	String	Read	Contains the customer's currency. Written to the DETAIL.ASS_CBD.CUSTOMER_CURRENCY and DETAIL.ASS_CBD.CP.INTERN_BILLING_CURRENCY field.

Table 36–43 (Cont.) FCT_CustomerRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
CUST_A_INTERN_PP_INDEX DETAIL.CUST_A.INTERN_FOUND_PP_INDEX	Integer	Read	Contains an index of the customer's purchased products identified by the FCT_Account module.
CUST_A_RATEPLAN_NAME DETAIL.CUST_A.PRODUCT.RATEPLAN_NAME	String	Read	Contains the rate plan name. Written to the DETAIL.ASS_CBD.CP.RATEPLAN_CODE field.
CUST_A_PROFILE DETAIL.CUST_A.ERA.PROFILE	String	Read	Contains the customer's account-related ERA data.
CUST_A_KEY DETAIL.CUST_A.ERA.PA.KEY	String	Read	Contains the key for the account-related ERA data.
CUST_A_VALUE DETAIL.CUST_A.ERA.PA.VALUE	String	Read	Contains the value for the account-related ERA data.
CUST_A_PRODUCT_PROFILE DETAIL.CUST_A.PRODUCT.ERA.PROFILE	String	Read	Contains the customer's service-related ERA data.
CUST_A_PRODUCT_KEY DETAIL.CUST_A.PRODUCT.ERA.PA.KEY	String	Read	Contains the key for the service-related ERA data.
CUST_A_PRODUCT_KEY DETAIL.CUST_A.PRODUCT.ERA.PA.KEY	String	Read	Contains the value for the service-related ERA data.
DETAIL.CUST_A.INTERN_RATING_PRODUCTS	String	Read	Contains the product rating indexes. This is a comma-separated list of all rating products' indexes associated with the same service and event, and their priorities.
DETAIL.CUST_A.LEAST_COST	Integer	Write	Indicates whether to use least cost rating for an EDR. **1** turns it off; any other integer turns it on. This entry overrides the least cost rating entry in the registry file.
DETAIL.CUST_A.PRODUCT_PRIORITY	String	Read	Contains a list of the priorities for all products that are associated with the same service and event.

Table 36–43 (Cont.) FCT_CustomerRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
DETAIL.CUST_A.PRODUCT.USAGE_START	Date	Read	Contains a list of the start times for all products that are associated with the same service and event.
DETAIL.CUST_A.PROMOTIONAL_SAVING	Integer	Write	Indicates whether to calculate the promotional savings for an EDR. **1** turns off promotional savings; any other integer turns it on.
DETAIL.ASS_CBD.CP.RATEPLAN_CODE	String	Write	A comma-separated list of rate plan codes for all rating products. The list is arranged by product priority, with highest priority first and lowest priority last.

Database Tables

The FCT_CustomerRating module uses the IFW_SEGRATE_LNK database table for multi-segment rating.

For information about the fields in database tables, see the documentation in *Pipeline_Home***database**.

FCT_Dayrate

The FCT_Dayrate module calculates charges for special day rates, for example, a discount for calls made on January 1. See "About Special Day Rates" in *BRM Setting Up Pricing and Rating*.

Dependencies

Requires a connection to the DAT_Dayrate module.

This module must run after the FCT_MainRating module to adjust the rate.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–44 lists the FCT_Dayrate registry entries.

Table 36–44 FCT_Dayrate Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	No
DayrateDataModule	Specifies the connection to the DAT_Dayrate module. See "Connecting a Pipeline Manager Module to Another Module" in *BRM System Administrator's Guide* and "DAT_Dayrate".	No

Sample Registry

```
Dayrate
{
  ModuleName = FCT_Dayrate
  Module
  {
    Active = True
    DayrateDataModule = DayrateData
  }
}
```

Semaphore File Entries

Table 36–45 lists the FCT_Dayrate Semaphore file entry.

Table 36–45 FCT_Dayrate Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.SpecialDayRate.Module.Active = False
```

EDR Container Fields

Table 36–46 lists the FCT_Dayrate EDR container fields.

Table 36–46 FCT_Dayrate EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging time stamp. The field is used to determine which discount to use.
ASS_CBD.CP.INTERN_RATEPLAN DETAIL.ASS_CBD.CP.INTERN_RATEPLAN	String	Read	Contains the rate plan code. This field is used to determine which discount to use.
ASS_CBD.CP.INTERN_RATEPLAN_VERSION DETAIL.ASS_CBD.CP.INTERN_RATEPLAN_VERSION	Block	Read	Contains the rate plan version. This field is used to determine which discount to use.
ASS_CBD.CP.CHARGED_AMOUNT_VALUE DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE	Block	Read/Write	Contains the recalculated charged amount value.
ASS_CBD.CP DETAIL.ASS_CBD.CP	Block	Read	Contains the block index to charge packages.

FCT_Discard

The FCT_Discard module discards or skips EDRs based on configurable EDR properties.

- Skipping an EDR removes it from the pipeline.
- Discarding an EDR sends it to a different output stream.

In both the cases the state of the EDR becomes invalid.

See "Discarding and Skipping EDRs".

Dependencies

Requires a connection to the Pipeline Manager database.

Because you can discard or split EDRs based on service codes, this module should run after the FCT_ServiceCodeMap module. Should be early in the function pool, but must be run after FCT_CallAssembling.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–47 lists the FCT_Discard registry entries.

Table 36–47 FCT_Discard Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
Function	Specifies whether to discard or skip the EDR. Default = **Discard** You can use this entry in a semaphore file.	No
StreamName	Specifies the output stream for discarded EDRs. You can use this entry in a semaphore file. Default = **DevNull** See "Configuring Output of Discarded EDRs".	No

Sample Registry

```
Discard
{
   ModuleName = FCT_Discard
   Module
   {
      Active = True

      DataConnection = ifw.DataPool.Database
      Function = Discard
      StreamName = DevNull
   }
}
```

Semaphore File Entries

Table 36–48 lists the FCT_Discard Semaphore file entries.

Table 36–48 FCT_Discard Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
Function	Specifies whether to discard or skip the EDR. Default = **Discard**
Reload	Reloads the discard rules.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.EventDiscarding.Module.Reload {}
```

EDR Container Fields

Table 36–49 lists the FCT_Discard EDR container fields.

Table 36–49 FCT_Discard EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
RECORD_TYPE DETAIL.RECORD_TYPE	String	Read	Contains the record type.
SOURCE_NETWORK DETAIL.SOURCE_NETWORK	String	Read	Contains the source network code. This could either be PLMN ID or any logical operator code.
DESTINATION_NETWORK DETAIL.DESTINATION_NETWORK	String	Read	Contains the destination network code.
CALL_COMPLETION_INDICATOR DETAIL.CALL_COMPLETION_INDICATOR	String	Read	Indicates if a call was successfully completed.
LONG_DURATION_INDICATOR DETAIL.LONG_DURATION_INDICATOR	String	Read	Contains the long duration indicator: ■ **F** = First ■ **I** = Intermediate ■ **L** = Last
USAGE_CLASS DETAIL.USAGE_CLASS	String	Read	Contains the external usage class.
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code.
ASS_GSMW_EXT.ORIGINATING_SWITCH_IDENTIFICATION DETAIL.ASS_GPRS_EXT.ORIGINATING_SWITCH_IDENTIFICATION	String	Read	Contains the GSM MSC or Switch ID handling the origin of the call.
ASS_GPRS_EXT_ORIGINATING_SWITCH_IDENTIFICATION DETAIL.ASS_GPRS_EXT.ORIGINATING_SWITCH_IDENTIFICATION	String	Read	Contains the GPRS MSC or Switch ID handling the origin of the call.
TARIFF_CLASS DETAIL.TARIFF_CLASS	String	Read	Contains the tariff class.
TARIFF_SUB_CLASS DETAIL.TARIFF_SUB_CLASS	String	Read	Contains the tariff subclass.
CONNECT_SUB_TYPE DETAIL.CONNECT_SUB_TYPE	String	Read	Contains the connection subtype.
B_NUMBER DETAIL.B_NUMBER	String	Read	Contains the B number.
CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging time stamp.

Table 36–49 (Cont.) FCT_Discard EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
WHOLESALE_CHARGED_AMOUNT_VALUE DETAIL.WHOLESALE_CHARGED_AMOUNT_VALUE	Decimal	Read	Contains the sum of the wholesale charged amount value.
DISCARDING DETAIL.DISCARDING	Integer	Write	Indicates if the EDR should be discarded.
DETAIL.DISCARD_REASON	String	Write	Specifies the name of the discarding rule that applies to the EDR. Discarding rules are defined in the ifw_discarding table. If any of the rules in the table applies to the EDR, the EDR is discarded or skipped.

Database Tables

The FCT_Discard module uses the data in the IFW_Discarding table to determine which EDRs should be discarded. See "Discarding and Skipping EDRs".

To enter data in this table, use Pricing Center.

For information about the fields in database tables, see the documentation in *Pipeline_Home***database**.

Note: For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

FCT_Discount

The FCT_Discount module performs discounting operations. The module supports discounting for events rated in real time and events rated in batch by Pipeline Manager. See "About Discounts".

When the module is called for discount calculation as part of prepaid authorization or reauthorization, it splits charge packets if necessary to create linear segments to which a single net rate applies. See "Credit Limit Checks during Prepaid Authorization" in *BRM Telco Integration*.

Dependencies

Requires a connection to the following:

- The DAT_Discount module. See "DAT_Discount".
- A balance data module, either DAT_BalanceRealtime or DAT_BalanceBatch depending on whether the module is being used for real-time or batch discounting. See "DAT_BalanceRealtime" or "DAT_BalanceBatch".
- An account data module, either DAT_AccountRealtime or DAT_AccountBatch depending on whether the module is being used for real-time or batch

discounting. See "DAT_AccountRealtime" or "DAT_AccountBatch".

- The DAT_Currency module. This module converts resource codes to resource IDs. See "DAT_Currency".

For batch discounting, you must also configure the FCT_ApplyBalance module, which should be in the same function pool as this module for performance reasons and must run after this module.

This module must run after FCT_MainRating.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–50 lists the FCT_Discount registry entries.

Table 36–50 FCT_Discount Registry Entries

Entry	Description	Mandatory
AccountDataModule	Specifies the connection to the DAT_AccountRealtime or DAT_AccountBatch module. See "Connecting a Pipeline Manager Module to Another Module" in *BRM System Administrator's Guide*.	Yes
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry as a semaphore command.	Yes
AvoidMatchFactorCalculation	Specifies whether to calculate the amount of usage that is discounted (the match factor) for parallel and sequential discounts. The match factor is typically used only for cascading discounts. See "Calculating the Match Factor of Parallel and Sequential Discounts".	No
BackOut	Specifies if the module is used in a back-out pipeline for rerating. Default = **False** See "Configuring Rerating in Pipeline Manager" in *BRM Setting Up Pricing and Rating*.	No
BalanceDataModule	Specifies the connection to the DAT_BalanceRealtime or DAT_BalanceBatch module. See "Connecting a Pipeline Manager Module to Another Module" in *BRM System Administrator's Guide*.	Yes
CurrencyDataModule	Specifies the connection to DAT_Currency module. See "Connecting a Pipeline Manager Module to Another Module" in *BRM System Administrator's Guide*.	Yes
DiscountDataModule	Specifies the connection to the DAT_Discount module. See "Connecting a Pipeline Manager Module to Another Module" in *BRM System Administrator's Guide*.	Yes
DiscountMoreThanPossible	Specifies if a discount can be higher than the real revenue. Default = **False**	No

Table 36–50 (Cont.) FCT_Discount Registry Entries

Entry	Description	Mandatory
DiscountTrace	Specifies whether to generate discount trace file. ■ **True** indicates that a discount trace file is generated. ■ **False** indicates that a discount trace file is not generated. Default = **False**	No
EvalMultipleBalImpact	Specifies to create a discount balance impact for each charge packet when the discount base expression is not a simple expression (StepC, StepQ, TotalC, or TotalQ). Default = **False**	No
IgnoreEDROnDeadlock	Specifies whether to ignore EDRs causing the deadlock. ■ **True** indicates that the module should ignore the EDRs and continue processing the EDR file. The module places the EDR causing the deadlock into the discountError directory. ■ **False** indicates that the module should roll back already processed EDRs and start reprocessing the same file.	No
IgnoreEDROnDiscountError	Specifies whether to ignore EDRs with discounting error. ■ **True** indicates that the module should ignore the EDRs with discounting error and continue processing the remaining EDRs in the EDR file. The module places the EDRs that failed discounting into the discountError directory. ■ **False** indicates that the module should roll back the EDRs with discounting error that has already been processed and reprocess them. Default = **False**	No
IgnoreEDROnLock	Specifies whether to ignore the EDR if the balance group object is locked by another transaction. The ignored or rejected EDRs with the locked balance group are placed into the **discountError** directory. ■ **True** indicates that the module ignores the EDRs and continues processing the EDR file. ■ **False** indicates that the module rolls back already processed EDRs and begins reprocessing the same file. Default = **False**	No
ShowZeroDiscount	Specifies whether Discount Packet are generated when granted discount amount is 0. ■ **True** indicates that Discount Packets are generated when the granted discount amount is 0. ■ **False** indicates that Discount Packets are not generated when the granted discount amount is 0. Default = **False**	No
SupportBundleERA	Specifies if the support for ERAs is needed. Default = **False**	No

Table 36–50 (Cont.) FCT_Discount Registry Entries

Entry	Description	Mandatory
TaxationMode	Used only when FCT_Discount is configured for discounting in the real-time pipeline. Specifies when events are taxed. The value of this entry should be the same as the value of **taxation_switch** entry in the Connection Managers (CM) configuration file. See "Enabling Taxation during Real-Time Tating or Billing" in *BRM Calculating Taxes*. Possible values are: **0** - Taxation is disabled. **1** - Enable real-time tax calculation. **2** - Enable deferred (cycle-time) tax calculation. **3** - Enable real-time and deferred tax calculation. Default = **3**	No
ZeroValuePacketFilterDisabled	Used with the aggregation scenario to filter out charge packets where either charge or quantity is zero. Default = **True**	No
ProrateFixedDiscount	Specifies whether to prorate fixed cycle-event discounts: ■ **False** indicates that fixed cycle-event discounts are not prorated. This is the default. ■ **True** indicates that fixed cycle-event discounts are prorated. To prorate fixed cycle-event discounts, you must set this entry to **True** before starting the real-time pipeline.	No

Sample Registry

```
GeneralDiscounting
{
  ModuleName = FCT_Discount
  Module
  {
    Active                      = TRUE
    DiscountDataModule          = ifw.DataPool.DiscountModelDataModule
    BalanceDataModule           = ifw.DataPool.BalanceDataModule
    AccountDataModule           = ifw.DataPool.CustomerData
    CurrencyDataModule          = ifw.DataPool.CurrencyDataModule
    DiscountMoreThanPossible    = False
    ProrateFixedDiscount        = True
    TaxationMode                = 3
    EvalMultipleBalImpact       = True
  }
}
```

Semaphore File Entries

Table 36–51 lists the FCT_Discount Semaphore file entries.

Table 36–51 FCT_Discount Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
IgnoreEDROnLock	Specifies whether to ignore the EDR if the balance group object is locked by another transaction. The ignored or rejected EDRs with the locked balance group are placed into the **discountError** directory. ■ **True** indicates that the module ignores the EDRs and continues processing the EDR file. ■ **False** indicates that the module rolls back already processed EDRs and begins reprocessing the same file. Default = **False**

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ApolloDiscountModule.
Module.Active = False
```

EDR Container Fields

Table 36–52 lists the FCT_Discount EDR container fields.

Table 36–52 FCT_Discount EDR Container Fields

Entry	Format	Access	Description
BDR_CHARGING_END_TIMESTAMP DETAIL.CHARGING_END_TIMESTAMP	Date	Read	Used to calculate the duration.
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Used to calculate the duration.
CREDIT_LIMIT_CHECK DETAIL.CREDIT_LIMIT_CHECK	Integer	Read	Determines whether the module applies discounts normally or as part of a credit limit check. If set to **1**, the module applies discounts as part of a credit limit check. It set to **0** (or any value other that **1**), the module operates normally.
ERROR_REJECT_TYPE DETAIL.ERROR_REJECT_TYPE	String	Read	Used by FCT_Reject to reject the DETAIL to a stream other than the standard reject stream.
EVENT_TYPE DETAIL.EVENT_TYPE	String	Read	BRM event type.
EVENT_BALANCE_GROUP_ID DETAIL.INTERN_BALANCE_GROUP_ID	String	Read	POID of the balance group charged.

Table 36–52 (Cont.) FCT_Discount EDR Container Fields

Entry	Format	Access	Description
INTERN_PROCESS_STATUS DETAIL.INTERN_PROCESS_STATUS	Integer	Read	Process status. If set to **2**, a recycle test is in progress, and this container is skipped.
INTERN_USAGE_CLASS DETAIL.INTERN_USAGE_CLASS	Date	Read	Usage class. This is used for matching the usage class in the discount detail.
REFRESH_BALANCE DETAIL.REFRESH_BALANCE	Integer	Read	Specifies whether the latest balance information should be retrieved from the database. When this field is set, this module calls the balance module to get the latest balance information from the database, whether or not a balance packet is present in the EDR.
USAGE_TYPE DETAIL.USAGE_TYPE	Date	Read	Usage type. Used for matching the usage type in the discount detail.
BDR_UTC_TIME_OFFSET DETAIL.UTC_TIME_OFFSET	String	Read	UTC time offset, if the discount owner is not the A customer.
FU_DISCOUNT_OBJECTS DETAIL.ASS_CBD.FU_DISCOUNT_OBJECTS	String	Write	ID of the account's discounts that have first-usage start times which were used to discount the event.
ASS_CBD_RECORD_TYPE DETAIL.ASS_CBD.RECORD_TYPE	String	Read	Record type. Used for matching the record type in the discount detail.
CHARGED_CURRENCY DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_CURRENCY	String	Read	Currency of the charge.
CHARGED_AMOUNT_VALUE DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE	Decimal	Read	Amount of the charge. Used as the base value for discounting.
DISCOUNTMODEL_CODE DETAIL.ASS_CBD.CP.DISCOUNTMODEL_CODE	String	Read	Discount model. Used for matching the discount model in the discount detail.
ASS_CBD_IMPACT_CATEGORY DETAIL.ASS_CBD.CP.IMPACT_CATEGORY	String	Read	Impact category Used for matching the impact category in the discount detail.
CP_DISCOUNT_PACKET_INDEX DETAIL.ASS_CBD.CP.INTERN_PACKET_INDEX	Integer	Read	Packet ID.

Table 36–52 (Cont.) FCT_Discount EDR Container Fields

Entry	Format	Access	Description
PRICEMODEL_CODE DETAIL.ASS_CBD.CP.PRICEMODEL_CODE	String	Read	Price model code. Used for matching the price model in the discount detail.
QUANTITY_FROM DETAIL.ASS_CBD.CP.QUANTITY_FROM	Decimal	Read	Charge packet start quantity. Used to determine whether to split charge packets.
QUANTITY_TO DETAIL.ASS_CBD.CP.QUANTITY_TO	Decimal	Read	Charge packet end quantity. Used to determine whether to split charge packets.
RATEPLAN_CODE DETAIL.ASS_CBD.CP.RATEPLAN_CODE	String	Read	Rate plan code. Used for filtering in the discount detail.
RATETAG_CODE DETAIL.ASS_CBD.CP.RATETAG_CODE	String	Read	Concatenation of rate plan definition. Used for filtering in the discount detail.
RESOURCE DETAIL.ASS_CBD.CP.RESOURCE	String	Read	Resource code. Used for filtering in the discount detail.
RESOURCE_ID DETAIL.ASS_CBD.CP.RESOURCE_ID	Integer	Read	Numeric ID of the resource. Used for filtering in the discount detail.
ROUNDED_QUANTITY_FROM DETAIL.ASS_CBD.CP.ROUNDED_QUANTITY_FROM	Decimal	Read	From quantity after rounding.
ROUNDED_QUANTITY_TO DETAIL.ASS_CBD.CP.ROUNDED_QUANTITY_TO	Decimal	Read	To quantity after rounding.
ROUNDED_QUANTITY_VALUE DETAIL.ASS_CBD.CP.ROUNDED_QUANTITY_VALUE	Decimal	Read	Rounded quantity. Used as the base value for discounting.
RUM DETAIL.ASS_CBD.CP.RUM	String	Read	RUM name. Used for matching the RUM in the discount detail.
SERVICE_CLASS_USED DETAIL.ASS_CBD.CP.SERVICE_CLASS_USED	String	Read	Service class. Used for matching the service class in the discount detail.
SERVICE_CODE_USED DETAIL.ASS_CBD.CP.SERVICE_CODE_USED	String	Read	Service code. Used for matching the service code in the discount detail.
TIMEMODEL_CODE DETAIL.ASS_CBD.CP.TIMEMODEL_CODE	String	Read	Time model code. Used for matching the time model in the discount detail.
TIMEZONE_CODE DETAIL.ASS_CBD.CP.TIMEZONE_CODE	String	Read	Time zone code. Used for matching the time zone in the discount detail.

Table 36–52 (Cont.) FCT_Discount EDR Container Fields

Entry	Format	Access	Description
USAGE_GL_ACCOUNT_CODE DETAIL.ASS_CBD.CP.USAGE_GL_ACCOUNT_CODE	String	Read	G/L code. Used for matching the G/L code in the discount detail.
ASS_CBD_ZONEMODEL_CODE DETAIL.ASS_CBD.CP.ZONEMODEL_CODE	String	Read	Zone model code. Used for matching the zone model in the discount detail.
DETAIL.ASS_CBD.CP.SPLIT_CP	Sub-block	Write	Optional sub-block added when charge packets are split.
DETAIL.ASS_CBD.CP.SPLIT_CP.RESOURCE_ID	Integer	Write	Numeric ID of the resource. Used for filtering in the discount detail. Copied from the original charge packet.
DETAIL.ASS_CBD.CP.SPLIT_CP.RUM	String	Write	RUM name. Copied from the original charge packet.
DETAIL.ASS_CBD.CP.SPLIT_CP.QUANTITY_FROM	Decimal	Write	Split charge packet start quantity. Calculated by the module.
DETAIL.ASS_CBD.CP.SPLIT_CP.QUANTITY_TO	Decimal	Write	Split charge packet end quantity. Calculated by the module.
DETAIL.ASS_CBD.CP.SPLIT_CP.CHARGED_AMOUNT_VALUE	Decimal	Write	Amount of the charge for this split charge packet. Calculated by the module.
DETAIL.ASS_CBD.CP.SPLIT_CP.INTERN_PACKET_INDEX	Integer	Write	The index of the split charge packet.
DETAIL.ASS_CBD.CP.SPLIT_CP.INTERN_SRC_PACKET_INDEX	Integer	Write	The packet index of the charge packet from which this split charge packet was generated.
DP_DISCOUNT_BALANCE_GROUP_ID DETAIL.ASS_CBD.DP.BALANCE_GROUP_ID	String	Write	POID of the balance group impacted by this discount packet.
DP_CREATED DETAIL.ASS_CBD.DP.CREATED	String	Write	Creation date of the element.
DP_DISCOUNTBALIMPACTID DETAIL.ASS_CBD.DP.DISCOUNTBALIMPACTID	Integer	Write	Discount balance impact ID.
DP_DISCOUNTMODEL DETAIL.ASS_CBD.DP.DISCOUNTMODEL	String	Write	Discount model used to create this discount packet.
DP_DISCOUNTRULE DETAIL.ASS_CBD.DP.DISCOUNTRULE	String	Write	Discount rule used to create this discount packet
DP_DISCOUNTSTEPID DETAIL.ASS_CBD.DP.DISCOUNTSTEPID	Integer	Write	Discount step used to create this discount packet

Table 36–52 (Cont.) FCT_Discount EDR Container Fields

Entry	Format	Access	Description
DP_DISCOUNT_GLID DETAIL.ASS_CBD.DP.GLID	String	Write	G/L ID as specified in the discount balance impact, otherwise copied from the charge packet.
DP_GRANTED_AMOUNT DETAIL.ASS_CBD.DP.GRANTED_AMOUNT	Decimal	Write	Granted discount/sponsorship amount. Can be currency or non-currency.
DP_GRANTED_QUANTITY DETAIL.ASS_CBD.DP.GRANTED_QUANTITY	Decimal	Write	Discount base value used to compute granted amount.
DP_DISCOUNT_IMPACT_CATEGORY DETAIL.ASS_CBD.DP.IMPACT_CATEGORY	String	Write	Impact category specified for this discount packet, otherwise copied from the charge packet.
DP_INTERN_DISC_MATCH_FACTOR DETAIL.ASS_CBD.DP.INTERN_DISC_MATCH_FACTOR	Decimal	Write	Discount match factor
DP_DISCOUNT_PACKET_INDEX DETAIL.ASS_CBD.DP.INTERN_PACKET_INDEX	Integer	Write	Packet ID.
DP_DISCOUNT_SRC_PACKET_INDEX DETAIL.ASS_CBD.DP.INTERN_SRC_PACKET_INDEX	Integer	Write	Source packet ID.
DP_INTERN_TOTAL_MATCH_FACTOR DETAIL.ASS_CBD.DP.INTERN_TOTAL_MATCH_FACTOR	Decimal	Write	Total discounted match factor.
DP_NODE_LOCATION DETAIL.ASS_CBD.DP.NODE_LOCATION	String	Write	Node location.
DP_OBJECT_ACCOUNT DETAIL.ASS_CBD.DP.OBJECT_ACCOUNT	Integer	Write	POID of discount owner.
DP_OBJECT_ID DETAIL.ASS_CBD.DP.OBJECT_ID	String	Write	Discount/sponsor object ID.
DP_OBJECT_OWNER_ID DETAIL.ASS_CBD.DP.OBJECT_OWNER_ID	Integer	Write	POID of the account or service that owns the discount.
DP_OBJECT_OWNER_TYPE DETAIL.ASS_CBD.DP.OBJECT_OWNER_TYPE	String	Write	POID type of discount owner, **/account** or **/service**.
DP_OBJECT_TYPE DETAIL.ASS_CBD.DP.OBJECT_TYPE	String	Write	Discount/sponsor object that generated this discount.
DP_DISCOUNT_PRICEMODEL_CODE DETAIL.ASS_CBD.DP.PRICEMODEL_CODE	String	Write	Price model used to generate this discount.
DP_DISCOUNT_QUANTITY DETAIL.ASS_CBD.DP.QUANTITY	Decimal	Write	Discounted non-currency amount.

Table 36–52 (Cont.) FCT_Discount EDR Container Fields

Entry	Format	Access	Description
DP_QUANTITY_FROM DETAIL.ASS_CBD.DP.QUANTITY_FROM	Decimal	Write	Discount packet start quantity. Aligns with the QUANTITY_FROM value in a charge packet or a split charge packet.
DP_QUANTITY_TO DETAIL.ASS_CBD.DP.QUANTITY_TO	Decimal	Write	Discount packet end quantity. Aligns with the QUANTITY_TO value in a charge packet or a split charge packet.
DP_DISCOUNT_RATEPLAN DETAIL.ASS_CBD.DP.RATEPLAN	String	Write	Rate plan used to generate this discount.
DP_DISCOUNT_RATETAG DETAIL.ASS_CBD.DP.RATETAG	String	Write	Concatenation of the definition of the rate plan used to generate this discount.
DP_DISCOUNT_RESOURCE DETAIL.ASS_CBD.DP.RESOURCE_ID	Integer	Write	Resource ID of the resource impacted by this discount.
DP_DISCOUNT_RUM DETAIL.ASS_CBD.DP.RUM	String	Write	The RUM entered for filtering in the discount master.
DP_DISCOUNT_SERVICE_CLASS DETAIL.ASS_CBD.DP.SERVICE_CLASS	String	Write	Service class entered for filtering in the discount master.
DP_DISCOUNT_SERVICE_CODE DETAIL.ASS_CBD.DP.SERVICE_CODE	String	Write	Service code entered for filtering in the discount master.
DP_DISCOUNT_SUB_BALANCE.GRANTOR DETAIL.ASS_CBD.DP.SUB_BALANCE.GRANTOR	String	Read	ID of the product or discount that granted this resource.
DP_DISCOUNT_SUB_BALANCE.VALID_FROM_DETAILS DETAIL.ASS_CBD.DP.SUB_BALANCE.VALID_FROM_DETAILS	Integer	Read	The sub-balance start time mode (such as first-usage or relative) and relative offset details. This field is used in conjunction with SUB_BALANCE_VALID_FROM to determine the validity period start time.
DETAIL.ASS_CBD.DP.SUB_BALANCE.VALID_TO_DETAILS	Integer	Read	The sub-balance end time mode (such as relative) and relative offset details. This field is used in conjunction with SUB_BALANCE_VALID_TO to determine the validity period end time.

Table 36–52 (Cont.) FCT_Discount EDR Container Fields

Entry	Format	Access	Description
DP_DISCOUNT_TAX_CODE DETAIL.ASS_CBD.DP.TAX_CODE	String	Write	Tax code as specified in the discount balance impact. If not specified in the balance impact, copied from the charge packet for this discount packet.
DP_DISCOUNT_TIMEMODEL_CODE DETAIL.ASS_CBD.DP.TIMEMODEL_CODE	String	Write	Time model entered for filtering in the discount master.
DP_DISCOUNT_TIMEZONE_CODE DETAIL.ASS_CBD.DP.TIMEZONE_CODE	String	Write	Time zone entered for filtering in the discount master.
DP_DISCOUNT_VALID_FROM DETAIL.ASS_CBD.DP.VALID_FROM	Date	Write	Valid-from date for the grant in the discount packet.
DP_DISCOUNT_VALID_TO DETAIL.ASS_CBD.DP.VALID_TO	Date	Write	Valid-to date for the grant in the discount packet.
DP_DISCOUNT_VALID_FROM_DETAIL DETAIL.ASS_CBD.DP.VALID_FROM_DETAIL	Integer	Read	The valid-from mode (such as first usage or relative) and relative offset details. This field is used in conjunction with PIN_FLD_VALID_FROM to determine the validity period start time.
DP_DISCOUNT_VALID_TO_DETAIL DETAIL.ASS_CBD.DP.VALID_TO_DETAIL	Integer	Read	The valid-to mode (such as relative) and relative offset details. This field is used in conjunction with PIN_FLD_VALID_TO to determine the validity period end time.
DP_DISCOUNT_ZONEMODEL_CODE DETAIL.ASS_CBD.DP.ZONEMODEL_CODE	String	Write	Zone model code enter for filtering in the discount master.
SUB_BAL_AMOUNT DETAIL.ASS_CBD.DP.SUB_BALANCE.AMOUNT	Decimal	Write	Amount of a sub-balance impacted by the discount packet.
SUB_BAL_CONTRIBUTOR DETAIL.ASS_CBD.DP.SUB_BALANCE.CONTRIBUTOR	String	Write	Contributor to a sub-balance impacted by the discount packet.
SUB_BAL_REC_ID DETAIL.ASS_CBD.DP.SUB_BALANCE.REC_ID	Integer	Write	ID of a sub-balance impacted by the discount packet.
SUB_BAL_VALID_FROM DETAIL.ASS_CBD.DP.SUB_BALANCE.VALID_FROM	Date	Write	Beginning validity date for a sub-balance impacted by the discount packet.

Table 36–52 (Cont.) FCT_Discount EDR Container Fields

Entry	Format	Access	Description
SUB_BAL_VALID_TO DETAIL.ASS_CBD.DP.SUB_BALANCE.VALID_TO	Date	Write	End validity date for a sub-balance impacted by the discount packet.
ACCOUNT_PARENT_ID DETAIL.CUST_A.ACCOUNT_PARENT_ID	String	Read	Customer account POID.
ACTG_LAST_DATE DETAIL.CUST_A.ACTG_LAST_DATE	Date	Read	The date that the current monthly cycle began.
ACTG_NEXT_DATE DETAIL.CUST_A.ACTG_NEXT_DATE	Date	Read	Date that the current monthly cycle ends.
ACTG_USED_DATE DETAIL.CUST_A.ACTG_USED_DATE	Date	Read	Date used for this EDR.
FIRST_USGAE_INDICATOR DETAIL.CUST_A.DL.PD.FIRST_USAGE_INDICATOR	Integer	Read	Specifies whether the discount's validity period is configured to start when first used.
DETAIL.CUST_A.DL.PD.USAGE_END		Write	
INTERN_FOUND_PP_INDEX DETAIL.CUST_A.INTERN_FOUND_PP_INDEX	Integer	Read	Purchased product index of the product or service.
SERVICE_ID DETAIL.CUST_A.PRODUCT.SERVICE_ID	String	Read	POID of the service instance.
BG_BG_ID DETAIL.CUST_A.BG.BALANCE_GROUP_ID	String	Read	POID of the balance group for the account.
BG_BELEM_CURR_BAL DETAIL.CUST_A.BG.BAL_ELEM.CURR_BAL	Decimal	Read	Current balance of the balance group element.
BG_BELEM_RESOURCE_ID DETAIL.CUST_A.BG.BAL_ELEM.RESOURCE_ID	Integer	Read	Resource ID of the balance group element.
DISCOUNT_BALANCE_GROUP_ID DETAIL.CUST_A.DL.BALANCE_GROUP_ID	String	Read	Balance group used to evaluate this discount instance.
DISCOUNT_OWNER_ACCT_ID DETAIL.CUST_A.DL.DISCOUNT_OWNER_ACCT_ID	String	Read	POID of the account that owns the discount, either directly or indirectly through a service.
DISOUNT_OWNER_ID DETAIL.CUST_A.DL.DISCOUNT_OWNER_ID	String	Read	POID of the account or service that owns the discount. Identical to DISCOUNT_OWNER_ACCT_ID if the discount is owned directly by an account.
DISOUNT_OWNER_TYPE DETAIL.CUST_A.DL.DISCOUNT_OWNER_TYPE	String	Read	Discount owner type, either **/account** or **/service**.
PD_DISCOUNTID DETAIL.CUST_A.DL.PD.DISCOUNT_ID	String	Read	POID of the discount object.

Table 36–52 (Cont.) FCT_Discount EDR Container Fields

Entry	Format	Access	Description
PD_DISCOUNTMODEL DETAIL.CUST_A.DL.PD.DISCOUNT_MODEL	String	Read	Code of a discount model referenced in the discount object.
PD_FLAGS DETAIL.CUST_A.DL.PD.FLAGS	Integer	Read	Proration setting.
PD_MODE DETAIL.CUST_A.DL.PD.MODE	Integer	Read	Discount mode, either cascading or parallel.
PD_NODE_LOCATION DETAIL.CUST_A.DL.PD.NODE_LOCATION	String	Read	Unique ID of the discount object.
PD_QUANTITY DETAIL.CUST_A.DL.PD.QUANTITY	Integer	Read	Number of purchased discounts. This is multiplied by balance impact of this discount instance.
PD_SCALE DETAIL.CUST_A.DL.PD.SCALE	Decimal	Read	Proration scale.
PD_VALID_FLAG DETAIL.CUST_A.DL.PD.VALID_FLAG	Integer	Read	Indicates whether the discount is valid.
SPONSOR_BALANCE_GROUP_ID DETAIL.CUST_A.SL.BALANCE_GROUP_ID	String	Read	POID of the charge share group balance group.
SPONSOR_OWNER_ACCT_ID DETAIL.CUST_A.SL.SPONSOR_OWNER_ACCT_ID	String	Read	POID of the account that owns the chargeshare object, either directly or indirectly through a service.
SPONSOR_OWNER_ID DETAIL.CUST_A.SL.SPONSOR_OWNER_ID	String	Read	POID of the account or service that owns the chargeshare. Identical to SPONSOR_OWNER_ACCT_ID if the chargeshare is owned directly by an account.
SPONSOR_OWNER_TYPE DETAIL.CUST_A.SL.SPONSOR_OWNER_TYPE	String	Read	Chargeshare owner type, either **/account** or **/service**.
SD_DISCOUNTMODEL DETAIL.CUST_A.SL.SD.DISCOUNT_MODEL	String	Read	Discount model used for this chargeshare.
SD_SPONSORID DETAIL.CUST_A.SL.SD.SPONSORSHIP_ID	String	Read	The POID of the chargeshare (**/sponsorship**) object.
SD_VALID_FLAG DETAIL.CUST_A.SL.SD.VALID_FLAG	Integer	Read	Indicates whether the chargeshare is valid.

Table 36–52 (Cont.) FCT_Discount EDR Container Fields

Entry	Format	Access	Description
DETAIL.CUST_A.SUBCYCLE_PL.DL.PD.USAGE_END	N/A	Read	N/A
DETAIL.CUST_A.SUBCYCLE_PL.DL.PD.FIRST_USAGE_INDICATOR	N/A	Read	N/A
TRANSACTION_ID INTERNAL.TRANSACTION_ID	Decimal	Read	The transaction ID. Needed for queuing.

FCT_DiscountAnalysis

The FCT_DiscountAnalysis module determines which discounts apply to a given event.

See:

- Pipeline Discounting Architecture
- Configuring Discounting Modules and Components

Dependencies

The FCT_DiscountAnalysis module requires a connection to the following data modules:

- DAT_Discount
- DAT_ModelSelector

For pipeline rating, this module must run after the FCT_Account module and before the FCT_Discount module.

For real-time rating, this module must run before the FCT_Discount module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–53 lists the FCT_DiscountAnalysis registry entries.

Table 36–53 FCT_DiscountAnalysis Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes

Table 36–53 (Cont.) FCT_DiscountAnalysis Registry Entries

Entry	Description	Mandatory
DiscountModelDataModule	Specifies the connection to the DAT_Discount module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes
Filter_SetModule	Specifies the connection to the FCT_Filter_Set module. Use this entry if the FCT_Filter_Set module is configured. See: ■ About Using Filter Sets to Apply System Products and Discounts ■ FCT_Filter_Set	No
ModelSelectorDataModule	Specifies the connection to the DAT_ModelSelector module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes

Sample Registry

```
Discount
{
    ModuleName = FCT_DiscountAnalysis
    Module
    {
       Active = True
       DiscountModelDataModule = ifw.DataPool.DiscountModelDataModule
       Filter_SetModule = ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.Filter_Set
       ModelSelectorDataModule = ifw.DataPool.ModelSelectorDataModule
    }
}
```

Semaphore File Entries

Table 36–54 lists the FCT_DiscountAnalysis Semaphore file entry.

Table 36–54 FCT_DiscountAnalysis Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.DiscountAnalysisModule.
Module.Active = False
```

EDR Container Fields

Table 36–55 lists the FCT_DiscountAnalysis EDR container fields.

Table 36–55 FCT_DiscountAnalysis EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_CBD DETAIL.ASS_CBD	Block	R	n/a
ASS_CBD_CHARGE_PACKET DETAIL.ASS_CBD.CP	Block	R	Charge packet; used to check for any discount ERAs set.
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	R	Event start time.
BDR_CHARGING_END_TIMESTAMP DETAIL.CHARGING_END_TIMESTAMP	Date	R	Event end time.
EVENT_TYPE DETAIL.EVENT_TYPE	String	R	The event type used to locate the event discount.
BDR_UTC_TIME_OFFSET DETAIL.UTC_TIME_OFFSET	String	R	The UTC offset used to adjust the start and end time.
INTERN_BALANCE_GROUP_ID DETAIL.INTERN_BALANCE_GROUP_ID	String	RW	Account level balance group of the event owner account.
INTERN_DISCOUNT_OWNER_ACCT_ID DETAIL.INTERN_DISCOUNT_OWNER_ACCT_ID	String	RW	Event owner account ID.
DISCOUNT_LIST DETAIL.CUST_A.DL	Block	R	Purchased discounts that belong to an account or service, or are shared by an account or service.
BALANCE_GROUP_ID DETAIL.CUST_A.DL.BALANCE_GROUP_ID	String	RW	ID of the balance group whose resources are used for the discounts in the discount list.
DISCOUNT_OWNER_ACCT_ID DETAIL.CUST_A.DL.DISCOUNT_OWNER_ACCT_ID	String	RW	ID of the account that owns the set of purchased discounts in the discount list.
PURCHASED_DISCOUNTS DETAIL.CUST_A.DL.PD	String	CW	Information about the discount.
DISCOUNT_ID DETAIL.CUST_A.DL.PD.DISCOUNT_ID	String	RW	Discount object ID.
STATUS DETAIL.CUST_A.DL.PD.STATUS	String	RW	Discount state (active, inactive, or cancelled).
PURCHASE_START DETAIL.CUST_A.DL.PD.PURCHASE_START	Date	RW	Discount purchase start time.
PURCHASE_END DETAIL.CUST_A.DL.PD.PURCHASE_END	Date	RW	Discount purchase end time.

Table 36–55 (Cont.) FCT_DiscountAnalysis EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
USAGE_START DETAIL.CUST_A.DL.PD.USAGE_START	Date	RW	Discount usage start time.
USAGE_END DETAIL.CUST_A.DL.PD.USAGE_END	Date	RW	Discount usage end time.
PRIORITY DETAIL.CUST_A.DL.PD.PRIORITY	Integer	RW	Discount priority.
MODE DETAIL.CUST_A.DL.PD.MODE	Integer	RW	Discount mode (parallel or cascading).
VALID_FLAG DETAIL.CUST_A.DL.PD.VALID_FLAG	Integer	RW	A value indicating discount validity.
TYPE DETAIL.CUST_A.DL.PD.TYPE	Integer	RW	Discount type (system, subscription, or item).
QUANTITY DETAIL.CUST_A.DL.PD.QUANTITY	Decimal	RW	Purchase quantity.
SCALE DETAIL.CUST_A.DL.PD.SCALE	Decimal	RW	Proration scale.
OFFERING_POID DETAIL.CUST_A.DL.PD.OFFERING_POID	String	RW	A value that identifies the purchased discount associated with the account.
DISCOUNT_MODEL DETAIL.CUST_A.DL.PD.DISCOUNT_MODEL	String	RW	A discount model.
SPONSOR_LIST DETAIL.CUST_A.SL	Block	R	The list of sponsors that split the charges with the event user.
SPONSOR_BALANCE_GROUP_ID DETAIL.CUST_A.SL.BALANCE_GROUP_ID	String	R	The balance group whose resources are used for the sponsorship list.
SPONSORSHIP_DETAILS DETAIL.CUST_A.SL.SD	Block	R	Sponsorships that belong to an account or service or are shared by the account or service.
SPONSORSHIP_ID DETAIL.CUST_A.SL.SD.SPONSORSHIP_ID	String	R	Sponsorship object ID.
SPONSOR_VALID_FLAG DETAIL.CUST_A.SL.SD.VALID_FLAG	Integer	RW	A value that indicates sponsorship validity.
SPONSOR_DISCOUNT_MODEL DETAIL.CUST_A.SL.SD.DISCOUNT_MODEL	String	RW	Sponsorship model.

FCT_DroppedCall

The FCT_DroppedCall module identifies phone calls that were terminated due to technical reasons and then resumed again through a customer's subsequent phone call. See "About Finding Dropped Calls and Continuation Calls" in *BRM Telco Integration*.

Dependencies

Run the FCT_DroppedCall module *after* the FCT_Account module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–56 lists the FCT_DroppedCall registry entries.

Table 36–56 FCT_DroppedCall Registry Entries

Entry	Description	Mandatory
Active	Specifies whether the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
FilePath	Specifies the path to the dropped calls data file. The default is the data directory (**./data**).	No
FileName	Specifies the name of the dropped calls data file, which stores back-up information about dropped calls. You use this file to restore information in case of system error or system restart.	Yes
TempPrefix	Specifies the prefix for the temporary dropped calls data files. The default is **tmp_**.	No
CheckField	Section that specifies the EDR field and values used to identify a dropped call. See "Specifying the EDR fields for finding dropped calls" in *BRM Telco Integration*.	Yes
CheckField.Name	Specifies the EDR field that is used to identify a dropped call. **Note:** Only one EDR field can be used to identify a dropped call.	Yes
CheckField.Value	Specifies the values for identifying a dropped call. **Note:** If more than one value qualifies an EDR as a dropped call, enter multiple values separated by a comma (,) with no spaces; for example: 5,6,7. BRM interprets the comma as a Boolean OR value.	Yes
WrittenFields	Section that specifies the dropped call EDR fields that are written into memory and are used to detect continuation calls. You use dummy key values such as 1 and 2 to list the EDR fields, as shown below: **1 =** *EDR_field* **2 =** *EDR_field* **3 =** *EDR_field* **Note:** BRM automatically writes the DETAIL.A_NUMBER, DETAIL.B_NUMBER, DETAIL.CHARGING_END_TIMESTAMP, and DETAIL.CUST_A.BILL_NEXT_DATE EDR fields to memory, so you should not list these fields. See "Specifying the EDR fields for identifying continuation calls" in *BRM Telco Integration*.	No

Table 36–56 (Cont.) FCT_DroppedCall Registry Entries

Entry	Description	Mandatory
AddedFields	Section that specifies the dropped call EDR fields that are written into memory and then added to the continuation call EDR. Important: When you map a dropped call EDR field to a continuation call EDR field, both fields must have the same data type. You can find a field's data type by reading the container description file (**container.dsc**). By default, the module does not enrich the continuation call EDR. See "Mapping dropped call fields to continuation call fields" in *BRM Telco Integration*.	No
AddedFields.Field*x*	Section that maps one dropped call EDR field to one continuation call EDR field. You create a **Field***x* section for each pair of EDR fields that you want to map. For example, if you want to map three EDR pairs, create a **Field1** section, a **Field2** section, and a **Field3** section.	No
AddedFields.Field*x***. ContinuationCallField**	Specifies the continuation call EDR field. The module adds the value of the dropped call EDR field specified in **DroppedCallField** to the continuation call EDR field that you specify.	No
AddedFields.Field*x***. DroppedCallField**	Specifies the dropped call EDR field to write into memory. This field's value is added to the continuation call EDR field specified in **ContinuationCallField**.	No

Sample Registry

```
DroppedCall
{
  ModuleName = FCT_DroppedCall
  Module
  {
    Active = TRUE
    FilePath = ./data
    FileName = dropped_call
    TempPrefix = tmp_
    CheckField
    {
      Name = DETAIL.CALL_COMPLETION_INDICATOR
      Value = 5
    }
    WrittenFields
    {
      1 = DETAIL.RECORD_TYPE
    }
    AddedFields
    {
      Field1
      {
        ContinuationCallField = DETAIL.DROPPED_CALL_QUANTITY
        DroppedCallField = DETAIL.DURATION
      }
    }
  }
}
```

With this sample registry configuration, the FCT_DroppedCall module identifies:

- *Dropped calls* by finding all EDRs with a DETAIL.CALL_COMPLETION_INDICATOR EDR field set to **5**.
- *Continuation calls* by using the DETAIL.RECORD_TYPE EDR field.

Note: By default, the module also automatically writes the DETAIL.A_NUMBER, DETAIL.B_NUMBER, DETAIL.CUST_A.BILL_NEXT_DATE, and DETAIL.CHARGING_END_TIMESTAMP EDR fields to memory.

When the module detects a continuation call, it adds the value of the dropped call's DETAIL.DURATION EDR field to the continuation call's DETAIL.DROPPED_CALL_QUANTITY EDR field.

Semaphore File Entries

Table 36–57 lists the FCT_DroppedCall Semaphore file entries.

Table 36–57 FCT_DroppedCall Semaphore File Entries

Entry	Description
Active	Specifies whether the module is active or inactive. **True** = Active **False** = Inactive
RemoveLimit	Specifies to remove all calls from the memory map and data file that are older than the specified number of days. For example, if you specify **7**, BRM removes from the memory map all entries older than 7 days. The time is calculated from the current system time. **Note:** Set the time to a large enough value to allow for late-arriving and recycled EDRs.

Sample Semaphore File Entries

```
ifw.Pipelines.ALL_RATE.Functions.FunctionPool.DroppedCall.Active = True
ifw.Pipelines.ALL_RATE.Functions.FunctionPool.DroppedCall.RemoveLimit = 7
```

EDR Container Fields

The FCT_DroppedCall module accesses the EDR container fields shown in Table 36–58. You can configure the module to access additional EDR container fields by using the module's **CheckField**, **WrittenFields**, and **AddedFields** registry entries.

Table 36–58 FCT_DroppedCall EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
A_NUMBER DETAIL.A_NUMBER	String	Read	Contains the customer A number.
B_NUMBER DETAIL.B_NUMBER	String	Read	Contains the customer B number.
CUST_A_BILL_NEXT_DATE DETAIL.CUST_A.BILL_NEXT_DATE	Date	Read	Contains the timestamp for the customer's next billing cycle.
CHARGING_END_TIMESTAMP DETAIL.CHARGING_END_TIMESTAMP	Date	Read	Contains the dropped call's ending timestamp.
DURATION DETAIL.DURATION	Decimal	Read	Contains the duration of the current call.

Table 36–58 (Cont.) FCT_DroppedCall EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
CUST_A_PROFILE DETAIL.CUST_A.ERA.PROFILE	String	Read	Contains the customer's account-related ERA data.
CUST_A_KEY DETAIL.CUST_A.ERA.PA.KEY	String	Read	Contains the key for the account-related ERA data.
CUST_A_VALUE DETAIL.CUST_A.ERA.PA.VALUE	String	Read	Contains the value for the account-related ERA data.
CALL_COMPLETION_INDICATOR DETAIL.CALL_COMPLETION_INDICATOR	String	Read	Contains the reason the current call session was terminated.
DROPPED_CALL_STATUS DETAIL.DROPPED_CALL_STATUS	Integer	Write	Flags the status of the call: **0** = Normal call **1** = Dropped call **2** = Continuation call **3** = Both a dropped call and a continuation call **4** = A call that was processed by FCT_DroppedCall but didn't qualify as a dropped call or a continuation call.
DROPPED_CALL_QUANTITY DETAIL.DROPPED_CALL_QUANTITY	Decimal	Write	When the EDR is flagged as a continuation call, this field contains the duration of the associated dropped call.

FCT_DuplicateCheck

The FCT_DuplicateCheck module checks for duplicate EDRs. See "Handling Duplicate EDRs".

Note: Before using the FCT_DuplicateCheck module, load the duplicate check stored procedures in the Pipeline Manager database. See "Loading procedures for FCT_DuplicateCheck" in *BRM Installation Guide*.

Dependencies

To enable your system to check for duplicate EDRs without using excessive disk space, connect the FCT_DuplicateCheck module to the Pipeline Manager database.

The FCT_DuplicateCheck module is typically the second module in a pipeline, directly following the FCT_PreSuspense module. This ensures that no further processing is done on duplicate EDRs.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–59 lists the FCT_DuplicateCheck registry entries.

Table 36–59 FCT_DuplicateCheck Registry Entries

Entry	Description	Mandatory
Active	Specifies whether the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
BufferLimit	Specifies the oldest date that previously processed EDRs can be stored in memory. The format is *YYYYMMDD*. Note: The **BufferLimit** date must be equal to or later than the **StoreLimit** date. For example, if the **StoreLimit** date is June 1, the **BufferLimit** must be June 1 or later. You can use this entry in a semaphore file. See "Setting Date Parameters for Storing Processed EDRs".	Yes
BulkInsertArraySize	Specifies the maximum number of rows for bulk insert when the data in memory flushes to the database. Default = **10000**.	No
DataConnection	Specifies a connection to the Pipeline Manager database. **Important:** To avoid using excessive disk space when checking for duplicate EDRs, enable this entry.Default = **False**. See "About Storing EDRs in a Database Instead of Files".	No
DuplicateIndicatorField	Specifies the EDR field to set if an EDR is a duplicate. The field specified must be an integer field. You can use any integer field in the EDR. This entry is used by the FCT_CiberOcc module to determine whether to create an OCC record. OCC records are not created for duplicate EDRs. See "About Settling Roaming Charges" in *BRM Configuring Roaming in Pipeline Manager*.	No
Fields	Specifies the EDR fields that are used for checking. **Important:** Do not use the DETAIL_CHARGING_START_TIMESTAMP field for duplicate checking. See "Specifying the Fields to Use for Duplicate Check".	Yes
FileName	Specifies the base file name of the data files (the transaction ID and suffix are appended). Default = **.** See "Managing FCT_DuplicateCheck Data Files".	Yes
IndexSpaceName	Index space name where the run-time duplicate check index is created (database mode only). For example: `IndexSpaceName = INTEGRATE_TS_1_IDX`	Yes, if using a database connection.
Path	Specifies the directory for the data files that store EDRs. See "Managing FCT_DuplicateCheck Data Files".	No
SearchKey	Identifies duplicate EDRs. See "Specifying a Search Key for Duplicate Check". **Important:** If you use the **SearchKey** registry entry, don't list the **SearchKey** value as a field in the **Fields** list.	No

Table 36–59 (Cont.) FCT_DuplicateCheck Registry Entries

Entry	Description	Mandatory
StoreLimit	Specifies the oldest date that previously processed EDRs can be stored. If an EDR is dated earlier than the **StoreLimit** date, the EDR is not processed by the FCT_DuplicateCheck module. The format is *YYYYMMDD*. **Note:** The **StoreLimit** date must be equal to or earlier than the **BufferLimit** date. For example, if the **StoreLimit** date is June 1, the **BufferLimit** must be June 1 or later.You can use this entry in a semaphore file. See "Setting Date Parameters for Storing Processed EDRs".	Yes
StreamName	Specifies the output stream for duplicate EDRs. See "Configuring Output for Rejected or Duplicate EDRs".	Yes
TableSpaceName	Table space name where the run-time duplicate check table is created (database mode only). When a **StoreLimit** or **BufferLimit** semaphore is sent, FCT_DuplicateCheck needs to know where to store data. This entry, and the **IndexSpaceName** entry specify the location in the database. For example: `TableSpaceName = INTEGRATE_TS_1_DAT`	Yes, if using a database connection.
TableSuffix	Allows you to create multiple IFW_DUPLICATECHECK tables when you run multiple pipelines.	No

Sample Registry

Note: When entering data in the **Fields** entry, use dummy key values such as **1**, **2**, and **3**, as shown in this example.

```
DuplicateCheck
{
   ModuleName  =  FCT_DuplicateCheck
   Module
   }
      Active  =  True
      DataConnection  =  ifw.DataPool.DupLogin1
      Path  =  ./data/dup
      Filename  =  call.duplicate
      StreamName  =  DuplicateOutput
      BufferLimit  =  20040105
      StoreLimit  =  20040101
      SearchKey  =  DETAIL.A_NUMBER
      Fields
      {
        1  =  DETAIL.BASIC_SERVICE
        2  =  DETAIL.B_NUMBER
      }
   }
}
```

With this sample registry configuration, the following occurs:

- EDRs dated January 5, 2004 (**BufferLimit**) or later are stored in the duplicate list in memory.

- EDRs dated January 1, 2004 (**StoreLimit**) through January 4, 2004 are stored in the IFW_DUPLICATECHECK database table.
- EDRs dated December 31, 2003 or earlier are ignored.

The following day, the module receives the following update registry:

```
DuplicateCheck
{
   ModuleName  =  FCT_DuplicateCheck
   Module
   {
      BufferLimit  =  20040106
      StoreLimit  =  20040102
   }
}
```

With this updated registry sample configuration, the following occurs:

- EDRs dated January 6, 2004 (new **BufferLimit**) or later are stored in the duplicate check list in memory.
- EDRs dated January 5, 2004 are moved to the IFW_DUPLICATECHECK database table.
- EDRs dated January 2, 2004 (new **StoreLimit**) through January 4, 2004 continue to be stored in the database table.

> **Note:** EDR data for duplicate checks is stored in the IFW_DUPLICATECHECK database table. This table can be hosted by any database. Normally, at the end of the day, all the EDR data in memory is flushed to the database.

Semaphore File Entries

Table 36–60 lists the FCT_DuplicateCheck Semaphore file entries.

Table 36–60 FCT_DuplicateCheck Semaphore File Entries

Entry	Description
Active	Specifies whether the module is active or inactive. **True** = Active **False** = Inactive
BufferLimit	Specifies the oldest date that previously processed EDRs can be stored in memory. The format is *YYYYMMDD*. **Note:** The **BufferLimit** date must be equal to or later than the **StoreLimit** date. For example, if the **StoreLimit** date is June 1, the **BufferLimit** must be June 1 or later. See "Setting Date Parameters for Storing Processed EDRs".
StoreLimit	Specifies the oldest date that previously processed EDRs can be stored. If an EDR is dated earlier than the **StoreLimit** date, the EDR is not processed by the FCT_DuplicateCheck module. The format is *YYYYMMDD*. **Note:** The **BufferLimit** date must be equal to or later than the **StoreLimit** date. For example, if the **StoreLimit** date is June 1, the **BufferLimit** must be June 1 or later. See "Setting Date Parameters for Storing Processed EDRs".

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.DuplicateCheck.
Module.StoreLimit = 20020101

ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.DuplicateCheck.
Module.BufferLimit = 20020125
```

Sample Output Configuration

You configure the output stream for the FCT_DuplicateCheck module in the Output section of the registry, for example:

```
# Output stream for duplicate events
DuplicateOutput
{
 ModuleName = OUT_Reject
 Module
 {
  OutputStream
  {
   ModuleName = EXT_OutFileManager
   Module
   {
    OutputPath = ./samples/wireless/data/rej
    OutputPrefix = test
    OutputSuffix = .dup
    TempPrefix   = tmp

    TempDataPath    = ./samples/wireless/data/rej
    TempDataPrefix  = dup.tmp.
    TempDataSuffix  = .data

    Replace = TRUE
    DeleteEmptyFile = TRUE
   }
  }
 }
} # end of DuplicateOutput
```

Note: To ensure output file integrity, specify a unique combination of OutputPath, OutputSuffix, and OutputPrefix values for each output stream defined in the registry.

EDR Container Fields

You specify the EDR container fields in the FCT_DuplicateCheck module **Fields** startup registry entry.

Database Tables

The FCT_DuplicateCheck module uses the IFW_DUPLICATECHECK database table.

If you use the database instead of a file to define the data that the FCT_DuplicateCheck module uses for comparing EDRs, you need to create this table. The FCT_DuplicateCheck module uses the data in the IFW_DUPLICATECHECK table to check for duplicate EDRs. See "Handling Duplicate EDRs".

The IFW_DUPLICATECHECK table should have a unique index. A duplicate EDR is detected by the database reporting a violation of the uniqueness when attempting to INSERT.

> **Note:** Oracle recommends that you have multiple partitions to increase the INSERT performance.

For information about the fields in database tables, see the documentation in *Pipeline_Home***database**.

FCT_EnhancedSplitting

The FCT_EnhancedSplitting module specifies different output streams for EDRs based on rules. For example:

- You can split EDRs for different service types to different output streams.
- You can split EDRs from roaming outcollects and incollects into different streams.

See "Using Rules to Send EDRs to Different Output Streams".

Dependencies

Requires a connection to the Pipeline Manager database.

Because you can split EDRs based on service codes, this module should run after the FCT_ServiceCodeMap and FCT_UsageClassMap modules.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–61 lists the FCT_EnhancedSplitting registry entries.

Table 36–61 FCT_EnhancedSplitting Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
DefaultOutput	Specifies the default output stream if no splitting rule matches the current EDR. If no default output stream is specified, the EDR receives the error message ERR_NO_SPLITTING_PERFORMED. Default = **.** See "Configuring EDR Output Processing".	No
SystemBrands	Maps the system brand table to the output stream. Each entry in this section has the format SYSBRAND=OUTPUT-STREAM. See "Using Rules to Send EDRs to Different Output Streams".	Yes

Sample Registry

```
Splitting
{
   ModuleName = FCT_EnhancedSplitting
   Module
      {
      Active = True
      DataConnection = Login
      DefaultOutput = EdrOutput0
      SystemBrands
         {
         1 = EdrOutput1
         2 = EdrOutput2
      }
   }
}
```

Semaphore File Entries

Table 36–62 lists the FCT_EnhancedSplitting Semaphore file entries.

Table 36–62 FCT_EnhancedSplitting Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
Reload	Reloads data from the database. See "Reloading data into a Pipeline Manager module" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry

```
ifw.Pipelines.PRE_PRODCESS.Functions.PreProcessing.FunctionPool.
EnhancedSplitting.Module.Active = True
```

EDR Container Fields

Table 36–63 lists the FCT_EnhancedSplitting EDR container fields.

Table 36–63 FCT_EnhancedSplitting EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
RECORD_TYPE DETAIL.RECORD_TYPE	String	Read	Contains the record type.
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code.
INTERN_USAGE_CLASS DETAIL.INTERN_USAGE_CLASS	String	Read	Contains the internal usage class.
SOURCE_NETWORK DETAIL.SOURCE_NETWORK	String	Read	Contains the source network code. This could either be the PLMN ID or any logical operator code.

Table 36–63 (Cont.) FCT_EnhancedSplitting EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
DESTINATION_NETWORK DETAIL.DESTINATION_NETWORK	String	Read	Contains the destination network code.
ASS_GSMW_EXT.ORIGINATING_SWITCH_IDENTIFICATION DETAIL.ASS_GSMW_EXT.ORIGINATING_SWITCH_IDENTIFICATION	String	Read	Contains the GSM MSC or Switch ID handling the origin of the call.
ASS_GPRS_EXT.ORIGINATING_SWITCH_IDENTIFICATION DETAIL.ASS_GPRS_EXT.ORIGINATING_SWITCH_IDENTIFICATION	String	Read	Contains the GPRS MSC or Switch ID handling the origin of the call.
ASS_GSMW_EXT.TRUNK_INPUT DETAIL.ASS_GSMW_EXT.TRUNK_INPUT	String	Read	Contains the trunk identification (in-route address in network switches).
ASS_GSMW_EXT_TRUNK_OUTPUT DETAIL.ASS_GSMW_EXT.TRUNK_OUTPUT	String	Read	Contains the trunk identification (out-route address in network switches).
A_NUMBER DETAIL.A_NUMBER	String	Read	Contains the A number.
B_NUMBER DETAIL.B_NUMBER	String	Read	Contains the B number.
INTERN_C_NUMBER_ZONE DETAIL.INTERN_C_NUMBER_ZONE	String	Read	Contains the number where the call was first terminated if it was forwarded or routed.
CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the time-stamp used for start of charging.

Database Tables

The FCT_EnhancedSplitting module uses the following database tables:

- IFW_SPLITTING_TYPE. The FCT_EnhancedSplitting module uses the data in the IFW_SPLITTING_TYPE table to determine how to route EDRs to different output streams based on rules. See "Using Rules to Send EDRs to Different Output Streams".
- IFW_SYSTEM_BRAND. The IFW_SYSTEM_BRAND table stores the system brand codes. See "Using Rules to Send EDRs to Different Output Streams".

For information about the fields in database tables, see the documentation in *Pipeline_Home***database**.

Note: For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

FCT_EventOrder

When an event is rated, the FCT_EventOrder module uses the criteria and the event timestamps to determine if the event needs to be rerated. For more information, see "About Automatic Rerating of Out-Of-Order Events" in *BRM Setting Up Pricing and Rating*.

Dependencies

This module must run *after* the FCT_MainRating and FCT_Discount modules and *before* the FCT_Reject module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–64 lists the FCT_EventOrder registry entries.

Table 36–64 FCT_EventOrder Registry Entries

Entry	Description	Mandatory
Active	Specifies whether the module is active or inactive: **True** = Active **False** = Inactive	Yes
AccountDataModule	Specifies the connection to the DAT_AccountBatch module.	Yes
PortalConfigDataModule	Specifies the connection to the DAT_PortalConfig module.	Yes
RerateDelayTolerance	Specifies a time in minutes that controls how much out-of-order EDR data FCT_EventOrder writes to a rerate-request file before creating a new file. For more information, see "About automatic rerating of out-of-order events" in *BRM Setting Up Pricing and Rating*.	No
NumberOfAccountLimit	Specifies the number of accounts FCT_EventOrder assigns to each rerate job. This entry affects batch rerating throughput. For more information, see "Configuring event notification for rerating backdated events" in *BRM Setting Up Pricing and Rating*.	No
OutputDirectory	Specifies the output directory for out-of-order events. **Important:** You must create this directory. It is not created by BRM installation scripts.	Yes
OutputPrefix	Specifies the prefix of the rerate-request file name. The default is **ood**.	Yes
SkipPrevBillingCycle	Specifies whether to skip events that belong to previous billing cycles. **True** = Skip the events **False** = Do not skip the events (Default)	No

Sample Registry

```
EventOrder
{
  ModuleName                = FCT_EventOrder
  Module
  {
    Active                  = True
    AccountDataModule       = ifw.DataPool.CustomerData
```

```
    PortalConfigDataModule   = ifw.DataPool.PortalConfigDataModule

#delay tolerance in minutes for creating rerate jobs
    RerateDelayTolerance     =  180

#maximum number of accounts in the rerate-request file
#this should match the value of the pin_rerate "per_job"
#configuration entry
    NumberOfAccountLimit     = 1000

#output directory and prefix of the rerate-request file
    OutputDirectory          = Pipeline_Home/data/out/ood
    OutputPrefix             = ood_

  }
}
```

FCT_ExchangeRate

The FCT_ExchangeRate module converts the currency in the charge packets, discount packets, and tax packets to the home (system) currency or the customer's billing currency.

See "Defining Currency Exchange Rates" in *BRM Setting Up Pricing and Rating*.

Dependencies

Requires a connection to the DAT_ExchangeRate module and the DAT_Currency module.

This module must run *after* the FCT_MainRating module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–65 lists the FCT_ExchangeRate registry entries.

Table 36–65 FCT_ExchangeRate Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
CurrencyDataModule	Specifies the connection to the DAT_Currency module. When currency module is specified, the resourceid field of the charge packet is updated to the new currency. The resourceid is needed only when loading the records into the BRM database by using Rated Event Loader. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	No
ErrorMessages	Specifies whether error messages should be appended to the EDR container or should be suppressed.	Yes

Table 36–65 (Cont.) FCT_ExchangeRate Registry Entries

Entry	Description	Mandatory
ExchangeRateDataModule	Specifies the connection to the DAT_ExchangeRate module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes
HomeCurrency	Specifies the local currency used by your company. The code must use three characters; for example, USD or DEM. You must set this value when the **RatingDateHome** registry value is set. This entry is used only if the **RatingDateHome** entry is used.	No
Mode	Specifies how to apply the exchange rate. Values are: ■ Normal - Converts the From Currency to the To Currency by multiplying the From Currency amount and exchange rate. ■ Reverse - Converts the To Currency to the From Currency by multiplying the To Currency amount and 1/exchange rate. Default mode is **Normal**. See "Defining Currency Exchange Rates" in *BRM Setting Up Pricing and Rating*.	No
RatingDateBilling	Specifies how to determine the validity date for the conversion from the rating currency to the billing currency. Values are: ■ SYSTEM ■ FILE ■ CDR ■ NONE Default = **NONE** **Note:** FCT_ExchangeRate module converts currency to home currency or billing currency. If you specify RatingDateBilling in addition to RatingDateHome and HomeCurrency parameters, it converts the currency to the home currency only.	No
RatingDateHome	Specifies how to determine the validity date for the conversion from the rating currency to the home currency. Values are: ■ SYSTEM ■ FILE ■ CDR ■ NONE Default = **NONE**	No

Sample Registry

```
ExchangRate
{
  ModuleName = FCT_ExchangeRate
  Module
  {
    Active = True
```

```
    ExchangeRateDataModule = ExchangeRateData
    RatingDateBilling  =  SYSTEM
    RatingDateHome = CDR
    HomeCurrency = DEM
    ErrorMessages = False
  }
}
```

Semaphore File Entries

Table 36–66 lists the FCT_ExchangeRate Semaphore file entries.

Table 36–66 FCT_ExchangeRate Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ExchangeRate.Module.Active = False
```

EDR Container Fields

Table 36–67 lists the FCT_ExchangeRate EDR container fields.

Table 36–67 FCT_ExchangeRate EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
CREATION_TIMESTAMP HEADER.CREATION_TIMESTAMP	Date	Read	Contains the EDR creation time stamp. This field is used if the exchange rate time (Home or Billing) is set to the file date.
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging time stamp. This field is used if the exchange rate time (Home or Billing) is set to the CDR date.
ASS_CBD DETAIL.ASS_CBD	Block-Index	Read	Block used for iteration over all associated charge breakdown records.
ASS_CBD_CHARGE_PACKET DETAIL.ASS_CBD.CP	Block-Index	Read	Block used for iteration over all charge packages.

Table 36–67 (Cont.) FCT_ExchangeRate EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_CBD_RECORD_TYPE DETAIL.ASS_CBD.RECORD_TYPE	Integer	Read	Contains the record type. The billing and/or home currency is calculated for these record types: ■ 981 ■ 982 ■ 983 ■ 984 ■ 990 ■ 991 For record type 980, only the home currency is calculated.
CHARGE_CURRENCY_TYPE DETAIL.ASS_CBD.CP.CHARGED_CURRENCY_TYPE	String	Read/Write	Contains the currency type. The type **R** is read and the types **B** and **H** are calculated if specified.
CHARGED_AMOUNT_VALUE DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE	Integer	Read/Write	Contains the charge amount that needs to be converted.
CHARGED_AMOUNT_CURRENCY DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_CURRENCY	String	Read/Write	Contains the amount in the charge currency for calculation.
INTERN_BILLING_CURRENCY DETAIL.ASS_CBD.CP.INTERN_BILLING_CURRENCY	String	Read	Contains the billing currency which is calculated and added with a charge package.
INTERN_HOME_CURRENCY DETAIL.ASS_CBD.CP.INTERN_HOME_CURRENCY	String	Read	Contains the home currency which is calculated and added with an charge package. If no home currency is found the home currency from the registry is used for calculation.
EXCHANGERATE DETAIL.ASS_CBD.CP.EXCHANGERATE	String	Write	Contains the exchange rate recommended for TAP.
EXCHANGECURRENCY DETAIL.ASS_CBD.CP.EXCHANGECURRENCY	String	Read/Write	Contains the exchange currency recommended for TAP.
CHARGED_AMOUNT_VALUE_ORIG DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE_ORIG	Decimal	Write	Contains the charge amount in rating currency.
RESOURCE DETAIL.ASS_CBD.CP.RESOURCE	String	Write	Contains the resource name

Table 36–67 (Cont.) FCT_ExchangeRate EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
RESOURCE_ID DETAIL.ASS_CBD.CP.RESOURCE_ID	Integer	Read/Write	Contains the resource ID.
RESOURCE_ID_ORIG DETAIL.ASS_CBD.CP.RESOURCE_ID_ORIG	Integer	Read	Contains the rating resource ID.
DISCOUNT_PACKET DETAIL.ASS_CBD.DP	Block	Read	Contains the discount related information for the event.
DISCOUNT_PACKET_GRANTED_AMOUNT DETAIL.ASS_CBD.DP.GRANTED_AMOUNT	Decimal	Read/Write	Contains the discounted amount.
DISCOUNT_PACKET_GRANTED_AMOUNT_ORIG DETAIL.ASS_CBD.DP.GRANTED_AMOUNT_ORIG	Decimal	Write	Contains the discounted amount in the initial rated currency.
DISCOUNT_PACKET_INTERN_SRC_PACKET_INDEX DETAIL.ASS_CBD.DP.INTERN_SRC_PACKET_INDEX	Integer	Read	Contains the charge packet number for which this discount is given.
DISCOUNT_PACKET_RESOURCE_ID DETAIL.ASS_CBD.DP.RESOURCE_ID	Integer	Read/Write	Contains the resource ID.
DISCOUNT_PACKET_RESOURCE_ID_ORIG DETAIL.ASS_CBD.DP.RESOURCE_ID_ORIG	Integer	Write	Contains the currency for which discount has been given originally.
ASS_CBD_RECORD_NUMBER DETAIL.ASS_CBD.RECORD_NUMBER	Integer	Read	Contains the block creation number. These numbers (980, 981, 982, 983, 984, 990, 991) are only considered for exchange rate. Any other number of ASS_CBD will be ignored.
ASS_CBD_CHARGE_PACKET_TAX_PACKET DETAIL.ASS_CBD.TP	Block	Read	Contains tax related information for the event.
CHARGED_INFO_ID DETAIL.ASS_CBD.TP.RELATED_CHARGE_INFO_ID	Integer	Read	Contains charge packet index corresponding to the tax.
TP_TAX_VALUE DETAIL.ASS_CBD.TP.TAX_VALUE	Decimal	Read/Write	Contains the tax amount
TP_TAX_VALUE_ORIG DETAIL.ASS_CBD.TP.TAX_VALUE_ORIG	Decimal	Write	Contains the tax amount in the originally rated currency.
CUST_A_CURRENCY DETAIL.CUST_A.CURRENCY	String	Read	Contains the billing currency of the customer.

FCT_Filter_Set

The FCT_Filter_Set module determines whether an event data record (EDR) is eligible for the system products and system discounts contained in a filter set. If so, it adds those system products and discounts to a customer's list of purchased products.

For information about filter sets, see the following:

- About Using Filter Sets to Apply System Products and Discounts
- "Creating filter sets" in *BRM Setting Up Pricing and Rating*

Dependencies

This module must run *after* the FCT_Account module.

FCT_Filter_Set requires the following connections:

- BRM database
- DAT_Discount module
- DAT_AccountBatch module

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–68 lists the FCT_Filter_Set registry entries.

Table 36–68 FCT_Filter_Set Registry Entries

Entry	Description	Mandatory
AccountDataModule	Specifies the connection to the DAT_AccountBatch module. See "DAT_AccountBatch" and "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes
Active	Specifies if the module is active or inactive. ■ **True** = Active ■ **False** = Inactive **Note:** When the module is active, it takes over the function of applying system discounts from the FCT_DiscountAnalysis module.	Yes
DiscountDataModule	Specifies the connection to the DAT_Discount module. See "DAT_Discount" and "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes
InfranetConnection	Specifies the connection to the BRM database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes

Sample Registry

```
#------------------------
# Segment FCT
#------------------------
Segment
{
    ModuleName = FCT_Filter_Set
    Module
    {
```

```
            Active = True
            DiscountDataModule = ifw.DataPool.DiscountModelDataModule
            AccountDataModule = ifw.DataPool.CustomerData
            InfranetConnection = ifw.DataPool.LoginInfranet
        }
    }
```

Semaphore File Entries

Table 36–69 lists the FCT_Filter_Set Semaphore file entries.

Table 36–69 FCT_Filter_Set Semaphore File Entry

Entry	Description
Reload	Reloads data from the database into FCT_Filter_Set. See "Reloading data into a Pipeline Manager module" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.Segment.Module.Reload{}
```

EDR Container Fields

This module can read any valid EDR fields (defined in **container.dsc**) for matching criteria (EDR field name and value).

> **Important:** EDR field names read by this module cannot exceed a depth of 6 levels. Exceeding this limit prevents the pipeline from starting and results in an error message.

FCT_FirstUsageNotify

The FCT_FirstUsageNotify module sets the batch rating output stream for products and discounts that start on first usage and sets an error code in the EDR for suspending events that use those products and discounts while their validity periods are being set.

For more information, see "About Suspending EDRs for Products and Discounts that Start on First Usage".

> **Note:** To process first-usage events in the real-time rerating pipeline, use the "ISC_FirstProductRealtime" iScript.

Dependencies

Run this module before the FCT_ApplyBalance and FCT_Reject modules.

Requires a connection to DAT_AccountBatch and FCT_Reject.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–70 lists the FCT_FirstUsageNotify registry entries.

Table 36–70 FCT_FirstUsageNotify Registry Entries

Entry	Description	Mandatory
Active	Specifies whether the module is active or inactive. ■ **True** = Active ■ **False** = Inactive You can use this entry in a semaphore file.	Yes
CustomerDataModule	Specifies a connection to the DAT_AccountBatch module.	Yes
FirstUsageNotifyOutput	Specifies the output stream for the list of products and discounts used that are set to start on first usage. Default = **FirstUsageNotifyOutput**	Yes
RejectModule	Specifies a connection to the FCT_Reject module. FCT_FirstUsageNotify uses FCT_Reject to determine whether an EDR will be rejected for reasons other than setting the validity. FCT_FirstUsageNotify does not set validity if the EDR will be rejected.	Yes

Sample Registry

```
FirstUsageNotify
{
    ModuleName = FCT_FirstUsageNotify
    Module
    {
        Active = True
        CustomerDataModule = ifw.DataPool.Account
        RejectModule = ifw.Pipelines.MyPipeline1.Reject
        FirstUsageNotifyOutput = FirstUsageNotifyOutput
    }
}
```

Semaphore File Entries

Table 36–71 lists the FCT_FirstUsageNotify Semaphore file entry.

Table 36–71 FCT_FirstUsageNotify Semaphore File Entry

Entry	Description
Active	Specifies whether the module is active or inactive. ■ **True** = Active ■ **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.FirstUsageNotify.Module.Active = False
```

EDR Container Fields

Table 36–72 lists the FCT_FirstUsageNotify EDR container fields.

Table 36–72 FCT_FirstUsageNotify EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
TRANSACTION_ID INTERNAL.TRANSACTION_ID	String	Read	Specifies the transaction ID.
CUST_A_INTERN_PP_INDEX DETAIL.CUST_A.INTERN_FOUND_PP_INDEX	String	Read	Contains an index of the customer's purchased products that were used for rating.
CUST_A_ACCOUNT_PARENT_ID DETAIL.CUST_A.ACCOUNT_PARENT_ID	String	Read	Specifies the customer account POID.
CUST_A_PRODUCT_ID DETAIL.CUST_A.PRODUCT.OFFERING_POID	String	Read	Specifies the POID of the account's product used to rate the event.
CUST_A_PRODUCT_FIRST_USAGE_INDICATOR DETAIL.CUST_A.PRODUCT.FIRST_USAGE_INDICATOR	String	Read	Specifies whether the product is configured to start when first used and the first-usage validity period status.
FU_DISCOUNT_OBJECTS DETAIL.ASS_CBD.FU_DISCOUNT_OBJECTS	String	Read	Specifies the account's first-usage discounts that were applied to the event.
RECYCLE_KEY DETAIL.ASS_SUSPENSE_EXT.RECYCLE_KEY	String	Write	Specifies the recycle key. If the product or discount starts on first usage and its validity period is not set, this field is set to **FirstUsageValidity**.
UTC_TIME_OFFSET DETAIL.UTC_TIME_OFFSET	String	Read	Specifies the difference between local time and UTC time.
ASSOCIATED_INFRANET_BILLING DETAIL.ASS_PIN	String	Write	The Associated BRM Billing record.
FIRST_USAGE_SET_VALIDITY DETAIL.ASS_PIN.FIRST_USAGE	String	Write	The First-usage data block.
FIRST_USAGE_ACCOUNT_POID DETAIL.ASS_PIN.FIRST_USAGE.ACCOUNT_POID_STR	String	Write	Specifies the customer account POID.
FIRST_USAGE_OFFERING_POID DETAIL.ASS_PIN.FIRST_USAGE.OFFERING_POID_STR	String	Write	Specifies the POID of the account's product or discount used to rate or discount the event.
FIRST_USAGE_START_TIME DETAIL.ASS_PIN.FIRST_USAGE.START_T	String	Write	Specifies the validity period start time, which is based on the event start time.
FU_UTC_TIME_OFFSET DETAIL.ASS_PIN.FIRST_USAGE.UTC_TIME_OFFSET	String	Read	Specifies the difference between first-usage start time and UTC time.

Table 36–72 (Cont.) FCT_FirstUsageNotify EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	String	Read	Specifies the EDR start timestamp.
CUST_A_PRODUCT_SERVICE_ID DETAIL.CUST_A.PRODUCT.SERVICE_ID	String	Read	Specifies the POID of the service object
CUST_A_PRODUCT_SERVICE_TYPE DETAIL.CUST_A.PRODUCT.SERVICE_TYPE	String	Read	Specifies the POID type of the service object
CUST_A_PRODUCT_SERVICE_TYPE DETAIL.ASS_PIN.FIRST_USAGE.SERVICE_POID_STR	String	Read	Specifies the POID of the service object which has FU product uninitialized

FCT_GlobalRating

The FCT_GlobalRating module rates all EDRs with a default set of rate plans. See "About Global Rating".

Dependencies

This module must run after FCT_Account.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–73 lists the FCT_GlobalRating registry entries.

Table 36–73 FCT_GlobalRating Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
EdrRatePlans	Specifies a set of rate plans that are used for rating. You can use this entry in a semaphore file.	Yes

Sample Registry

```
GlobalRating
{
  ModuleName = FCT_GlobalRating
  Module
  {
    Active = True
    EdrRatePlans
    {
      RatePlan_1
      RatePlan_2
      RatePlan_3
    }
```

```
    }
}
```

Semaphore File Entries

Table 36–74 lists the FCT_GlobalRating Semaphore file entries.

Table 36–74 FCT_GlobalRating Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
EdrRatePlans	Specifies a set of rate plans that are used for rating.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.GlobalRating.Module.Active = True
```

EDR Container Fields

Table 36–75 lists the FCT_GlobalRating EDR container fields.

Table 36–75 FCT_GlobalRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_CBD DETAIL.ASS_CBD	Block	Create	Block to hold the rating data.
ASS_CBD_CHARGE_PACKET DETAIL.ASS_CBD.CP	Block	Create	Block created for each **EdrRatePlans** entry.
CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the EDR start time stamp. This value is used in the charge packet.
INTERN_A_NUMBER_ZONE DETAIL.INTERN_A_NUMBER_ZONE	String	Read	Contains the A number for zoning. This value is used in the charge packet.
INTERN_B_NUMBER_ZONE DETAIL.INTERN_B_NUMBER_ZONE	String	Read	Contains the B number for zoning. This value is used in the charge packet.
ASS_CBD_RECORD_TYPE DETAIL.ASS_CBD.RECORD_TYPE	String	Write	Contains the record type. This value is always set to **980**.
CHARGE_TYPE DETAIL.ASS_CBD.CP.CHARGE_TYPE	String	Write	Contains the charge type. This value is always set to **N**.
ASS_CBD_CHARGING_START_TIMESTAMP DETAIL.ASS_CBD.CP.CHARGING_START_TIMESTAMP	Date	Write	Contains the charging start timestamp from the DETAIL field.

Table 36–75 (Cont.) FCT_GlobalRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
RATEPLAN_CODE DETAIL.ASS_CBD.CP.RATEPLAN_CODE	String	Write	Contains the rate plan code that is used for zoning and rating.
INTERN_ORIGIN_NUM_ZONE DETAIL.ASS_CBD.CP.INTERN_ORIGIN_NUM_ZONE	String	Write	Contains the A number zoning information used by the FCT_ PreRating module. See "FCT_PreRating".
INTERN_DESTIN_NUM_ZONE DETAIL.ASS_CBD.CP.INTERN_DESTIN_NUM_ZONE	String	Write	Contains the B number zoning information used by the FCT_ PreRating module. See "FCT_PreRating".

FCT_IRules

The FCT_IRules module evaluates iRules. You use iRules to perform functions such as mapping EDR data fields and splitting EDR containers to different output streams. You group rules together in a rule set.

You can store rules in the database or in an ASCII file.

> **Important:** FCT_IRules cannot read files written in XML format. You can, however, use the **irules2db.pl** script to load iRules written in XML into the database. See "Importing and Exporting Validation Rules" in *BRM Developer's Guide*.

For more information, see:

- "About configuring iRules" in *BRM System Administrator's Guide*
- "Creating iScripts and iRules" in *BRM Developer's Guide*

Dependencies

If the data is read from the database, this module requires a connection to the Pipeline Manager database.

This module can run anywhere, depending on the data that is being processed.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–76 lists the FCT_IRules registry entries.

Table 36–76 FCT_IRules Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database if rules are stored in the database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
Descriptions	Specifies the rule set descriptions. The rule sets are evaluated in the specified order. You can use this entry in a semaphore file.	No
Rules	Specifies the rule sets that the module evaluates. You can use this entry in a semaphore file.	Yes, when using the database interface. No, when using a file interface.
Source	Specifies the source of the rules: ■ **File** ■ **Database**	Yes

Sample Registry Entry for the Database Interface

```
IRules
{
    ModuleName = FCT_IRules
    Module
    {
        Active = TRUE
        Source = Database
        DataConnection = ifw.DataPool.DataConnection
        Rules
        {
            CIBER_VAL
        }
    }
}
```

Sample Registry Entry for the File Interface

```
Router
{
    ModuleName = FCT_IRules
    Module
    {
        Active = TRUE
        Source = File
        Rules
        {
        }
        Descriptions
        {
            ServiceRequestRouter = ./iScriptLib/AAA/IRL_Router.irl
        }
```

```
    }
}
```

Semaphore File Entries

Table 36–77 lists the FCT_IRules Semaphore file entries.

Table 36–77 FCT_IRules Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
Descriptions	Specifies the rule set descriptions. The rule sets are evaluated in the specified order.
Reload	Reloads rules from the database or an ASCII file.
Rules	Specifies the rule sets that the module evaluates.

Sample Semaphore File Entry

```
ifw.Pipelines.PRE_RECYCLE.Functions.PreProcessing.FunctionPool.PipelineSplit.Module.Reload {}
```

EDR Container Fields

The EDR container fields that are accessed by FCT_IRule are those that you specify in the rules.

Database Interface

FCT_IRules uses the following database tables to store the generic rules:

- **IFW_RULESET**. Specifies a rule set for linking a related set of rules.
- **IFW_RULESETLIST**. Links a rule set with its rules. Each rule has a rank that specifies the order in which it is evaluated.
- **IFW_RULE**. Stores the iRules.
- **IFW_RULEITEM**. Contains the conditions, the result and the rank for a rule item.

For information about the fields in database tables, see the documentation in *Pipeline_Home***database**.

> **Note:** For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

File Interface

You can store iRules in an ASCII file by using a syntax that is similar to XML:

- Each tag must start on a separate line.
- Blank lines are allowed.
- Comment lines must start with a pound symbol: **#**

The rules and rule items are ranked by their order in the file; the first having the highest rank.

Loading Rule Sets from the Database

You can load rule sets from the Pipeline Manager database. iRule components are stored in the following tables:

- **IFW_RULESET**
- **IFW_RULESETLIST**
- **IFW_RULE**
- **IFW_RULEITEM**

Loading Rule Sets from an ASCII File

The following example file shows the syntax of a rule set:

```
################################################################################
# Example ruleset file
################################################################################
<RULESET>
#===============================================================================
# The first rule of this ruleset
#===============================================================================
<RULE MapRule1>
<INIT_SCRIPT>
String code = edrString(DETAIL.SERVICE_CODE) + "_GK2";
</INIT_SCRIPT>
#-------------------------------------------------------------------------------
# The first ruleitem of this rule
#-------------------------------------------------------------------------------
<RULEITEM discount>
<CONDITION>
/* The text between <CONDITION> and </CONDITON> is directly passed to the
* the interpreter. Lines with # are no comments here!
*/
edrString(DETAIL.RECORD_TYPE ) =~ "02*"; // this is a pattern compare rule
edrLong(DETAIL.QUANTITY) > 30; // this is a normal condition
</CONDITION>
<RESULT>
/* This iScript is executed if the conditions specified in the <CONDITION>
* tag before are matching the current edr container */
edrDecimal(DETAIL.CHARGED_AMOUNT) = 2.50; // set the amount to DM 2.50
</RESULT>
</RULEITEM>
#-------------------------------------------------------------------------------
# Here can be specified some more RULEITEMS
#-------------------------------------------------------------------------------
</RULE>
#===============================================================================
# The second rule of this ruleset
#===============================================================================
<RULE MapRule2>
#-------------------------------------------------------------------------------
# Put the ruleitems for MapRule2 here
#-------------------------------------------------------------------------------
</RULE>
</RULESET>
```

FCT_IScript

The FCT_IScript module runs iScripts. The scripts are run in the order specified in the registry.

See:

- "About Configuring iScripts" in *BRM System Administrator's Guide*
- "Creating iScripts and iRules" in *BRM Developer's Guide*

Dependencies

If the iScripts are stored in the database, this module requires a connection to the Pipeline Manager database.

This module can run anywhere, depending on the data that is being processed.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–78 lists the FCT_IScript registry entries.

Table 36–78 FCT_IScript Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies a connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes, if the module gets data from the database.
Scripts	Specifies the scripts to run. ■ If the scripts are stored in the database, each value in the entry specifies the SCRIPTCODE field in the INT_ISCRIPT database table. ■ If the scripts are stored in a file, each section in the file must have a registry entry FileName = *file*. **Note:** The section for each script can store a number of entries that are passed as global constants to the interpreter.	Yes
Scripts.*ScriptName*	The name of the script as used in the registry.	N/A
Scripts.*ScriptName*.**FileName**	The name and path of the script.	N/A
Scripts.*ScriptName.Registry_Parameter*	Registry parameters used in the iScript. The example below uses the GL_Code parameter: GL_CODE = 1514	N/A
Source	Specifies the source of the iScripts. ■ **File** ■ **Database**	Yes

Sample Registry for the File Interface

```
ConsolidatedCP
```

```
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = File
      Scripts
      {
         ConsolidatedCPIScript
         {
            FileName = ./iScriptLib/iScriptLib_Roaming/ISC_ConsolidatedCP.isc
            GL_CODE = 1514
         }
      }
   }
}
```

Sample Registry for the Database Interface

```
IScript
{
  ModuleName = FCT_IScript
  Module
  {
    Active = True
    Source = Database
    DataConnection = ifw.DataPool.Login
    Scripts
    {
      Mapping
      {
        # can be used as reg.Arg1 in the IScript
        Arg1 = any_argument_value
        }
        Specials
        {
      }
    }
  }
}
```

Semaphore File Entries

Table 36–79 lists the FCT_IScript Semaphore file entries.

Table 36–79 FCT_IScript Semaphore File Entries

Entry	Description
Active	Activates or deactivates the module. **TRUE** = Activate. **FALSE** = Deactivate.
Reload	Reloads iScripts from the database or an ASCII file.
Scripts	Section with scripts to execute. In case of Source = DATABASE each value in the section specifies the SCRIPTCODE of a script in database table INT_ISCRIPT. In case of Source = FILE each section within the section must have a registry entry FileName = *file*.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Standard.FunctionPool.IScript.Module.Reload {}
```

Database Tables

The FCT_IScript module uses the IFW_ISCRIPT database table to store iScripts.

For information about the fields in database tables, see the documentation in *Pipeline_Home***database**.

File Interface

The iScript source code is stored in a simple ASCII file. The file is loaded and compiled at startup. See "Creating iScripts and iRules" in *BRM Developer's Guide*.

FCT_ItemAssign

The FCT_ItemAssign module assigns bill items to events.

See:

- "Setting Up Batch Rating To Assign Items Based On Event Attributes" in *BRM Configuring and Running Billing*
- "Creating custom bill items" in *BRM Configuring and Running Billing*

Dependencies

FCT_ItemAssign requires a connection to the DAT_ItemAssign module.

Must run after the FCT_Account, rating, and discounting modules and before the FCT_BillingRecord module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–80 lists the FCT_ItemAssign registry entries.

Table 36–80 FCT_ItemAssign Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive	Yes
DataModule	Specifies the connection to the DAT_ItemAssign module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes

Sample Registry

```
Item Assignment
{
  ModuleName = FCT_ItemAssign
  Module
  {
    Active = True
```

```
    DataModule = ifw.DataPool.ItemAssignDataModule
  }
}
```

EDR Container Fields

The FCT_ItemAssign module uses the EDR container fields listed in Table 36–80:

Table 36–81 FCT_ItemAssign EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
DETAIL.ITEM_TAG	String	Read	Contains the item tag.
DETAIL.CUST_A. PRODUCT.SERVICE_USED_ITEM_POID	String	Write	Contains the item POID for the event.
DETAIL.CUST_A.INTERN_FOUND_PP_INDEX	Integer	Read	Contains the product ID of the product used for rating.
DETAIL.CUST_A.ACCOUNT_PARENT_ID	String	Read	Contains the customer's account number.
DETAIL.CUST_A.PRODUCT.SERVICE_ID	String	Read	Contains the ID of the product.
DETAIL.UTC_TIME_OFFSET	String	Read	Contains the UTC time offset that normalizes the charging start timestamp to the UTC time zone. All validity timestamps in the BRM customer data are stored in normalized UTC time. The format is +/-HHMM.
DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the start timestamp of the event. The time zone information for this timestamp is stored in the BDR_UTC_TIME_OFFSET field.

FCT_MainRating

The FCT_MainRating module performs the main rating functionality in a pipeline. See "About Main Rating".

Dependencies

The FCT_MainRating module requires a connection to the following data modules:

- DAT_Calendar
- DAT_Currency
- DAT_TimeModel
- DAT_Rateplan

- DAT_PriceModel
- DAT_ModelSelector

This module must run after at least one of the following modules:

- FCT_CarrierIcRating
- FCT_CustomerRating
- FCT_GlobalRating
- FCT_SegRateNoCust

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–82 lists the FCT_MainRating registry entries.

Table 36–82 FCT_MainRating Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
CalendarDataModule	Specifies the connection to the DAT_Calendar module.	Yes
CurrencyDataModule	Specifies the connection to the DAT_Currency module.	No
ModelSelectorDataModule	Specifies the connection to the DAT_ModelSelector module.	Yes
PriceDataModule	Specifies the connection to the DAT_PriceModel module.	Yes
RateplanDataModule	Specifies the connection to the DAT_Rateplan module.	Yes
TimeDataModule	Specifies the connection to the DAT_TimeModel module.	Yes

Sample Registry

```
MainRating
{
  ModuleStart = FCT_MainRating
  Module
  {
    Active = True
    RateplanDataModule = RateplanDataModule
    CalendarDataModule = CalendarDataModule
    TimeDataModule = TimeDataModule
    PriceDataModule = PriceDataModule
    CurrencyDataModule = ifw.DataPool.CurrencyDataModule
    ModelSelectorDataModule = Model selector module
  }
}
```

Semaphore File Entries

Table 36–83 lists the FCT_MainRating Semaphore file entry.

Table 36–83 FCT_MainRating Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.MainRating.Module.Active = False
```

EDR Container Fields

Table 36–84 lists the FCT_MainRating EDR container fields.

Table 36–84 FCT_MainRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_CBD DETAIL.ASS_CBD	Block	Read	Data block for rate data.
ASS_CBD_CHARGE_PACKET DETAIL.ASS_CBD.CP	Block	Read	Charge packet for rate data.
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging time stamp.
WHOLESALE_CHARGED_AMOUNT_VALUE DETAIL.WHOLESALE_CHARGED_AMOUNT_VALUE	Decimal	Read	Contains the wholesale charge amount value.
ASS_CBD_RECORD_NUMBER DETAIL.ASS_CBD.RECORD_NUMBER	Integer	Read	Contains the record number.
ASS_CBD_IMPACT_CATEGORY DETAIL.ASS_CBD.CP.IMPACT_CATEGORY	String	Read	Contains the impact category.
RATEPLAN_CODE DETAIL.ASS_CBD.CP.RATEPLAN_CODE	String	Read/Write	Contains the rate plan code.
RATEPLAN_TYPE DETAIL.ASS_CBD.CP.RATEPLAN_TYPE	String	Write	Contains the rate plan type.
TIMEMODEL_CODE DETAIL.ASS_CBD.CP.TIMEMODEL_CODE	String	Write	Contains the time model code.
TIMEZONE_CODE DETAIL.ASS_CBD.CP.TIMEZONE_CODE	String	Write	Contains the time zone code.
DAY_CODE DETAIL.ASS_CBD.CP.DAY_CODE	String	Write	Contains the special day code.
TIME_INTERVAL_CODE DETAIL.ASS_CBD.CP.TIME_INTERVAL_CODE	String	Write	Contains the time interval code.
PRICEMODEL_CODE DETAIL.ASS_CBD.CP.PRICEMODEL_CODE	String	Write	Contains the price model code.

Table 36–84 (Cont.) FCT_MainRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
PRICEMODEL_TYPE DETAIL.ASS_CBD.CP.PRICEMODEL_TYPE	String	Write	Contains the price model type.
SERVICE_CODE_USED DETAIL.ASS_CBD.CP.SERVICE_CODE_USED	String	Write	Contains the service code.
SERVICE_CLASS_USED DETAIL.ASS_CBD.CP.SERVICE_CLASS_USED	String	Write	Contains the service class.
CHARGED_AMOUNT_VALUE DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE	Decimal	Write	Contains the charge amount value for the event.
CHARGED_AMOUNT_CURRENCY DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_CURRENCY	String	Write	Contains the currency amount.
CHARGED_CURRENCY_TYPE DETAIL.ASS_CBD.CP.CHARGED_CURRENCY_TYPE	String	Write	Contains the currency type.
CHARGED_TAX_TREATMENT DETAIL.ASS_CBD.CP.CHARGED_TAX_TREATMENT	String	Write	Contains the tax treatment.
ASS_CBD_CHARGING_START_TIMESTAMP DETAIL.ASS_CBD.CP.CHARGING_START_TIMESTAMP	Date	Write	Contains the charging time stamp.
ROUNDED_QUANTITY_VALUE DETAIL.ASS_CBD.CP.ROUNDED_QUANTITY_VALUE	Decimal	Write	Contains the rounded quantity value.
ROUNDED_QUANTITY_UOM DETAIL.ASS_CBD.CP.ROUNDED_QUANTITY_UOM	String	Write	Contains the rounded quantity UoM.
USAGE_GL_ACCOUNT_CODE DETAIL.ASS_CBD.CP.USAGE_GL_ACCOUNT_CODE	String	Write	Contains the G/L code.
REVENUE_GROUP_CODE DETAIL.ASS_CBD.CP.REVENUE_GROUP_CODE	String	Write	Contains the revenue group.
RUMGROUP DETAIL.ASS_CBD.CP.RUMGROUP	String	Write	Contains the RUM group.
RUM DETAIL.ASS_CBD.CP.RUM	String	Write	Contains the RUM.
RESSOURCE DETAIL.ASS_CBD.CP.RESOURCE	String	Write	Contains the resource.
INTERN_DISCOUNT_MODEL DETAIL.ASS_CBD.CP.INTERN_DISCOUNT_MODEL	Integer	Write	Contains the discount model.
INTERN_RATEPLAN DETAIL.ASS_CBD.CP.INTERN_RATEPLAN	String	Write	Contains the internal rate plan.
INTERN_RATEPLAN_VERSION DETAIL.ASS_CBD.CP.INTERN_RATEPLAN_VERSION	Integer	Write	Contains the rate plan version.
INTERN_FIX_COST DETAIL.ASS_CBD.CP.INTERN_FIX_COST	Decimal	Read	Contains the fixed cost amount.

Table 36–84 (Cont.) FCT_MainRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
INTERN_PRICE_MDL_STEP_INFO DETAIL.ASS_CBD.CP.INTERN_PRICE_MDL_STEP_INFO	String	Write	Contains information about the price model steps used to calculate the charge in the charge packet.
DETAIL.CUST_A.PRODUCT.RATEPLAN_CODE	String	Read	Contains the rate plan code of the product to match with the rate plan in the charge breakdown record.
DETAIL.CUST_A.INTERN_RATING_PRODUCTS	String	Read	Contains the rating product indexes. This is a comma-separated list of all rating products' indexes associated with the same service and event, and their priorities.
DETAIL.CUST_A.INTERN_FOUND_PP_INDEX	String	Write	Contains the index of the highest priority rating product. This is the product with the highest rate priority for an event.
DETAIL.CUST_A.LEAST_COST	Integer	Read	Indicates whether to use least cost rating for an EDR. **1** turns it off; **2** turns it on.
DETAIL.CUST_A.PROMOTIONAL_SAVING	Integer	Read	Indicates whether to use promotional savings for an EDR. **1** turns it off; **2** turns it on.
DETAIL.CUST_A.PROMOTION	Integer	Read	Indicates whether to use overlay promotions for an EDR. **1** turns it off; **2** turns it on.

FCT_MainZoning

The FCT_MainZoning module performs zoning for multi-segment zoning. See "Setting Up Multi-Segment Zoning" in *BRM Setting Up Pricing and Rating*.

Dependencies

Requires a connection to the DAT_Zone module.

This module must run after the FCT_SegZoneNoCust module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–85 lists the FCT_MainZoning registry entries.

Table 36–85 FCT_MainZoning Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
ZoneDataModule	Specifies the connection to the DAT_Zone module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes

Sample Registry

```
MainZoning
{
  ModuleName = FCT_MainZoning
  Module
  {
    Active = True
    ZoneDataModule = integRate.DataPool.ZoneDataModule
  }
}
```

Semaphore File Entries

Table 36–86 lists the FCT_MainZoning Semaphore file entries.

Table 36–86 FCT_MainZoning Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.MainZoning.Module.Active = False
```

EDR Container Fields

Table 36–87 lists the FCT_MainZoning EDR container fields.

Table 36–87 FCT_MainZoning EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging time stamp.
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code.
INTERN_A_NUMBER_ZONE DETAIL.INTERN_A_NUMBER_ZONE	String	Read	Contains the zone for the A number.

Table 36–87 (Cont.) FCT_MainZoning EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
INTERN_B_NUMBER_ZONE DETAIL.INTERN_B_NUMBER_ZONE	String	Read	Contains the zone for the B number.
ASS_ZBD_ZONE_DESCRIPTION DETAIL.ASS_ZBD.ZP.ZONE_DESCRIPTION	String	Write	Contains the zone description for displaying on invoices.
ASS_ZBD_RECORD_NUMBER DETAIL.ASS_ZBD.RECORD_NUMBER	Integer	Read	Contains the record number.
ASS_ZBD_ZONE_RESULT_VALUE_WS DETAIL.ASS_ZBD.ZP.ZONE_RESULT_VALUE_WS	String	Write	Contains the wholesale zone.
ASS_ZBD_ZONE_RESULT_VALUE_RT DETAIL.ASS_ZBD.ZP.ZONE_RESULT_VALUE_RT	String	Write	Contains the retail zone.
ASS_ZBD_ZONEMODEL_CODE DETAIL.ASS_ZBD.ZP.ZONEMODEL_CODE	String	Write	Contains the zone model code.
ASS_ZBD_INTERN_ZONE_MODEL DETAIL.ASS_ZBD.ZP.INTERN_ZONE_MODEL	Integer	Read	Contains the zone model.
ASS_ZBD_INTERN_APN_GROUP DETAIL.ASS_ZBD.ZP.INTERN_APN_GROUP	String	Write	Contains the APN group.
ASS_ZBD_ZONE_ENTRY_NAME DETAIL.ASS_ZBD.ZP.ZONE_ENTRY_NAME	String	Write	Contains the destination description for this combination of service code and impact category.

FCT_NOSP

The FCT_NOSP module maps network source and destination to new values. See "Identifying the Network Operator/Service Provider".

Dependencies

Requires a connection to the DAT_NOSP module.

This module must be run before segment rating is performed.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–88 lists the FCT_NOSP registry entries.

Table 36–88 FCT_NOSP Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataModule	Specifies the connection to the DAT_NOSP module. See "DAT_NOSP". See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes
MapGroup	Specifies the map group that the NOSP mappings belong to. You can use this entry in a semaphore file.	Yes

Sample Registry

```
Zoning
{
  ModuleName = FCT_NOSP
  Module
  {
    Active = True
    DataModule = ifw.DataPool.NospData
    MapGroup = MOBILE
  }
}
```

Semaphore File Entries

Table 36–89 lists the FCT_NOSP Semaphore file entries.

Table 36–89 FCT_NOSP Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
MapGroup	Specifies the map group.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.NOSP.Module.Active = False
```

EDR Container Fields

Table 36–90 lists the FCT_NOSP EDR container fields.

Table 36–90 FCT_NOSP EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
SOURCE_NETWORK DETAIL.SOURCE_NETWORK	String	Read/Write	Contains the source network.
DESTINATION_NETWORK DETAIL.DESTINATION_NETWORK	String	Read/Write	Contains the destination network.
A_NUMBER DETAIL.A_NUMBER	String	Read	Contains the A number.

FCT_NumberPortability

The FCT_NumberPortability module specifies the new network operator for an existing phone number. See "Managing Number Portability" in *BRM Telco Integration*.

Dependencies

Requires a connection to the DAT_NumberPortability module.

This module must be run before the zoning and rating modules.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–91 lists the FCT_NumberPortability registry entries.

Table 36–91 FCT_NumberPortability Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive Default = **False**. You can use this entry in a semaphore file.	Yes
DataModule	Specifies the connection to the DAT_NumberPortability module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes
DefaultSourceNetwork	Specifies the default network operator ID of the source network. This ID is used if there is no ID present in the data retrieved from the DAT_NumberPortability module.	Yes

Table 36–91 (Cont.) FCT_NumberPortability Registry Entries

Entry	Description	Mandatory
DefaultDestinationNetwork	Specifies the default network operator ID of the destination network. This ID is used if there is no ID present in the data retrieved from the DAT_NumberPortability module.	Yes
OverwriteNetwork	If the **DefaultSourceNetwork** and **DefaultDestinationNetwork** fields are empty, overwrites the source and destination network with the value configured in the DAT_NumberPortability module. Default = **True**.	No
OverwriteNetworkType	If the SOURCE_NETWORK_TYPE and DESTINATION_NETWORK_TYPE fields are empty, overwrites the SOURCE_NETWORK_TYPE and DESTINATION_NETWORK_TYPE fields with the type of network that is populated in the source and destination network. Default = **True**.	No

Sample Registry

```
NumberPortability
{
  ModuleName = FCT_NumberPortability
  Module
  {
    Active = True
    DataModule = integrate.DataPool.NPortData
    DefaultSourceNetwork = D030
    DefaultDestinationNetwork = D017
  }
}
```

Semaphore File Entries

Table 36–92 lists the FCT_NumberPortability Semaphore file entry.

Table 36–92 FCT_NumberPortability Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.NumberPortability.
Module.Active = False
```

EDR Container Fields

Table 36–93 lists the FCT_NumberPortability EDR container fields.

Table 36–93 FCT_NumberPortability EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
A_NUMBER DETAIL.A_NUMBER	String	Read	Specifies the event originator.
B_NUMBER DETAIL.B_NUMBER	String	Read	Specifies the event receiver.
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Specifies the charging timestamp. The format is: *YYYYMMDDhhmmss*
IGNORE_NP DETAIL.IGNORE_NP	Integer	Read/ Write	Specifies whether FCT_NumberPortability should look for network operator IDs for A and B number. Default is **0** (cleared).
SOURCE_NETWORK DETAIL.SOURCE_NETWORK	String	Write	Specifies the source network. This can either be the PLMN ID or any logical operator code.
SOURCE_NETWORK_TYPE DETAIL.SOURCE_NETWORK_TYPE	String	Write	Specifies the source network type, for example GSM 900.
DESTINATION_NETWORK DETAIL.DESTINATION_NETWORK	String	Write	Specifies the network to which an event is routed.
DESTINATION_NETWORK_TYPE DETAIL.DESTINATION_NETWORK_TYPE	String	Write	Specifies the destination network type, for example GSM 900.

FCT_Opcode

The FCT_Opcode module uses the DAT_ConnectionPool module to connect to the CM and calls the appropriate opcode for the request.

Dependencies

The FCT_Opcode module requires a connection to the "DAT_ConnectionPool" module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–94 lists the FCT_Opcode registry entries.

Table 36–94 FCT_Opcode Registry Entries

Entry	Description	Mandatory
Active	Specifies whether the module is active or inactive **True** = Active (default) **False** = Inactive	Yes
Retries	Specifies the number of times to try the request on the CM Default = **2**	Yes
Logging	Logs each opcode called from the processing pipeline Default = **False**	Yes
ConnectionPoolDataModule	Specifies a connection to the DAT_ConnectionPool module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes

Sample Registry

```
EdrOpcodeCall
{
  ModuleName = FCT_Opcode
  Module
  {
    Active = True
    Retries = 2
    Logging = True
    ConnectionPoolDataModule = ifw.DataPool.CMConnectionPool.Module
  }
}
```

EDR Container Fields

FCT_Opcode uses the EDR container fields listed in Table 36–95:

Table 36–95 FCT_Opcode Container Fields

Alias Field Name Default Field Name	Type	Access	Description
OPCODE_FLAG DETAIL.OPCODE_FLAG	Integer	Read	The flag that specifies the behavior of the opcode
OPCODE_NODE DETAIL.OPCODE_NODE	String	Read	Name of the opcode
PCM_OP_EBUF DETAIL.PCM_OP_EBUF	pin_ebuf_t	Read	Error buffer

FCT_PrefixDesc

The FCT_PrefixDesc module maps phone number prefixes to destination descriptions. See "Creating Call Destination Descriptions".

Dependencies

Requires a connection to the DAT_PrefixDesc module.

This module can run from anywhere.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–96 lists the FCT_PrefixDesc registry entries.

Table 36–96 FCT_PrefixDesc Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
PrefixDataModule	Specifies the connection to the DAT_PrefixDesc module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes

Sample Registry

```
{
  ModuleName = FCT_PrefixDesc
  Module
  {
    Active = True
    PrefixDataModule = PrefixDescData
  }
}
```

Semaphore File Entries

Table 36–97 lists the FCT_PrefixDesc Semaphore file entry.

Table 36–97 FCT_PrefixDesc Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.PrefixDesc.Module.Active = False
```

EDR Container Fields

Table 36–98 lists the FCT_PrefixDesc EDR container fields.

Table 36–98 FCT_PrefixDesc EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
B_NUMBER DETAIL.B_NUMBER	String	Read	Contains the B number.
DESCRIPTION DETAIL.DESCRIPTION	String	Write	Contains the call destination description.

FCT_PreRating

The FCT_PreRating module calculates zones and creates impact category. See "Setting up prerating" in *BRM Setting Up Pricing and Rating*.

Dependencies

Requires a connection to the DAT_Rateplan and the DAT_Zone module.

This module must be run before the FCT_MainRating module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–99 lists the FCT_PreRating registry entries.

Table 36–99 FCT_PreRating Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
RateplanDataModule	Specifies the connection to the DAT_Rateplan module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes
ZoneDataModule	Specifies the connection to the DAT_Zone module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes

Sample Startup Registry

```
PreRating
{
  ModuleStart = FCT_PreRating
  Module
  {
    Active = True
    RateplanDataModule = ifw.DataPool.RateplanDataModule
    ZoneDataModule = ifw.DataPool.ZoneDataModule
  }
}
```

Semaphore File Entries

Table 36–100 lists the FCT_PreRating Semaphore file entry.

Table 36–100 FCT_PreRating Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.PreRatingZone.Module.Active = False
```

EDR Container Fields

Table 36–101 lists the FCT_PreRating EDR container fields.

Table 36–101 FCT_PreRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging time stamp. The time stamp is used for calculating the impact category by comparing it to the dates in the VALID_FROM and VALID_TO fields in the **IFW_STANDARD_ZONE** database table.
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the service code. The service code is used for calculating the impact category by comparing it with the value in the SERVICECODE field in the **IFW_STANDARD_ZONE** database table. The module writes the service code to the ASS_CBD_SERVICE_CODE and SERVICE_CLASS_USED fields.
INTERN_SERVICE_CLASS DETAIL.INTERN_SERVICE_CLASS	String	Read	Contains the service class. The module writes the service class to the ASS_CBD_SERVICE_CODE and SERVICE_CLASS_USED fields.

Table 36–101 (Cont.) FCT_PreRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_CBD_RECORD_NUMBER DETAIL.ASS_CBD.RECORD_NUMBER	String	Read	Contains the charge breakdown record number. Charge breakdown records are processed only if the record number = 0 (newly created).
ASS_CBD_SERVICE_CODE DETAIL.ASS_CBD.SERVICE_CODE	String	Write	Contains the service code. Set with the value of the INTERN_SERVICE_CODE field.
ASS_CBD_ZONEMODEL_CODE DETAIL.ASS_CBD.CP.ZONEMODEL_CODE	String	Write	Contains the zone model code. Set with the value of the INTERN_SERVICE_CLASSE field.
ASS_CBD_IMPACT_CATEGORY DETAIL.ASS_CBD.CP.IMPACT_CATEGORY	String	Write	Contains the impact category. Set with the zoning results by using the value from either the ZONE_WS or ZONE_RT in the **IFW_STANDARD_ZONE** database table, depending on rate plan type.
RATEPLAN_CODE DETAIL.ASS_CBD.CP.RATEPLAN_CODE	String	Write	Contains a comma-separated list of rate plan codes for all rating products. This list arranged by product priority, with the highest priority first and the lowest priority last. Set with the value of the CODE field in the **IFW_RATEPLAN** database table.
RATEPLAN_TYPE DETAIL.ASS_CBD.CP.RATEPLAN_TYPE	String	Write	Contains the rate plan type. Set with the value of the TYPE field in the **IFW_RATEPLAN** database table.
SERVICE_CODE_USED DETAIL.ASS_CBD.CP.SERVICE_CODE_USED	String	Write	Contains the service code. Set with the value from the INTERN_SERVICE_CODE field.

Table 36–101 (Cont.) FCT_PreRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
SERVICE_CLASS_USED DETAIL.ASS_CBD.CP.SERVICE_CLASS_USED	String	Write	Contains the service class. Set with the value from the INTERN_SERVICE_CLASS field.
INTERN_ORIGIN_NUM_ZONE DETAIL.ASS_CBD.CP.INTERN_ORIGIN_NUM_ZONE	String	Read	Contains the area code for the A number. The area code is used for calculating the impact category by comparing it to ORIGIN_AREACODE field in the **IFW_STANDARD_ZONE** database table.
INTERN_DESTIN_NUM_ZONE DETAIL.ASS_CBD.CP.INTERN_DESTIN_NUM_ZONE	String	Read	Contains the area code for the B number. The area code is used for calculating the impact category by comparing it to DESTIN_AREACODE field in the **IFW_STANDARD_ZONE** database table.
INTERN_RATEPLAN DETAIL.ASS_CBD.CP.INTERN_RATEPLAN	String	Read	Contains the rate plan. Set with the value from the RATEPLAN field in the **IFW_RATEPLAN** database table.
INTERN_RATEPLAN_VERSION DETAIL.ASS_CBD.CP.INTERN_RATEPLAN_VERSION	Integer	Write	Contains the rate plan version. Set with the value from the VERSION field in the **IFW_RATEPLAN_VER** database table.
ASS_CBD_INTERN_ZONE_MODEL DETAIL.ASS_CBD.CP.INTERN_ZONE_MODEL	Integer	Write	Contains the zone model. Set with the value from the ZONEMODEL field in the **IFW_RATEPLAN_VER** database table.
ASS_CBD_INTERN_APN_GROUP DETAIL.ASS_CBD.CP.INTERN_APN_GROUP	String	Write	Contains the APN group. Set with the value from the APN_GROUP field in the **IFW_ZONEMODEL** database table.

Table 36–101 (Cont.) FCT_PreRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_CBD_INTERN_GEOMODEL DETAIL.ASS_CBD.CP.INTERN_GEOMODEL	Integer	Write	Contains the geographical model, if the MODELTYPE field in the **IFW_ZONEMODEL** database table is set to **L**. Set with the value from the GEOMODEL field in the **IFW_ZONEMODEL** database table.
ASS_CBD_INTERN_RULESET DETAIL.ASS_CBD.CP.INTERN_RULESET	String	Write	Contains the rule set, if the MODELTYPE field in the **IFW_ZONEMODEL** database table is set to **L**. Set with the value from the RULESET field in the **IFW_GEOGRAPHICAL_MODEL** database table.
ASS_CBD_ZONE_DESCRIPTION DETAIL.ASS_CBD.CP.ZONE_DESCRIPTION	String	Write	Contains the zone description for displaying on invoices.
ASS_ZBD_ZONE_ENTRY_NAME DETAIL.ASS_ZBD.ZP.ZONE_ENTRY_NAME	String	Write	Contains the destination description for displaying on invoices.
RATE_PLAN_NAME DETAIL.CUST_A.PRODUCT.RATEPLAN_NAME	String	Write	Contains the rate plan name for the purchased product.
INTERN_RATING_PRODUCTS DETAIL.CUST_A.INTERN_RATING_PRODUCTS	String	Write	Contains the indexes of the candidate products that can be used for rating.
INTERN_FOUND_PP_INDEX DETAIL.CUST_A.INTERN_FOUND_PP_INDEX	String	Write	Contains the purchased product index of the product or service used.
CUST_A_LEAST_COST_RATING DETAIL.CUST_A.LEAST_COST	String	Write	Specifies if least cost rating is to be used for rating the EDR.

FCT_PreRecycle

The FCT_PreRecycle module gets the file of rejected EDRs from the reject stream output directory. The module puts the reject EDR file into the pipeline input directory for recycling. It uses the same input folder as the incoming CDR files.

See:

- Configuring Standard Recycling
- Recycling EDRs in Pipeline-Only Systems

Registry Entries

Table 36–102 lists the FCT_PreCycle registry entries.

Table 36–102 FCT_PreCycle Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive	Yes
RecycleSuffix	Specifies the suffix for the file that contains the EDRs that need recycling. The suffix is automatically appended when the file is moved from the reject directory to the input directory. If it is empty, no suffix is added. Default = **_Recy**	No
RecyFileName	Specifies the file name and path for the file that contains the EDRs that need recycling.	Yes

Sample Registry

```
PreRecycle
{
   ModuleName = FCT_PreRecycle
   Module
   {
      Active = True
      RecycleSuffix = RecycleFile
      RecyFileName = ./recycle.dat
   }
}
```

Semaphore File Entries

When you update the registry, you must select one of the following entries listed in Table 36–103:

Table 36–103 FCT_PreRecycle Semaphore File Entries

Entry	Description
Recycle	The module runs in real processing mode You can specify a list of files to recycle. If this entry is empty, all files from the reject directory are recycled.
RecycleTest	The module runs in test mode You can specify a list of files to test recycling with. If this entry is empty, all files from the reject directory are recycled.

Sample Semaphore File Entries

- Recycle all files:

```
ifw.Pipelines.PRE_RECYCLE.Functions.Processing.FunctionPool.PreRecycle.Module.Recycle {}
```

- Recycle only specific files:

```
ifw.Pipelines.PRE_RECYCLE.Functions.Processing.FunctionPool.PreRecycle.Module.Recycle.
File = ./format_a/abc.cdr

ifw.Pipelines.PRE_RECYCLE.Functions.Processing.FunctionPool.PreRecycle.Module.Recycle.
```

```
File = ./format_a/xyz.cdr
```

- Test recycle all files:

```
ifw.Pipelines.PRE_RECYCLE.Functions.Processing.FunctionPool.PreRecycle.Module.RecycleTest {}
```

- Test recycle only specific files:

```
ifw.Pipelines.PRE_RECYCLE.Functions.Processing.FunctionPool.PreRecycle.Module.RecycleTest.
File = ./format_a/abc.cdr

ifw.Pipelines.PRE_RECYCLE.Functions.Processing.FunctionPool.PreRecycle.Module.RecycleTest.
File = ./format_a/xyz.cdr
```

EDR Container Fields

Table 36–104 lists the FCT_PreRecycle EDR container fields.

Table 36–104 FCT_PreRecycle EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
INTERN_PROCESS_STATUS DETAIL.INTERN_PROCESS_STATUS	Integer	Write	Contains the internal process status.
TRANSACTION_ID INTERNAL.TRANSACTION_ID	Decimal	Read	Contains the transaction ID.
STREAM_NAME INTERNAL.STREAM_NAME	String	Read	Contains the stream name.
PROCESS_STATUS INTERNAL.PROCESS_STATUS	Integer	Write	Contains the process status.

FCT_PreSuspense

This module is used both by the standard recycling mechanism and by the Suspense Manager service integration component that you purchase separately. Both implementations are described below.

Important: This module stores the contents of the EDR before any other modules change it. This module must take the original version of an EDR as input, so that it can be recycled after being suspended.

Standard Recycling Implementation

The BRM FCT_PreSuspense module adds suspense-related information to EDRs. It adds the DETAIL.ASS_SUSPENSE_EXT data block to the EDR if that data block does not already exist.

Suspense Manager Implementation - Adding Queryable Fields

When used with Suspense Manager, FCT_PreSuspense configures the queryable fields for EDRs suspended in a specific pipeline. You must enter the table and field names from the **/suspended_usage** object, as well as the corresponding EDR container fields. See "Registry Entries" for syntax and formatting information.

If no **QueryableFields** registry entry is present, the HEADER.QUERYABLE_FIELDS_MAPPING and DETAIL.ASS_SUSPENSE_EXT.QUERYABLE_FIELDS_MAPPING are set to empty strings.

> **Important:** Each table listed in the FCT_PreSuspense registry must also be configured in the RE Loader **Infranet.properties** file so that RE Loader can load into these tables.

This module adds queryable field mapping information to the HEADER.QUERYABLE_FIELDS_MAPPING field of the EDR.This information is passed to the Suspended Event (SE) Loader to generate control files for loading suspended usage records.

> **Note:** You add one set of queryable fields representing one **/suspended_usage** subclass *per pipeline*. For example, for a single pipeline that accepts **/suspended_usage/telco/gsm** records, you can pick queryable fields from the **/suspended_usage/telco** and **/suspended_usage/telco/gsm** subclasses. You could not pick queryable fields from **/suspended_usage/telco/gprs**, because it requires a separate pipeline.

FCT_PreSuspense serializes the original EDR container and stores it in DETAIL.ASS_SUSPENSE_EXT.EDR_BUF. It also stores the EDR size in DETAIL.ASS_SUSPENSE_EXT.EDR_SIZE.

When call records are being recycled, this module sets values for:

- HEADER.BATCH_ID, based on value already set by INP_Recycle in pre-recycle pipeline. This ID is appended with information to ensure that it remains unique.
- DETAIL.BATCH_ID with the batch ID value from the header record.

Changing the Way Batch IDs Are Set

You use the **KeepExistingBatchIds** registry entry to determine whether an EDR's batch ID is preserved as it is processed by the pipeline. A value of **False** directs the pipeline to change the batch ID; a value of **True** preserves it.

Set **KeepExistingBatchIds** to **False** (the default value) if the current pipeline is not part of a chain. Also set this entry to **False** if the current pipeline is the first pipeline in a chain of pipelines that includes the FCT_PreSuspense module.

Set this entry to **True** if the current pipeline is part of a chain of pipelines, and it is not the first pipeline in the chain that includes FCT_PreSuspense.

See "Tracking EDRs by using batch IDs" in *BRM Collecting Revenue Assurance Data*.

Dependencies

This module must be the first preprocessing module in a pipeline.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–105 lists the FCT_PreSuspense registry entries.

Table 36–105 FCT_PreSuspense Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
KeepExistingBatchIds	A value of **True** preserves the Batch ID in the detail record of each EDR in an EDR file. A value of **False** sets the Batch ID of each EDR to the Batch ID contained in the header record of the batch input file. The default is **False**.	No

QueryableFields
(Suspense Manager only)
	Specifies which fields in which tables are queryable from the Suspense Management Center. Includes the EDR container fields that correspond to the database fields.
	This entry is only useful to customers who have purchased Suspense Manager.
	Format:
	`QueryableFields`
	`{`
	`  table_name_1`
	`  {`
	`  database_column_name_1 = edr_container_field_1`
	`  database_column_name_2 = edr_container_field_2`
	`  }`
	`  table_name_2`
	`  {`
	`  database_column_name_3 = edr_container_field_3`
	`  database_column_name_4 = edr_container_field_4`
	`  }`
	`}`
	If this entry is not present, this module sets HEADER.QUERYABLE_FIELDS_MAPPING and DETAIL.ASS_SUSPENSE_EXT.QUERYABLE_FIELDS to empty strings.
		Yes

Sample Registry

```
#---------------------------
# PreSuspense FCT
#---------------------------
PreSuspense
{
  ModuleName                = FCT_PreSuspense
  Module
  {
    Active                  = True
    QueryableFields
    {
      # table name. If more than one table, use a separate block
      SUSP_USAGE_TELCO_INFO_T
      {
        # format : <database_column_name> = <edr_conatiner_field_name>
        BYTES_IN = DETAIL.VOLUME_RECEIVED
        BYTES_OUT = DETAIL.VOLUME_SENT
        CALLED_TO = DETAIL.B_NUMBER
        #CALLING_FROM = DETAIL.B_NUMBER
```

```
        CALL_DURATION = DETAIL.DURATION
        PRIMARY_MSID = DETAIL.A_NUMBER
        SERVICE_TYPE = DETAIL.BASIC_SERVICE
        START_TIME = DETAIL.CHARGING_START_TIMESTAMP
        USAGE_TYPE = DETAIL.USAGE_TYPE
      }
      #Examples for GPRS calls
      #SUSP_USAGE_TELCO_GPRS_INFO_T
      #{
        # format : <database_column_name> = <edr_conatiner_field_name>
        #APN = DETAIL.ASS_GPRS_EXT.APN_ADDRESS
        #GGSN_ADDRESS = DETAIL.ASS_GPRS_EXT.GGSN_ADDRESS
        #NODE_ID = DETAIL.ASS_GPRS_EXT.NODE_ID
        #SECONDARY_MSID = DETAIL.ASS_GPRS_EXT.PORT_NUMBER
        #SGSN_ADDRESS = DETAIL.ASS_GPRS_EXT.SGSN_ADDRESS
      #}
      #SUSP_USAGE_TELCO_GSM_INFO_T
      #{
        #APN = DETAIL.ASS_GSMW_EXT.APN_ADDRESS
        #CELL_ID = DETAIL.ASS_GSMW_EXT.CELL_ID
        #DESTINATION_SID = DETAIL.ASS_GSMW_EXT.TERMINATING_SWITCH_IDENTIFICATION
        #DIALED_NUMBER = DETAIL.ASS_GSMW_EXT.DIALED_DIGITS
        #ORIGIN_SID = DETAIL.ASS_GSMW_EXT.ORIGINATING_SWITCH_IDENTIFICATION
        #SECONDARY_MSID = DETAIL.ASS_GSMW_EXT.PORT_NUMBER
      #}
    }
  }
}
```

Semaphore File Entries

Table 36–106 lists the FCT_PreSuspense Semaphore file entry.

Table 36–106 FCT_PreSuspense Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.PreSuspense.Module.Active = False
```

EDR Container Fields

Table 36–107 lists the FCT_PreSuspense EDR container fields.

Table 36–107 FCT_PreSuspense EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
QUERYABLE_FIELDS_MAPPING HEADER.QUERYABLE_FIELDS_MAPPING	String	Write	Database column names and data types that map queryable fields. This field is used by Suspense Manager only. This is an empty string for standard recycling implementations.
PIPELINE_NAME DETAIL.ASS_SUSPENSE_EXT.PIPELINE_NAME	String	Write	The name of the pipeline, derived from the pipeline registry.
SOURCE_FILENAME DETAIL.ASS_SUSPENSE_EXT.SOURCE_FILENAME	String	Write	The source file name. The same as INTERNAL.STREAM_NAME.
EDR_BUF DETAIL.ASS_SUSPENSE_EXT.EDR_BUF	String	Write	A stored representation of the EDR container with its original field values.
EDR_SIZE DETAIL.ASS_SUSPENSE_EXT.EDR_SIZE	Integer	Write	The size of DETAIL.ASS_SUSPENSE_EXT.EDR_BUF.
QUERYABLE_FIELDS DETAIL.ASS_SUSPENSE_EXT.QUERYABLE_FIELDS	String	Write	The queryable field values defined in the registry. This field is used by Suspense Manager only. This is an empty string for standard recycling implementations.
BATCH_ID DETAIL.BATCH_ID	String	Write	At recycling, the value is set from HEADER.BATCH_ID.
INTERN_PROCESS_STATUS DETAIL.INTERN_PROCESS_STATUS	Integer	Read	Indicates whether the EDR is being recycled or test recycled.
ORIGINAL_BATCH_ID DETAIL.ORIGINAL_BATCH_ID	String	Write	Set the first time EDRs go through the pipeline (not being recycled or rerated.) Sets the value from HEADER.BATCH_ID in the header record.
BATCH_ID HEADER.BATCH_ID	String	Read	At recycling, modifies this value to ensure that it is unique.

FCT_RateAdjust

The FCT_RateAdjust module adjusts the charge for an EDR after rating has been performed.

Dependencies

If the rate adjustment data is stored in the database, the module requires a connection to the Pipeline Manager database.

This module must run after the FCT_MainRating module to adjust the rate.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–108 lists the FCT_RateAdjust registry entries.

Table 36–108 FCT_RateAdjust Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes, if the data is stored in the database. Otherwise not used.
RateAdjustFile	Specifies file name that contains the rate adjustment data. See "Creating a Rate Adjustment Rules File". You can use this entry in a semaphore file.	Yes, if the data is stored in a file. Otherwise not used.
Source	Specifies where the rate adjustment data is stored: ■ **File** ■ **Database**	Yes

Sample Registry for the Database Interface

```
RateAdjust
{
  ModuleName = FCT_RateAdjust
  Module
  {
    Active = True
    Source = Database
    DataConnection = ifw.DataPool.DataConnection
  }
}
```

Sample Registry for the File Interface

```
RateAdjust
{
  ModuleName = FCT_RateAdjust
  Module
  {
    Active = True
    Source = File
    RateAdjustFile = /data/etc/discount.dat
  }
}
```

Semaphore File Entries

Table 36–109 lists the FCT_RateAdjust Semaphore file entries.

Table 36–109 FCT_RateAdjust Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
RateAdjustFile	Specifies file name that contains the rate adjustment data.
Reload	Reloads data from the database. See "Reloading data into a Pipeline Manager module" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.RateAdjustment.Module.Active = False
```

EDR Container Fields

Table 36–110 lists the FCT_RateAdjust EDR container fields.

Table 36–110 FCT_RateAdjust EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_CBD DETAIL.ASS_CBD	Block-Index	Read	Data block.
ASS_CBD_CHARGE_PACKET DETAIL.ASS_CBD.CP	Block-Index	Read	Data block.
ASS_CBD_RECORD_NUM DETAIL.ASS_CBD.RECORD_NUMBER	Integer	Read	Contains the record number.
USAGE_CLASS DETAIL.USAGE_CLASS	String	Read	Contains the usage class.
USAGE_TYPE DETAIL.USAGE_TYPE	String	Read	Contains the usage type.
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code.
INTERN_SERVICE_CLASS DETAIL.INTERN_SERVICE_CLASS	String	Read	Contains the internal service class.
BDR_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging time stamp.
DURATION DETAIL.DURATION	Decimal	Read	Contains the duration of the event.
SOURCE_NETWORK DETAIL.SOURCE_NETWORK	String	Read	Contains the source network.

Table 36–110 (Cont.) FCT_RateAdjust EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
DESTINATION_NETWORK DETAIL.DESTINATION_NETWORK	String	Read	Contains the destination network.
INTERN_RATEPLAN DETAIL.ASS_CBD.CP.INTERN_RATEPLAN	String	Read	Contains the rate plan code.
INTERN_RATEPLAN_VERSION DETAIL.ASS_CBD.CP.INTERN_RATEPLAN_VERSION	Integer	Read	Contains the rate plan version.
ASS_CBD_IMPACT_CATEGORY DETAIL.ASS_CBD.CP.IMPACT_CATEGORY	String	Read	Contains the impact category.
CHARGED_AMOUNT_VALUE DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE	Decimal	Read/Write	Contains the charge amount value.

Database Tables

The FCT_RateAdjust module uses the IFW_RATEADJUST table to set the rate adjustment rules. You define rate adjustments in Pricing Center. See "About Pipeline Rate Adjustments" in *BRM Setting Up Pricing and Rating*.

For information about the fields in database tables, see the documentation in *Pipeline_Home***database**.

> **Note:** For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

FCT_Recycle

The FCT_Recycle module runs at the end of the pipeline It does either of the following:

- When the FCT_PreRecycle module runs in test mode, the FCT_Recycle module creates a report about the processing, but does not send the EDRs to an output file.
- When the FCT_PreRecycle module runs in recycle mode, the FCT_Recycle module sends the results to an output file, and attaches a sequence number to the output file.

See:

- Configuring Standard Recycling
- Recycling EDRs in Pipeline-Only Systems

Dependencies

> **Important:** You must configure the FCT_Recycle module as the last module of all function modules in the pipeline.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–111 lists the FCT_Recycle registry entries.

Table 36–111 FCT_Recycle Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
RecycleLog	Specifies the log file parameters: ■ **MessageFilePath** Specifies the path where the log file can find the message database. ■ **MessageFilePrefix** Specifies the prefix for collecting the files from the message file path. ■ **MessageFileSuffix** Specifies the suffix for collecting the files from the message file path. ■ **FilePath** Specifies the path in which the log file is written. ■ **FilePrefix** Specifies the prefix for the log file. ■ **FileSuffix** Specifies the suffix for the log file.	Yes

Sample Registry

```
Recycle
{
   ModuleName = FCT_Recycle
   Module
   {
      Active = True
      RecycleLog
      {
         MessageFilePath = ..
         MessageFilePrefix = Framework
         MessageFileSuffix = msg
         FilePath = ../tmp/log01
         FilePrefix = rej_
         FileSuffix = .log
      }
   }
}
```

Semaphore File Entries

Table 36–112 lists the FCT_Recycle Semaphore file entry.

Table 36–112 FCT_Recycle Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.Recycle.Module.Active = True
```

EDR Container Fields

Table 36–113 lists the FCT_Recycle EDR container fields.

Table 36–113 FCT_Recycle EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
STREAM_NAME INTERNAL.STREAM_NAME	String	Read	Contains the stream name.
SEQ_CHECK INTERNAL.SEQ_CHECK	Integer	Write	Specifies the sequence check.
SEQ_GENERATION INTERNAL.SEQ_GENERATION	Integer	Write	Specifies the sequence generation.
OFFSET_GENERATION INTERNAL.OFFSET_GENERATION	Integer	Write	Specifies the offset generation.
PROCESS_STATUS INTERNAL.PROCESS_STATUS	Integer	Read	Contains the internal process status.

FCT_Reject

The FCT_Reject module analyzes the errors in an EDR and, if necessary, moves the EDR to a reject file.

See:

- Configuring Standard Recycling
- Recycling EDRs in Pipeline-Only Systems

Dependencies

This module must run after the rating and discount modules.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–114 lists the FCT_Reject registry entries.

Table 36–114 FCT_Reject Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
CallAssemblingModule	Provides data to the FCT_CallAssembling module to ensure that assembled calls are processed completely. See "Recycling Assembled EDRs".	No
MinErrorSeverity	Specifies to reject EDRs that have a specified severity. To allow warning and normal messages without rejecting the EDR, set this entry to 3. Default = Not used. See "Processing EDRs with Errors".	Yes
NotifyonReject	Specifies if other modules should be notified if an EDR is rejected. Default = **True**	No
StreamMap	Specifies a list of error types mapped to output streams. Important: A **UseRejectStream** entry is required to use **StreamMap**. See "Specifying Multiple Reject Streams".	No
UseRejectStream	Specifies whether to use the reject output stream: ■ **True**. Rejected EDRs are sent to the reject stream. ■ **False**. Rejected EDRs are sent to the normal output stream, but flagged as discarded.Important: A **StreamMap** entry is required to use **UseRejectStream**. See "Using a Reject Output Stream".	No

Sample Registry

```
Reject
{
   ModuleName = FCT_Reject
   {
         Active = True
         NotifyOnReject = True
         UseRejectStream = True
         CallAssemblingModule = ifw.Pipelines.Pipe.Functions.Standard.FunctionPool.CallAssembling
         StreamMap
               {
               error_type_1 = output_stream_1
               error_type_2 = output_stream_2
               intern = RejectStream
               logic = ReturnStream
               }
   }
}
```

Semaphore File Entries

Table 36–115 lists the FCT_Reject Semaphore file entry.

Table 36–115 FCT_Reject Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.Rejection.Module.Active = False
```

Sample Output Configuration

You configure the reject stream in the registry in two places:

- In the pipeline configuration, for example:

```
Pipelines
{
ALL_RATE
{
Active   =   TRUE
.
.
.
RejectStream   =   RejectOutput
```

- In the Output configuration, for example:

```
# Output stream for rejected events
RejectOutput
{
 ModuleName = OUT_Reject
 Module
 {
  OutputStream
  {
   ModuleName = EXT_OutFileManager
   Module
   {
    OutputPath = ./samples/wireless/data/rej
    OutputPrefix = test
    OutputSuffix = .rej
    TempPrefix   = tmp

    TempDataPath    = ./samples/wireless/data/rej
    TempDataPrefix  = rej.tmp.
    TempDataSuffix  = .data

    Replace = TRUE
    DeleteEmptyFile = TRUE
   }
  }
 }
} # end of Reject
```

See "Configuring Output for Rejected or Duplicate EDRs".

Important: To ensure output file integrity, specify a unique combination of OutputPath, OutputSuffix, and OutputPrefix values for each output stream defined in the registry.

EDR Container Fields

Table 36–116 lists the FCT_Reject container fields.

Table 36–116 FCT_Reject Container Fields

Field name	Access	Description
DISCARDING DETAIL.DISCARDING	Read/Write	**Read:** This field is used to detect pre-rejected EDRs. If this field is 1, the EDR is always rejected. **Write:** If this field is 0 and the EDR is rejected, the field is set to 1.
ERROR_REJECT_TYPE DETAIL.ERROR_REJECT_TYPE	Read	Specifies the type of error in the EDR. This determines which stream the EDR is directed to.

FCT_Rounding

The FCT_Rounding module performs rounding for rating and discounting. Add this module to the pipeline after the processing module for which it is rounding.

For more information, see "About configuring the FCT_Rounding module" and "About resource rounding" in *BRM Setting Up Pricing and Rating*.

Dependencies

Requires a connection to the DAT_Currency module.

This module must run after the FCT_RateAdjust module if you want rating results to be rounded and after FCT_Discount module if you want discount results to be rounded. FCT_Rounding must come after each module for which rounding should occur. For batch rating, it must come before the FCT_ApplyBalance module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–117 lists the FCT_Rounding registry entries.

Table 36–117 FCT_Rounding Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive: **True** = Active **False** = Inactive	Yes
CurrencyDataModule	Specifies the connection to the DAT_Currency module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes
Mode	Specifies the process for which rounding is applied: **Rating** = Round the balance impact of rating. **Taxation** = Round the balance impact of taxation. **Discounting** = Round the balance impact of discounting. See "About configuring the FCT_Rounding module" in *BRM Setting Up Pricing and Rating*.	Yes

Sample Registry

```
Rounding
{
   ModuleName = FCT_Rounding
   Module
   {
      Active = TRUE
      Mode   = Rating
      CurrencyDataModule = ifw.DataPool.CurrencyDataModule
   }
}
```

EDR Container Fields

Table 36–118 lists the FCT_Rounding EDR container fields.

Table 36–118 FCT_Rounding EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
EVENT_TYPE DETAIL.EVENT_TYPE	String	Read	Specifies the event type.
ASS_CBD_RECORD_NUMBER DETAIL.ASS_CBD.RECORD_NUMBER	Integer	Read	Specifies the record number.
ASS_CBD_RECORD_TYPE DETAIL.ASS_CBD.RECORD_TYPE	String	Read	Specifies the record type.
RESOURCE_ID DETAIL.ASS_CBD.DP.RESOURCE_ID	Decimal	Read	Specifies the resource ID.
CHARGED_AMOUNT_CURRENCY DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_CURRENCY	Decimal	Read	Specifies the currency of the charged amount.
CHARGED_AMOUNT_VALUE DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE	Integer	Write	Specifies the rounded charged amount.

Table 36–118 (Cont.) FCT_Rounding EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
TAX_VALUE DETAIL.ASS_CBD.TP.TAX_VALUE	Decimal	Write	Specifies the rounded tax value.
GRANTED_AMOUNT DETAIL.ASS_CBD.DP.GRANTED_AMOUNT	Integer	Write	Specifies the rounded non-currency discount amount.
RELATED_CHARGE_INFO_ID DETAIL.ASS_CBD.TP.RELATED_CHARGE_INFO_ID	Integer	Read	Specifies the index of the corresponding charge packet for which the tax was calculated.

FCT_RSC_Map

The FCT_RSC_Map module performs rate service class (RSC) mapping.

See "About Rate-Service Class Mapping".

Dependencies

Requires a connection to the Pipeline Manager database.

This module must run before FCT_MainRating module to change the Service Code and Service Class fields of the Charge Packet. These fields are used by the main rating module to find out the rate to be applied for a particular call.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–119 lists the FCT_RSC_Map registry entries.

Table 36–119 FCT_RSC_Map Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
DefaultRSCGroup	Specifies the default RSC group to use when the RSC group is not specified in the EDR.	Yes

Sample Registry

```
RSC_Mapping
{
  ModuleName = FCT_RSC_Map
  Module
  {
    Active = True
```

```
        DataConnection = integrate.DataPool.DataConnection
        DefaultRscGroup = testGroup
    }
}
```

Semaphore File Entries

Table 36–120 lists the FCT_RSC_Map Semaphore file entries.

Table 36–120 FCT_RSC_Map Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
Reload	Reloads data from the database into memory. See "Reloading data into a Pipeline Manager module" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.RateServiceClassMap.
Module.Active = False
```

EDR Container Fields

Table 36–121 lists the FCT_RSC_Map EDR container fields.

Table 36–121 FCT_RSC_Map EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging start time.
INTERN_SLA_RSC_GROUP DETAIL.INTERN_SLA_RSC_GROUP	String	Read	Contains the internal RSC group.
QOS_REQUESTED DETAIL.QOS_REQUESTED	String	Read	Contains the quality of service requested.
QOS_USED DETAIL.QOS_USED	String	Read	Contains the quality of service used.
USAGE_TYPE DETAIL.USAGE_TYPE	String	Read	Contains the usage type.
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code.
INTERN_SERVICE_CLASS DETAIL.INTERN_SERVICE_CLASS	String	Read	Contains the internal service class.
INTERN_USAGE_CLASS DETAIL.INTERN_USAGE_CLASS	String	Read	Contains the internal usage class.
INTERN_RATEPLAN DETAIL.ASS_CBD.CP.INTERN_RATEPLAN	String	Read	Contains the internal rate plan.

Table 36–121 (Cont.) FCT_RSC_Map EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_CBD_IMPACT_CATEGORY DETAIL.ASS_CBD.CP.IMPACT_CATEGORY	String	Read	Contains the impact category.
SERVICE_CODE_USED DETAIL.ASS_CBD.CP.SERVICE_CODE_USED	String	Write	Contains the service code used.
SERVICE_CLASS_USED DETAIL.ASS_CBD.CP.SERVICE_CLASS_USED	String	Write	Contains the service class used.

Database Interface

FCT_RSC_Map uses the following database tables:

- **IFW_RSC_MAP**. Stores the mapping rules. You create the mapping rules in Pricing Center. See "About Rate-Service Class Mapping".
- The **IFW_RSC_GROUP**. Stores the RSC groups used for RSC mapping.
- The **IFW_SERVICECLASS**. Stores the service class codes used when defining service codes.

For information about the fields in database tables, see the documentation in *Pipeline_Home***database**.

Note: For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

FCT_SegRateNoCust

The FCT_SegRateNoCust module assigns a segment to an EDR based on the source network instead of customer information. For information about multi-segment rating, see "About Multi-Segment Rating".

You can also assign segments by using customer data. To do so, use the FCT_CustomerRating module. See "About Using the FCT_CustomerRating Module for Multi-Segment Rating".

Dependencies

Requires a connection to the Pipeline Manager.

This module must be run after FCT_Account.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–122 lists the FCT_SegRateNoCust registry entries.

Table 36–122 FCT_SegRateNoCust Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
Segments	Specifies a list of mapping rules. See "Configuring Segments in the FCT_SegRateNoCust Module".	Yes

Sample Registry

```
SegRateNoCust
{
  ModuleName = FCT_SegRateNoCust
  Module
  {
    Active = True
    DataConnection = ifw.DataPool.Database
    Segments
    {
      26201 = SegmentD1
      26202 = SegmentD2
    }
  }
}
```

Semaphore File Entries

Table 36–123 lists the FCT_SegRateNoCust Semaphore file entries.

Table 36–123 FCT_SegRateNoCust Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
Reload	Reloads data from the database into memory. See "Reloading data into a Pipeline Manager module" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.SegRateNoCust.Module.Active = False
```

EDR Container Fields

Table 36–124 lists the FCT_SegRateNoCust EDR container fields.

Table 36–124 FCT_SegRateNoCust EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_CBD DETAIL.ASS_ZBD	Block	Create	Data block.
ASS_ZBD_ZONE_PACKET DETAIL.ASS_ZBD.ZP	Block	Create	Data block.
SOURCE_NETWORK DETAIL.SOURCE_NETWORK	String	Read	Contains the source network.
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging time stamp.
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code.
ASS_ZBD_RECORD_TYPE DETAIL.ASS_ZBD.RECORD_TYPE	String	Write	Contains the record type.
ASS_ZBD_SEGMENT_CODE DETAIL.ASS_ZBD.SEGMENT_CODE	String	Write	Contains the segment code.
ASS_ZBD_SERVICE_CODE DETAIL.ASS_ZBD.SERVICE_CODE	String	Write	Contains the resulting service code.
ASS_ZBD_INTERN_ZONE_MODEL DETAIL.ASS_ZBD.ZP.INTERN_ZONE_MODEL	Integer	Write	Contains the zone model.

Database Interface

The FCT_SegRateNoCust module uses the **IFW_SEGRATE_LNK** database table. This table stores segments used for multi-segment rating and zoning.

For information about the fields in database tables, see the documentation in *Pipeline_Home***database**.

FCT_SegZoneNoCust

The FCT_SegZoneNoCust module finds the segment using the source network information instead of using the customer information. See "Setting Up Multi-Segment Zoning" in *BRM Setting Up Pricing and Rating*.

Dependencies

Requires a connection to the Pipeline Manager database.

This module must run before the FCT_MainZoning module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–125 lists the FCT_SegZoneNoCust registry entries.

Table 36–125 FCT_SegZoneNoCust Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
Segments	Specifies a list of mapping rules. Each rule defines the connection between the source network and the segment. See "Configuring segments in the FCT_SegZoneNoCust module" in *BRM Setting Up Pricing and Rating*.	Yes

Sample Registry

```
SegZoneNoCust
{
  ModuleName = FCT_SegZoneNoCust
  Module
  {
    Active = True
    DataConnection = ifw.DataPool.Database
    Segments
    {
       26201 = SegmentD1
       26202 = SegmentD2
    }
  }
}
```

Semaphore File Entries

Table 36–126 lists the FCT_SegZoneNoCust Semaphore file entries.

Table 36–126 FCT_SegZoneNoCust Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
Reload	Reloads data from the database into memory. See "Reloading data into a Pipeline Manager module" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.SegZoneNoCust.Active = False
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.SegZoneNoCust.Module.Reload { }
```

EDR Container Fields

Table 36–127 lists the FCT_SegZoneNoCust EDR container fields.

Table 36–127 FCT_SegZoneNoCust EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_ZBD DETAIL.ASS_ZBD	Block	Create	Data block.
ASS_ZBD_ZONE_PACKET DETAIL.ASS_ZBD.ZP	Block	Create	Data block.
SOURCE_NETWORK DETAIL.SOURCE_NETWORK	String	Read	Contains the source network.
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging time stamp.
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code.
ASS_ZBD_RECORD_TYPE DETAIL.ASS_ZBD.RECORD_TYPE	String	Write	Contains the record type.
ASS_ZBD_SEGMENT_CODE DETAIL.ASS_ZBD.SEGMENT_CODE	String	Write	Contains the segment code.
ASS_ZBD_SERVICE_CODE DETAIL.ASS_ZBD.SERVICE_CODE	String	Write	Contains the resulting service code.
ASS_ZBD_INTERN_ZONE_MODEL DETAIL.ASS_ZBD.ZP.INTERN_ZONE_MODEL	Integer	Write	Contains the zone model.

Database Tables

The FCT_SegZoneNoCust module uses the **IFW_SEGZONE_LNK** database table. This table stores segments used for multi-segment rating and zoning.

For information about the fields in database tables, see the documentation in *Pipeline_Home***database**.

FCT_ServiceCodeMap

The FCT_ServiceCodeMap module maps external service codes to internal service codes.

For information about mapping service codes, see "Mapping service codes and service classes" in *BRM Setting Up Pricing and Rating*.

Dependencies

Requires a connection to the Pipeline Manager database.

Some modules require an internal service code, so this module should run near the front of a pipeline.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–128 lists the FCT_ServiceCodeMap registry entries.

Table 36–128 FCT_ServiceCodeMap Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
MapGroup	Specifies the map group that the service code map belongs to. You can use this entry in a semaphore file.	Yes

Sample Registry

```
ServiceCodeMapping
{
  ModuleName = FCT_ServiceCodeMap
  Module
  {
     Active = True
     DataConnection = integrate.DataPool.Database
     MapGroup = serviceMapGroup
  }
}
```

Semaphore File Entries

Table 36–129 lists the FCT_ServiceCodeMap Semaphore file entries.

Table 36–129 FCT_ServiceCodeMap Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
MapGroup	Specifies the mapping rule set.
Reload	Reloads data from the database into memory. See "Reloading data into a Pipeline Manager module" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ServiceCodeMap.Module.Reload {}
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ServiceCodeMap.Module.MapGroup =
ALL_RATE
```

EDR Container Fields

The FCT_ServiceCodeMap module reads data from the EDR to map the external service code to the internal service code. The module then writes the internal service code and service class to the EDR.

Table 36–130 lists the FCT_ServiceCodeMap EDR container fields.

Table 36–130 FCT_ServiceCodeMap EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
BASIC_SERVICE DETAIL.BASIC_SERVICE	String	Read	Contains the external service code.
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Write	Contains the internal service code.
INTERN_SERVICE_CLASS DETAIL.INTERN_SERVICE_CLASS	String	Write	Contains the internal service class.
INTERN_USAGE_CLASS DETAIL.INTERN_USAGE_CLASS	String	Read	Contains the internal usage class.
ASS_GSMW_LOCATION_AREA_INDICATOR DETAIL.ASS_GSMW_EXT.LOCATION_AREA_INDICATOR	String	Read	Contains the GSMW extension location area.
ASS_GPRS_LOCATION_AREA_INDICATOR DETAIL.ASS_GPRS_EXT.LOCATION_AREA_INDICATOR	String	Read	Contains the GPRS extension location area.
QOS_REQUESTED DETAIL.QOS_REQUESTED	String	Read	Contains the quality of service requested.
QOS_USED DETAIL.QOS_USED	String	Read	Contains the quality of service used.
BDR_RECORD_TYPE DETAIL.RECORD_TYPE	String	Read	Contains the record type.

Database Tables

The FCT_ServiceCodeMap module uses the following database tables:

- **IFW_SERVICE_MAP**. Maps external service codes to internal service codes. See "About Mapping Services" in *BRM Setting Up Pricing and Rating*.

  To create service code mappings, use Pricing Center. See "About creating map groups" in *BRM Setting Up Pricing and Rating*.

- **IFW_MAP_GROUP**. Stores the map groups used for service code mapping. See "About Mapping Services" in *BRM Setting Up Pricing and Rating*.

For information about the fields in database tables, see the documentation in *Pipeline_Home***database**.

Note: For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

FCT_SocialNo

The FCT_SocialNo module flags social numbers for special processing. See "Setting Up Social Numbers".

Dependencies

If the social number data is stored in the database, the FCT_SocialNo module requires a connection to the Pipeline Manager database.

This module can run anywhere.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–131 lists the FCT_SocialNo registry entries.

Table 36–131 FCT_SocialNo Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	No
DataConnection	Specifies the connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes, if the data is stored in the database. Otherwise not used.
FileName	Specifies file that contains the social number data. See "Creating a Social Number Data File".	Yes, if the data is stored in a file.
ReuseOnFailure	Specifies if the module should continue to use the old data if the **Reload** command fails. If **True**, the old data is used. The default is **False**.	Yes
SocialNoMapSize	Specifies the size of the in-memory map that stores social numbers. For a large set of social numbers to be loaded, specifying this parameter will enhance loading performance.	No
Source	Specifies where the social number data is stored: ■ **File** ■ **Database**	Yes

Sample Registry for the Database Interface

```
SocialNo
{
  ModuleName = FCT_SocialNo
  Module
  {
    ReuseOnFailure = false
    Active = true
    Source = database
    DataConnection = dataPool
  }
}
```

Sample Registry for the File Interface

```
SocialNo
{
```

```
    ModuleName = FCT_SocialNo
    Module
    {
      Active = True
      ReuseOnFailure = False
      Source = File
      FileName = ../daten/socialno.dat

  }
```

Semaphore File Entries

Table 36–132 lists the FCT_SocialNo Semaphore file entries.

Table 36–132 FCT_SocialNo Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
FileName	Reloads data from the specified file if **Source** parameter is set to **File**.
Reload	Reloads data from the database if **Source** parameter is set to **Database**. See "Reloading data into a Pipeline Manager module" in *BRM System Administrator's Guide*.
ReuseOnFailure	Specifies if the module should continue to use the old data if the **Reload** command fails.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.SocialNo.Module.Active = False
```

EDR Container Fields

The FCT_SocialNo module uses the following EDR container fields listed in Table 36–133 to flag social numbers for further processing.

Table 36–133 FCT_SocialNo EDR Container Fields.

Alias Field Name Default Field Name	Type	Access	Description
B_MODIFICATION_INDICATOR DETAIL.B_MODIFICATION_INDICATOR	String	Read/Write	Contains the modification indicator.
B_NUMBER_ZONE DETAIL.INTERN_B_NUMBER_ZONE	String	Read	Contains the B number.

Database Interface

The FCT_SocialNo module uses the **IFW_SOCIALNUMBER** database table. This table stores social numbers. To define social numbers, use Pricing Center. See "Setting Up Social Numbers".

For information about the fields in database tables, see the documentation in *Pipeline_Home***database**.

Note: For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

FCT_Suspense

This module is used both by the standard recycling mechanism and by the Suspense Manager service integration component that you purchase separately. Both implementations are described below.

Important: With one exception, FCT_Suspense must be the last function module in the pipeline. This ensures that it processes the final EDR container, including overwritten field values and enrichment field values. However, if FCT_AggreGate is used, it can be after FCT_Suspense.

Standard Recycling Implementation

As part of the standard recycling mechanism, the BRM FCT_Suspense function module:

- Routes EDRs being recycled from SuspenseCreateOutput to suspenseUpdateOutput.
- Determines the brand for each suspended call.
- Logs the results of test recycling (if the **LogTestResults** registry entry is set).

Suspense reason and subreason codes are not supported with standard recycling, and these codes are all set to **O** (other).

Suspense Manager Implementation

As part of Suspense Manager, this module adds suspense reason and suspense subreason codes to EDRs. The specific errors that it adds are based on the error codes assigned to the EDR by the pipeline and the mapping information stored in the **/config/suspense_reason_code** object. If no **/config/suspense_reason_code** object is present, this module sets the suspense reason to **O** (other).

Dependencies

Requires a connection to the BRM database.

This module must be the last one in the pipeline.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–134 lists the FCT_Suspense registry entries.

Table 36–134 FCT_Suspense Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the BRM database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
LogTestResults	For standard recycling only. Determines whether the results of test recycling are logged. If this entry is not present, the results are not logged. If set to **True**, the **RecycleLog** entry must also be present in the FCT_Suspense registry. Default = **False** See "pin_recycle".	No
RecycleLog	Specifies the log file parameters.	Yes, when **LogTestResults** is set to **True**.
RecycleLog.MessageFilePath	Specifies the path where the log file can find the message database.	Yes, when **LogTestResults** is set to **True**.
RecycleLog.MessageFilePrefix	Specifies the prefix for collecting the files from the message file path.	Yes, when **LogTestResults** is set to **True**.
RecycleLog.MessageFileSuffix	Specifies the suffix for collecting the files from the message file path.	Yes, when **LogTestResults** is set to **True**.
RecycleLog.FilePath	Specifies the path in which the log file is written.	Yes, when **LogTestResults** is set to **True**.
RecycleLog.FilePrefix	Specifies the prefix for the log file.	Yes, when **LogTestResults** is set to **True**.
RecycleLog.FileSuffix	Specifies the suffix for the log file.	Yes, when **LogTestResults** is set to **True**.
SuspenseCreateStream	Specifies the output stream for newly suspended EDRs.	Yes
SuspenseUpdateStream	Specifies the output stream for recycled EDRs.	Yes

Sample Registry

```
#---------------------------
# Suspense FCT
#---------------------------
```

```
Suspense
{
  ModuleName                = FCT_Suspense
  Module
  {
    Active                  = True
    SuspenseCreateStream    = SuspenseCreateOutput
    SuspenseUpdateStream    = SuspenseUpdateOutput
    EdrFieldMap             = DETAIL.ASS_GSMW_EXT.PORT_NUMBER
    DataConnection          = ifw.DataPool.LoginInfranet
    LogTestResults          = True
    RecycleLog
    {
       MessageFilePath = ..
       MessageFilePrefix = Framework
       MessageFileSuffix = msg
       FilePath = ../tmp/log01
       FilePrefix = rej_
       FileSuffix = .log
    }
  }
}
```

Semaphore File Entries

Table 36–135 lists the FCT_Suspense file entry.

Table 36–135 FCT_Suspense Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.Suspense.Module.Active = False
```

EDR Container Fields

Table 36–136 lists the FCT_Suspense EDR container fields.

Table 36–136 FCT_Suspense EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ERROR_CODE DETAIL.ASS_SUSPENSE_EXT.ERROR_CODE	String	Write	The error code for the most severe error reported by the pipeline for the EDR.
SUSPENSE_REASON DETAIL.ASS_SUSPENSE_EXT.SUSPENSE_REASON	String	Write	The suspense reason. Mapped from the error code. Used by Suspense Manager only. This field is set to **0** for standard recycling implementations.
SUSPENSE_SUBREASON DETAIL.ASS_SUSPENSE_EXT.SUSPENSE_SUBREASON	String	Write	The suspense subreason. Mapped from the error code. Used by Suspense Manager only. This field is set to **0** for standard recycling implementations.
EDR_BUF DETAIL.ASS_SUSPENSE_EXT.EDR_BUF	String	Write	A stored representation of the EDR container including fields overwritten and enriched by the pipeline.
EDR_SIZE DETAIL.ASS_SUSPENSE_EXT.EDR_SIZE	Integer	Write	The size of DETAIL.ASS_ SUSPENSE_EXT.EDR_ BUF.
PROCESS_STATUS DETAIL.INTERN_PROCESS_STATUS	Integer	Read	Indicates whether the EDR is being recycled or test recycled.
SUSPENSE_STATUS DETAIL.ASS_SUSPENSE_EXT.SUSPENSE_STATUS	Integer	Write	Indicates whether the EDR is suspended or successfully recycled.
BATCH_ID DETAIL.BATCH_ID	String	Write	Writes the batch ID from the header record, except during recycling.
BATCH_ID HEADER.BATCH_ID	String	Read	Written during recycling.

FCT_Timer

The FCT_Timer module sets the timer ID for an EDR and stores a copy of the EDR when the original EDR is sent to the processing pipeline.

If an EDR times out, FCT_Timer sets the timeout flag to **True** and sends:

- The original EDR with a timeout flag to the exception pipeline, which is the pipeline to which the original EDR is sent when it times out.
- The duplicate EDR to the timeout pipeline, which is the pipeline to which the duplicate EDR is sent when it times out.

If the EDR is processed within the time limit, FCT_Timer sets the timeout flag to **False**.

FCT_Timer also handles heartbeat and keep-alive messages by automatically resetting the timer at the interval specified in the **KeepAliveInterval** registry entry when there is message traffic between the CM and the Intelligent Network (IN).

Dependencies

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–137 lists the FCT_Timer registry entries.

Table 36–137 FCT_Timer Registry Entries

Entry	Description	Mandatory
Active	Specifies whether the module is active or inactive **True** = Active (default). **False** = Inactive.	Yes
Threads	Specifies the number of threads running this module. Default = **1**. See "Configuring Single-Threaded or Multithreaded Operation" in *BRM System Administrator's Guide*.	Yes
Reactors	Specifies the number of reactor objects to use. The reactors detect timeout events and then dispatch the events to the timer handler. The recommended number of reactors is from 1 to 5.	No
TimeoutQueue	Specifies the pipeline queue to which the duplicate EDR is sent when the EDR times out.	Yes
KeepAliveInterval	Specifies the idle timeout value in milliseconds, which specifies how long to wait before sending the original EDR to the exception pipeline. Default = **3000**.	Yes
Timeout	Specifies the idle timeout value in microseconds, which specifies how long to wait before sending the duplicate EDR to the timeout pipeline. Default = **100000**.	Yes
NoOpcodeNumbers	Specifies the number of the BRM opcode that prevents duplicate EDRs from being created. When the OPCODE_NUM EDR field is set to the specified number, the module does not create a duplicate EDR for the Timeout pipeline. Opcode numbers are defined in header (***.h**) files in the *BRM_Home*/**include/ops** directory. *BRM_Home* is the directory where you installed BRM components.	No

Sample Registry

```
Timer
{
  ModuleName = FCT_Timer
   Module
  {
   Active = TRUE
   Threads = 1
   Reactors = 3
   TimeoutQueue = ifw.IPCQueues.TimeoutQueue
   KeepAliveInterval = 3000 ### milliseconds
   Timeout = 100000 ### microseconds
```

```
   NoOpcodeNumbers = 5000, 50001
   }
}
```

EDR Container Fields

FCT_Timer uses the EDR container fields listed in Table 36–138:

Table 36–138 FCT_Timer EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
TIMER_ID DETAIL.TIMER_ID	Integer	Write	ID assigned to the EDR container's timer when FCT_SetTimer schedules it. This ID is required to cancel the timer.
CURRENT_TIME DETAIL.CURRENT_TIME	Integer	Read	Current system time
TIMEOUT_OFFSET DETAIL.TIMEOUT_OFFSET	Integer	Write	Offset from the current system time
OPCODE_NUM DETAIL.OPCODE_NUM	Integer	Read	Specifies the number of the BRM opcode that performs the requested action. Opcode numbers are defined in header (***.h**) files in the *BRM_Home***/include/ops** directory.
TIMEOUT_FLAG DETAIL.TIMEOUT_FLAG	Integer	Write	Specifies whether the EDR has timed out: ■ **0** is False. ■ **1** is True.
SESSION_ID DETAIL.SESSION_ID	String	Write	ID required to cancel the timer.
MILLISEC_TIME DETAIL.MILLISEC_TIME	Integer	Read	The latency time in milliseconds.

FCT_TriggerBill

The FCT_TriggerBill module sends EDRs to the billing-trigger output stream to trigger immediate billing for the associated accounts. It also sets a billing-trigger error code used to route the EDRs to the suspense output stream, and the **Trigger_Billing** recycle key used to retrieve the suspended EDRs for recycling.

For more information, see "Setting up Pipeline-Triggered Billing" in *BRM Configuring and Running Billing*.

Dependencies

Configure the FCT_TriggerBill module to run before the FCT_MainRating module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–139 lists the FCT_TriggerBill registry entries.

Table 36–139 FCT_TriggerBill Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive: **True** = Active **False** = Inactive	Yes
TriggerBillCreateStream	Specifies the billing-trigger output stream module.	Yes

Sample Registry

```
TriggerBill
{
   ModuleName = FCT_TriggerBill
   Module
   {
      Active = TRUE
      TriggerBillCreateStream = TriggerBillCreateOutput
   }
}
```

Semaphore File Entries

Table 36–140 lists the FCT_TriggerBill Semaphore file entry.

Table 36–140 FCT_TriggerBill Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

EDR Container Fields

Table 36–141 lists the FCT_TriggerBill EDR container fields.

Table 36–141 FCT_TriggerBill EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ACTG_NEXT_DATE DETAIL.CUST.A.ACTG_NEXT_DATE	String	Read	The date that the current monthly cycle ends. Used to determine the accounting cycle to which the EDR belongs.
BILL_STATE DETAIL.CUST.A.BILL_STATE	String	Read	The billing state. Possible values are: ▪ **PIN_ACTG_CYCLE_OPEN** (unbilled) ▪ **PIN_ACTG_CYCLE_CLOSED** (billed)
CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	String	Read	The timestamp when the call started. Used to determine the accounting cycle to which the EDR belongs.

Table 36–141 (Cont.) FCT_TriggerBill EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ERROR_CODE DETAIL.ASS_SUSPENSE_EXT.ERROR_CODE	String	Write	Specifies the billing-trigger error code. Used to send the EDR to the suspense output stream.
RECYCLE_KEY DETAIL.ASS_SUSPENSE_EXT.RECYCLE_KEY	String	Write	The key value that identifies the EDRs suspended for pipeline-triggered billing. Set to **Trigger_Billing** when BILL_STATE is unbilled and ACTG_NEXT_DATE is passed.
UTC_TIME_OFFSET DETAIL.UTC_TIME_OFFSET	X(5)	Read	The UTC time offset.

FCT_UoM_Map

The FCT_UoM_Map module converts the unit of measurement (UoM) of an incoming EDR to a UoM needed for rating a particular service.

For more information, see "Converting Units of Measurement" in *BRM Setting Up Pricing and Rating*.

Dependencies

Requires a connection to the Pipeline Manager database.

Must run after the FCT_ServiceCodeMap module, and before the rating modules.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–142 lists the FCT_UoM_Map registry entries.

Table 36–142 FCT_UoM_Map Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
Mapping	Specifies the mapping rules.	Yes
Mapping.AssCBDServiceCode	Specifies the service code field in the associated charge breakdown records that is used for the mapping.	No
Mapping.InternServiceCode	Specifies the service code field in the basic detail block that is used for the mapping.	No

Sample Registry

```
UoM_Map
{
  ModuleName = FCT_UoM_Map
  Module
  {
    Active = True
    DataConnection = integrate.DataPool.Login
    Mapping
    {
      InternServiceCode = INTERN_SERVICE_CODE
      AssCBDServiceCode = ASS_CBD_SERVICE_CODE
    }
  }
}
```

Semaphore File Entries

Table 36–143 lists the FCT_UoM_Map Semaphore file entries.

Table 36–143 FCT_UoM_Map Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
Reload	Reloads data from the database into memory.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UoM_Map.Module.Active = False
```

EDR Container Fields

Table 36–144 lists the FCT_UoM_Map EDR container fields.

Table 36–144 FCT_UoM_Map EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code.
ASS_CBD_SERVICE_CODE DETAIL.ASS_CBD.SERVICE_CODE	String	Read	Contains the charge breakdown service code.
DETAIL.ASSOCIATED_CHARGE.CHARGE_PACKET ASS_CBD_CHARGE_PACKET	Struct	Read/Write	Contains charge packet data.

Database Interface

FCT_UoM_Map accesses the following database tables:

- **IFW_SERVICE**. This table stores data about services and associated RUMs. To define services, use Pricing Center.

- **IFW_RUMGROUP_LNK**. This table defines a list of RUM/UOM pairs with the RUM group value obtained from the **IFW_SERVICE** table. To create RUMs, use Pricing Center. See "About Defining Ratable Usage Metrics (RUMs)" in *BRM Setting Up Pricing and Rating*.
- **IFW_UOM_MAP**. This table maps a UoM to a basic detail or to an associated charge packet. To create UoMs, use Pricing Center. See "About Defining Units of Measurement (UoMs)" in *BRM Setting Up Pricing and Rating*.
- **IFW_UOM**. This table stores the UoMs for pipeline rating.
- **IFW_ALIAS_MAP**. This table stores an alias name for each RUM and UoM.

FCT_UsageClassMap

The FCT_UsageClassMap module maps external codes for supplementary services, such as call forwarding, to internal usage classes. See "Mapping Usage Classes" in *BRM Setting Up Pricing and Rating*.

Dependencies

Requires a connection to the Pipeline Manager database.

The FCT_UsageClassMap module is run before the zoning and rating modules.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–145 lists the FCT_UsageClassMap registry entries.

Table 36–145 FCT_UsageClassMap Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes

Table 36–145 (Cont.) FCT_UsageClassMap Registry Entries

Entry	Description	Mandatory
MapGroup	Specifies the map group. You can use this entry in a semaphore file.	Yes
OverwriteUsageClass	Specifies if the external usage class should be overwritten by the internal one. The default is to not overwrite the external usage class; if you map usage codes, you should enable this entry. Default = **False**	No
OptimizeFor	Specifies if the module is configured to optimize memory consumption as well as pipeline startup time. **Memory** = Optimizes memory consumption and pipeline startup time. **No memory optimization** = Does not optimize memory consumption and pipeline startup time (the default). **Note** ■ Enabling this entry might have an adverse impact on the number of call detail records (CDRs) processed in a specific time interval. ■ This entry is read only at pipeline start up. Its value cannot be changed by using a semaphore.	No

Sample Registry

```
UsageClassMapping
{
  ModuleName = FCT_UsageClassMap
  Module
  {
    Active  = True
    DataConnection  =  ifw.DataPool.Database
    OverwriteUsageClass = False
    MapGroup  = mapGroup0
    OptimizeFor = Memory
  }
}
```

Semaphore File Entries

Table 36–146 lists the FCT_UsageClassMap Semaphore file entries.

Table 36–146 FCT_UsageClassMap Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
MapGroup	Specifies the mapping rule set.
Reload	Reloads data from the database into memory. See "Reloading data into a Pipeline Manager module" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module.Reload {}
```

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module.MapGroup =
ALL_RATE
```

EDR Container Fields

The FCT_UsageClassMap module adds the internal usage class to the EDR. All other fields in Table 36–147 are used for mapping the usage class.

Table 36–147 FCT_UsageClassMap EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
BDR_RECORD_TYPE DETAIL.RECORD_TYPE	String	Read	Contains the event record type.
USAGE_CLASS DETAIL.USAGE_CLASS	String	Read	Contains the external usage class.
USAGE_TYPE DETAIL.USAGE_CLASS	String	Read	Contains the external usage type.
WHOLESALE_IMPACT_CATEGORY DETAIL.WHOLESALE_IMPACT_CATEGORY	String	Read	Contains the wholesale impact category.
TARIFF_CLASS DETAIL.TARIFF_CLASS	String	Read	Contains the tariff class.
TARIFF_SUB_CLASS DETAIL.TARIFF_SUB_CLASS	String	Read	Contains the tariff subclass.
INTERN_USAGE_CLASS DETAIL.INTERN_USAGE_CLASS	String	Write	Contains the internal usage class.
CONNECT_TYPE DETAIL.CONNECT_TYPE	String	Read	Contains the connection type.
CONNECT_SUB_TYPE DETAIL.CONNECT_SUB_TYPE	String	Read	Contains the connection subtype.
APN_ADDRESS DETAIL.ASS_GPRS_EXT.APN_ADDRESS	String	Read	Contains the GPRS APN type.
ACTION_CODE DETAIL.ASS_GSMW_EXT.SS_PACKET.ACTION_CODE	String	Read	Contains the GSM SS packet action code.
SS_EVENT DETAIL.ASS_GSMW_EXT.SS_PACKET.SS_EVENT	String	Read	Contains the GSM SS packet action event.
INTERN_C_NUMBER_ZONE DETAIL.INTERN_C_NUMBER_ZONE	String	Read	Contains the internal normalized C number.
DETAIL.ASS_GPRS_EXT ASS_GPRS	String	Read	Contains the GPRS extension.
DETAIL.ASS_GSMW_EXT.SS_PACKET ASS_GSMW_SS_PACKET	String	Read	Contains the GSMW SS packet.

Database Tables

The FCT_UsageClassMap module uses the following database tables:

- The **IFW_USAGECLASS_MAP** table maps external supplementary service codes in the EDR to internal usage classes. See "About Mapping Services" in *BRM Setting Up Pricing and Rating*.
- The **IFW_USAGECLASS** table stores the usage classes that can be used as a result of usage class mapping. See "About Mapping Services" in *BRM Setting Up Pricing and Rating*.
- The **IFW_MAP_GROUP** table stores the map groups used for usage class mapping. See "About Mapping Services" in *BRM Setting Up Pricing and Rating*.

For information about the fields in database tables, see the documentation in *Pipeline_Home***database**.

Note: For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

FCT_USC_Map

The FCT_USC_Map module performs usage scenario mapping. See "Setting Up Usage Scenario Mapping" in *BRM Setting Up Pricing and Rating*.

Dependencies

This module needs a connection to the DAT_USC_Map module.

This module must run after the following:

- FCT_UsageClassMap
- ISC_UsageType
- FCT_PreRating

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–148 lists the FCT_USC_Map registry entries.

Table 36–148 FCT_USC_Map Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataModule	Specifies the connection to the DAT_USC_Map data module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes

Table 36–148 (Cont.) FCT_USC_Map Registry Entries

Entry	Description	Mandatory
DefaultUSCGroup	Specifies the USC group that contains the mapping rules. If no matching rule is found, the FCT_USC_Map module uses the rule in the default USC map group. You can use this entry in a semaphore file.	Yes
LogZoneModelNotFoundEntries	Specifies, if set to **True**, that all log entries in INF_NO_USC_MAPPING_ENTRY are logged into the Stream log. The default value is **False**.	No
Mode	Specifies the mode in which USC mapping is done. The **Rating** mode (the default) specifies that USC mapping is done using the zone model from the charge packets. Mapping in this mode provides the impact category for charge packets. The **Zoning** mode specifies that USC mapping is done using the zone model from the EDR detail block. Mapping in this mode provides impact categories for the detail block. Using the **Zoning** mode requires that the DETAIL.RETAIL_IMPACT_CATEGORY and DETAIL.WHOLESALE_IMPACT_CATEGORY fields are populated.	No

Sample Registry

```
USC_Mapping
{
  ModuleName = FCT_USC_Map
  Module
  {
    Active = True
    DefaultUSCGroup = usc_group
    DataModule = ifw.DataPool.USCDataModule
  }
}
```

Semaphore File Entries

Table 36–149 lists the FCT_USC_Map Semaphore file entry.

Table 36–149 FCT_USC_Map Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageScenarioMap.
Module.Active = True

ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageScenarioMap.
Module. DefaultUSCGroup  =  usc_group
```

EDR Container Fields

Table 36–150 lists the FCT_USC_Map EDR container fields.

Table 36–150 FCT_USC_Map EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	String	Read	Contains the charging time stamp.
INTERN_SLA_USC_GROUP DETAIL.INTERN_SLA_USC_GROUP	String	Read	Contains the internal USC group.
USAGE_TYPE DETAIL.USAGE_TYPE	String	Read/Write	Contains the usage type code. This field is updated only when the **Mode** registry entry is set to **Zoning**. It is not updated when the **Mode** entry is set to **Rating**.
RETAIL_IMPACT_CATEGORY DETAIL.RETAIL_IMPACT_CATEGORY	String	Read/Write	Contains the retail impact category.
WHOLESALE_IMPACT_CATEGORY DETAIL.WHOLESALE_IMPACT_CATEGORY	String	Read/Write	Contains the wholesale impact category.
WHOLESALE_CHARGED_AMOUNT_VALUE DETAIL.WHOLESALE_CHARGED_AMOUNT_VALUE	String	Read	Contains the wholesale charged amount value.
IC_DESCRIPTION DETAIL.IC_DESCRIPTION	String	Write	Contains the zone description for displaying on invoices.
IC_DESCRIPTION DETAIL.ASS_CBD.CP.IC_DESCRIPTION	String	Write	Contains the zone description for displaying on invoices.
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code.
INTERN_SERVICE_CLASS DETAIL.INTERN_SERVICE_CLASS	String	Read	Contains the internal service class.
DURATION DETAIL.DURATION	String	Read	Contains the duration of the event.
INTERN_USAGE_CLASS DETAIL.INTERN_USAGE_CLASS	String	Read	Contains the internal usage class code.
BDR_INTERN_ZONE_MODEL DETAIL.INTERN_ZONE_MODEL	String	Read	Contains the zone model.
RATEPLAN_TYPE DETAIL.ASS_CBD.CP.RATEPLAN_TYPE	String	Read	Contains the charge breakdown record rate plan type.
ASS_CBD_INTERN_ZONE_MODEL DETAIL.ASS_CBD.CP.INTERN_ZONE_MODEL	String	Read	Contains the charge breakdown record zone model.

Table 36–150 (Cont.) FCT_USC_Map EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_CBD_IMPACT_CATEGORY DETAIL.ASS_CBD.CP.IMPACT_CATEGORY	String	Read/Write	Contains the charge breakdown record impact category.
ASS_CBD_ZONE_ENTRY_NAME DETAIL.ASS_CBD.CP.ZONE_ENTRY_NAME	String	Read	Contains the zone name used for the charge packet.
ZONE_ENTRY_NAME DETAIL.ZONE_ENTRY_NAME	String	Read	Contains the zone name of the event.

Database Interface

The FCT_USC_Map module uses the following database tables:

- IFW_USC_MAP. This table stores mapping rules for usage scenario maps. You define the rules in Pricing Center. See "Setting Up Usage Scenario Mapping" in *BRM Setting Up Pricing and Rating*.
- IFW_USC_GROUP. This table stores USC group codes used for usage scenario mapping. See "Setting Up Usage Scenario Mapping" in *BRM Setting Up Pricing and Rating*.
- IFW_USAGETYPE. This table stores usage type codes used for usage scenario mapping. See "Setting Up Usage Scenario Mapping" in *BRM Setting Up Pricing and Rating*.

For information about the fields in database tables, see the documentation in *Pipeline_Home***database**.

Note: For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

FCT_Zone

The FCT_Zone module computes zones when you use Pipeline Manager only for zoning. See "About Setting Up Zones" in *BRM Setting Up Pricing and Rating*.

Dependencies

The FCT_Zone module requires a connection to the DAT_Zone module.

This module must run after FCT_Account.

For more information, see "Function Module Dependencies".

Registry Entries

Table 36–151 lists the FCT_Zone registry entries.

Table 36–151 FCT_Zone Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataModule	Specifies the connection to the DAT_Zone module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes
EdrZoneModel	Specifies the zone model which should be used for the zoning. You can use this entry in a semaphore file.	Yes

Sample Registry

```
Zoning
{
  ModuleName = FCT_Zone
  Module
  {
    Active = True
    DataModule = ifw.DataPool.ZoneDataModule
    EdrZoneModel = ZM_ADD
  }
}
```

Semaphore File Entries

Table 36–152 lists the FCT_Zone Semaphore file entries.

Table 36–152 FCT_Zone Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
EdrZoneModel	Specifies the zone model which should be used for the zoning.

Sample Semaphore File Entry

```
ifw.Pipelines.Functions.Processing.FunctionPool.Zoning.Module.EdrZoneModel = D2_FUN
```

EDR Container Fields

Table 36–153 lists the FCT_Zone EDR container fields.

Table 36–153 FCT_Zone EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging time stamp.
RETAIL_IMPACT_CATEGORY DETAIL.RETAIL_IMPACT_CATEGORY	String	Write	Contains the retail impact category.
WHOLESALE_IMPACT_CATEGORY DETAIL.WHOLESALE_IMPACT_CATEGORY	String	Write	Contains the wholesale impact category.
ZONE_DESCRIPTION DETAIL.ZONE_DESCRIPTION	String	Write	Contains the zone description for displaying on invoices.
ZONE_ENTRY_NAME DETAIL.ZONE_ENTRY_NAME	String	Write	Contains the destination description for displaying on invoices.
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code.
INTERN_A_NUMBER_ZONE DETAIL.INTERN_A_NUMBER_ZONE	String	Read	Contains the A number.
INTERN_B_NUMBER_ZONE DETAIL.INTERN_B_NUMBER_ZONE	String	Read	Contains the B number.
BDR_INTERN_ZONE_MODEL DETAIL.INTERN_ZONE_MODEL	Integer	Write	Contains the resulting zone model ID.
BDR_INTERN_APN_GROUP DETAIL.INTERN_APN_GROUP	String	Write	Contains the zone model related APN_ GROUP for use by the FCT_APN_Map module.

37

Pipeline Manager Data Modules

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager data modules.

DAT_AccountBatch

DAT_AccountBatch retrieves account data from the BRM database for the "DAT_ItemAssign", "FCT_Account", and "FCT_AccountRouter" modules.

See "Adding Customer Balance Impact Data to EDRs" in *BRM Setting Up Pricing and Rating* and "Using Pipeline Manager with Multiple Database Schemas".

This module also maintains a list of the accounts that are being rerated by the **pin_rerate** utility. This information is used by the batch rating pipeline to suspend incoming call detail records (CDRs) for those accounts while rerating is in progress. See "About Comprehensive Rerating Using pin_rerate" in *BRM Setting Up Pricing and Rating*.

Dependencies

This module requires connections to the following:

- BRM database.
- Pipeline Manager database.
- DAT_Listener module. See "DAT_Listener".
- DAT_PortalConfig module. See "DAT_PortalConfig".

Registry Entries

Table 37–1 lists the DAT_AccountBatch registry entries.

Table 37–1 DAT_AccountBatch Registry Entries

Entry	Description	Mandatory
AcceptLoginSearchFailure	If set to **True,** when a customer login number is not found in memory, the EDR will be accepted, the pipeline will continue processing the EDR, and a warning will be reported in the stream log. If set to **False** (the default), when a customer login number is not found in memory, the EDR will be set as invalid and rejected and a major error will be reported. **Important:** If the **UseAsRouter** registry entry is set to **True**, this registry entry is not used.	No
AccountLocks	Use this entry to tune performance by managing thread contention. Default = **10** See "Locking Objects during DAT_AccountBatch Processing" in *BRM System Administrator's Guide*.	No
AddAliasList	Specifies whether all alias names and logins are added to the EDR. Default = **False** **Important:** If the **UseAsRouter** registry entry is set to **True**, this registry entry is not used.	No
ClosedAccountDelay	Specifies to not load closed accounts. Also specifies the number of days prior to the current date for which closed accounts are not loaded. For example, if **ClosedAccountDelay** is set to **10** and the current date it June 20, accounts that were closed prior to June 10 are not loaded into memory. Default = **0** See "Specifying To Not Load Closed Accounts" in *BRM Setting Up Pricing and Rating*.	No
Connections	Specifies the number of connections to the database. This value must be at least the number of threads plus 1. Default = **5** See "Configuring Threads for DAT_BalanceBatch Connections" in *BRM System Administrator's Guide*.	No
EnrichRatingProductOnly	If set to **True**, only the purchased products whose service ID matches the service ID in the EDR being rated are added to the EDR. If set to **False**, the purchased products whose service type matches the service type in the EDR being rated are added to the EDR. **Note:** This entry is present for backward compatibility.	No
InfranetConnection	Specifies the connection to the BRM database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes

Table 37–1 (Cont.) DAT_AccountBatch Registry Entries

Entry	Description	Mandatory
InitialLoading	Specifies whether the initial loading of service and account data is performed. Otherwise, loading occurs while processing. Login objects are always loaded. Setting this entry to **False** enables the system to start faster. Default = **True** **Important:** If the **UseAsRouter** registry entry is set to **True**, this registry entry is not used. See "Specifying Whether To Load All Account Data" in *BRM Setting Up Pricing and Rating*.	No
IntegrateConnection	Specifies the connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
Listener	Specifies the connection to the DAT_Listener module. See "Configuring the DAT_Listener module" in *BRM Installation Guide* and "DAT_Listener".	Yes
LoadAccountForSharingOnly	Specifies whether Pipeline Manager can load serviceless accounts that are owners of resource sharing groups. Default = **False** See "About Serviceless Accounts as Charge Sharing Owners" in *BRM Managing Accounts Receivable*.	No
LoadLogins	Specifies whether the login is loaded in case of an existing alias list. When set to **True**, logins are loaded from both the PIN_FLD_LOGIN field and the PIN_FLD_ALIAS_LIST array. When set to **False**, logins are only loaded from the PIN_FLD_ALIAS_LIST array. Default = **False** when **UseAsRouter** is disabled. When **UseAsRouter** is enabled, then **LoadLogins** is always **True**.	No
LoadPercentage	Indicates the percentage of account POIDs to store locally when determining the account blocks for which each thread is responsible. Values must be greater than 0.000000 and less than or equal to 100.0. Default = **10.0** See "Configuring Threads for DAT_BalanceBatch Connections" in *BRM System Administrator's Guide*.	Yes
LogEvents	Specifies whether received events should be written to a log file. Use this entry to troubleshoot Pipeline Manager event handling. Default = **False** See "Using Events to Start External Programs" in *BRM System Administrator's Guide*.	No
LoginLocks	Use this entry to tune performance by managing thread contention. Default = **10** See "Locking Objects during DAT_AccountBatch Processing" in *BRM System Administrator's Guide*.	No

Table 37–1 (Cont.) DAT_AccountBatch Registry Entries

Entry	Description	Mandatory
PerThreadJobsCount	Specifies the number of jobs per thread. **Important:** Setting the number of jobs per thread to a large number can decrease performance because of the system overhead associated with creating too many jobs. (Typically, three to eight jobs per thread is optimal). If you want to adjust the number of accounts or balances per job, increase or decrease the number of threads. See "Improving DAT_AccountBatch and DAT_BalanceBatch Load Balancing" in *BRM System Administrator's Guide*.	Yes
PortalConfigDataModule	Specifies the connection to the DAT_PortalConfig module. This enables DAT_AccountBatch to retrieve business parameter settings from the DAT_PortalConfig module. See "Using Business Parameter Settings From The BRM Database" in *BRM System Administrator's Guide*.	Yes
ReadAccountBalances	Specifies whether to load account resource data. The data includes resource IDs, such as 840. If enabled, the RESOURCE_LIST field in the CUSTOMER_DATA block is populated with the resource IDs for that account. Default = **False** **Important:** If the **UseAsRouter** registry entry is set to **True**, this registry entry is not used.	No
ReadAllProducts	If set to **True,** all the purchased products for the account are added to the EDR. If set to **False**, only those purchased products matching the service types and event types of the CDR processed are added to the EDR.	No
ReadPlans	If set to **True**, the module loads plan IDs into memory when loading purchased products. During EDR processing, the list of plan IDs for an account is returned to the FCT_Account module. See "Setting Up Exclusion Rules for Usage Discounts".	No
ReadSystemProductFromMain	If set to **True,** the module retrieves the latest system products from the main tables and uses the start and end dates to validate when the product is in effect. If set to **False** (the default), the module retrieves the system product information from the audit tables and uses the purchase creation date to validate when the product is in effect. **Important:** If the **UseAsRouter** registry entry is set to **True**, this registry entry is not used.	No
RejectClosedAccounts	Specifies whether to reject CDRs for accounts that are closed. If set to **True**, all closed account information is loaded from the database. Any CDR with a timestamp later than the account's closed date is rejected. Default = **False**	No
RowFetchSize	Specifies the number of rows of data to retrieve from the BRM database. Use this entry for performance tuning. Default = **1000** See "Specifying How Much Account Data To Retrieve On Startup" in *BRM Setting Up Pricing and Rating*.	No

Table 37–1 (Cont.) DAT_AccountBatch Registry Entries

Entry	Description	Mandatory
ServiceLocks	Tunes performance by managing thread contention. Default = **10** See "Locking Objects during DAT_AccountBatch Processing" in *BRM System Administrator's Guide*.	No
ThreadAccountHashMapSize	Controls the size of the temporary hash map built by each thread for accounts. See "Setting the Hash Map Size for Threads" in *BRM System Administrator's Guide*. **Important:** ■ Changing the default system-calculated values for this entry is not recommended. ■ If the **UseAsRouter** registry entry is set to **True,** this registry entry is not used.	No
ThreadGroupSharingChargesHashMapSize	Controls the size of the temporary hash map built by each thread for loading charge share group data. The system-calculated default value might not be appropriate. See "Setting the Hash Map Size for Threads" in *BRM System Administrator's Guide*. **Important:** If the **UseAsRouter** registry entry is set to **True**, this registry entry is not used.	No
ThreadGroupSharingDiscountsHashMapSize	Controls the size of the temporary hash map built by each thread for loading discount sharing group data. The system-calculated default value might not be appropriate. See "Setting the Hash Map Size for Threads" in *BRM System Administrator's Guide*. **Important:** If the **UseAsRouter** registry entry is set to **True**, this registry entry is not used.	No
ThreadGroupSharingMonitorsHashMapSize	Controls the size of a temporary hash map constructed for GroupSharingProfile object storage during multi-thread DAT_Account initialization. Default = (TotalAccounts / NumThreads) * 0.10. The default value for this entry is appropriate in most cases. However, the value should be increased if you exceed an average of 4 GroupSharingMonitors for every 10 accounts.	No
ThreadGroupSharingProfilesHashMapSizes	Controls the size of the temporary hash map built by each thread for loading profile sharing group data. The system-calculated default value might not be appropriate. See "Setting the Hash Map Size for Threads" in *BRM System Administrator's Guide*. **Important:** If the **UseAsRouter** registry entry is set to **True**, this registry entry is not used.	No
ThreadLoginHashMapSize	Controls the size of the temporary hash map built by each thread for loading logins. The system-calculated default value is appropriate for most BRM implementations. See "Setting the Hash Map Size for Threads" in *BRM System Administrator's Guide*.	No

Table 37–1 (Cont.) DAT_AccountBatch Registry Entries

Entry	Description	Mandatory
Threads	Specifies the number of threads. Set this value to at least the number of CPUs in the system. Increasing the number of threads increases performance, up to a point. Specifying too many threads decreases performance. Default = **4** See "Configuring Threads for DAT_BalanceBatch Connections" in *BRM System Administrator's Guide*.	Yes
ThreadServiceHashMapSize	Controls the size of the temporary hash map built by each thread for loading services. The system-calculated default value is appropriate for most BRM implementations. See "Setting the Hash Map Size for Threads" in *BRM System Administrator's Guide*. **Important:** If the **UseAsRouter** registry entry is set to **True**, this registry entry is not used.	No
TimesTenEnabled	Specifies whether your BRM system uses IMDB Cache. Set this to **True** if your BRM system uses IMDB Cache DM. Set this to **False** if your BRM system uses Oracle DM.	No
UseAsRouter	If set to **True,** the module is used by the FCT_AccountRouter module to route EDRs to separate Pipeline Manager instances. See "Using Pipeline Manager with Multiple Database Schemas" and "FCT_AccountRouter". If set to **False** (the default), the module is used by the FCT_Account module. See "Adding Customer Balance Impact Data to EDRs" in *BRM Setting Up Pricing and Rating*. **Important:** If set to **True**, the following registry entries are not used: ■ AcceptLoginSearchFailure ■ AddAliasList ■ InitialLoading ■ ReadAccountBalances ■ ReadSystemProductFromMain ■ ThreadAccountHashMapSize ■ ThreadGroupSharingChargesHashMapSize ■ ThreadGroupSharingDiscountsHashMapSize ■ ThreadGroupSharingProfilesHashMapSizes ■ ThreadServiceHashMapSize ■ UseProductCreatedTime ■ UseLatestProductAndDiscount	No

Table 37–1 (Cont.) DAT_AccountBatch Registry Entries

Entry	Description	Mandatory
UseLatestProductAndDiscount	If set to **True**, the module retrieves the latest purchased product and discount information from the main tables and uses the start and end dates to validate when the product is in effect. If set to **False** (the default), the module retrieves the purchased product and discount information from the audit tables and uses the purchase creation date to validate when the product is in effect. See "Configuring Account Product Validity Checking for Backdated Events" in *BRM Setting Up Pricing and Rating*. **Important:** If the **UseAsRouter** registry entry is set to **True**, this registry entry is not used.	No
UseProductCreatedTime	If set to **True** (the default), the product is selected only if an event occurs after the product's created time (PIN_FLD_CREATED_T) and between its start and end times. If set to **False**, product validity is checked based only on the start and end times (PIN_FLD_START_T and PIN_FLD_END_T) of the product. See "Configuring Product Validity Checking" in *BRM Setting Up Pricing and Rating*. **Important:** If the **UseAsRouter** registry entry is set to **True**, this registry entry is not used.	No
UseProfileEffectiveTime	If set to **True** (the default), the module uses EFFECTIVE_T to determine the validity of the profile objects. If set to **False**, the module uses CREATED_T to determine the validity of the profile objects.	No

Sample Registry Entry

```
CustomerData
  {
    ModuleName = DAT_AccountBatch
    Module
    {
      IntegrateConnection  = ifw.DataPool.Login
      InfranetConnection   = ifw.DataPool.LoginInfranet
      LogEvents            = True
      Listener             = ifw.DataPool.Listener
      TimesTenEnabled      = False
      ReadAccountBalances  = True
      Threads              = 4
      Connections          =  5
      LoadPercentage       = 10.0
    }
  }
```

Semaphore File Entries

Table 37–2 lists the DAT_AccountBatch Semaphore file entries.

Table 37–2 DAT_AccountBatch Semaphore File Entries

Entry	Description
LogEvents	Specifies whether events should be stored in a log file. You can also use this entry in the startup registry. See "Using Events to Start External Programs" in *BRM System Administrator's Guide*.
PrintData	Reports the account data for all accounts. See "Getting Information about Loading Accounts" in *BRM Setting Up Pricing and Rating*.
PrintDataLogin	Reports the account data for a single account identified by the BRM login ID (usually the phone number). See "Getting Information about Loading Accounts" in *BRM Setting Up Pricing and Rating*.
PrintDataSamples	Reports the account data for a specified number of accounts, chosen randomly. See "Getting Information about Loading Accounts" in *BRM Setting Up Pricing and Rating*.
PrintAmtData	Prints in-memory data about the Account Migration Manager (AMM) to the specified log file. See "Migrating Accounts with the Pipeline Manager Running" in *BRM System Administrator's Guide*.
PrintAmtJobData	Prints in-memory data about one account migration job to the specified log file. See "Migrating Accounts with the Pipeline Manager Running" in *BRM System Administrator's Guide*.
RejectClosedAccounts	Rejects CDRs with a timestamp later than the account's closed date.
Reload	Reloads data from the Pipeline Manager database. See "Reloading Data into a Pipeline Manager Module" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry

```
ifw.DataPool.CustomerData.Module.Reload {}
```

For more information, see "Semaphore File Syntax" in *BRM System Administrator's Guide*.

Database Tables

The DAT_AccountBatch module uses the following database tables:

- IFW_CURRENCY
- IFW_REF_MAP
- IFW_SERVICE

For information about the fields in these database tables, see the documentation in *Pipeline_Home*/**database**. *Pipeline_Home* is the directory where you installed Pipeline Manager.

DAT_AccountRealtime

The DAT_AccountRealtime module provides customer data from the BRM database in a real-time discounting pipeline.

Note: Unlike the DAT_AccountBatch module, the DAT_AccountRealtime module does not load account data in memory when you start Pipeline Manager. Instead, it gets account data in real time from the BRM database by using the NET_EM module.

The DAT_AccountRealtime module gets data for the FCT_Discount module. For information about the FCT_Discount module, see "FCT_Discount".

Dependencies

The DAT_AccountRealtime requires the NET_EM module. It makes a connection to the NET_EM module automatically; you don't need to configure the connection.

Registry Entries

There are no registry entries for the DAT_AccountRealtime module. You only need to enter the module in the registry DataPool section.

Sample Registry Entry

```
CustomerData
  {
    ModuleName = Dat_AccountRealtime
    Module
    {
      #
    }
  }
```

Semaphore File Entries

DAT_AccountRealtime does not support semaphore updates.

DAT_BalanceBatch

The DAT_BalanceBatch module maintains balance information in the Pipeline Manager memory. It uses Account Synchronization to retrieve balance information from the BRM database. Data is stored in memory only, not in the database or in a file.

When reading balances and sub-balances from the database, the DAT_BalanceBatch module ignores balance monitor impacts.

See the following documents:

- Configuring Discounting Modules and Components
- FCT_Discount

Dependencies

Requires the following connections:

- Pipeline Manager database.
- BRM database.
- DAT_AccountBatch module. See "DAT_AccountBatch".

- DAT_Listener module. See "DAT_Listener".
- DAT_Discount module. See "DAT_Discount".
- DAT_PortalConfig module. See "DAT_PortalConfig".

Registry Entries

Table 37–3 lists the DAT_BalanceBatch registry entries.

Table 37–3 DAT_BalanceBatch Registry Entries

Entry	Description	Mandatory
AccountDataModule	Specifies the connection to the DAT_AccountBatch module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide* and "DAT_AccountBatch".	Yes
BalanceDirectory	Specifies the directory that contains data and transaction files.	No
BalanceLocks	Specifies the number of locks that can be acquired during processing. Must be a positive integer. Default = **100** **Important**: Setting this value too low may decrease pipeline throughput performance.	No
BalanceLockStatusLog	Specifies that when an event transaction is locked by an EDR transaction, it is logged to the process logger. Default = **False**	No
BalancesPerThreadJobsCount	Specifies the number of jobs per thread. **Important:** Setting the number of jobs per thread to a large number can decrease performance because of the system overhead associated with creating too many jobs. (Typically, three to eight jobs per thread is optimal). If you want to adjust the number of accounts or balances per job, you can do this by increasing or decreasing the number of threads. See "Improving DAT_AccountBatch and DAT_BalanceBatch Load Balancing" in *BRM System Administrator's Guide*.	Yes
BalanceTrace	Specifies whether to generate a balance trace file. **True** indicates that a balance trace file is generated. **False** indicates that a balance trace file is not generated. Default = **False**	No
CustomEvents	Lists custom business events that include balance data needed by Pipeline Manager. Custom events are defined in the Account Synchronization DM payload configuration file (**payloadconfig_ifw_sync.xml**). See "Configuring Custom Business Events For Pipeline Discounting" and "About Publishing Additional Business Events" in *BRM Developer's Guide*.	No
DiscountDataModule	Specifies the connection to the DAT_Discount module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide* and "DAT_Discount".	Yes

Table 37–3 (Cont.) DAT_BalanceBatch Registry Entries

Entry	Description	Mandatory
InfranetConnection	Specifies the database connection to the BRM database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
IntegrateConnection	Specifies the database connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
ListenerDataModule	Specifies the connection to the DAT_Listener module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide* and "DAT_Listener".	Yes
LoadPercentage	Specifies how much data to load from the BRM database before the process log outputs status information. For example, to output status after every 10% of the data is loaded, enter **10**. Default = **10** See "Configuring Threads For DAT_BalanceBatch Connections" in *BRM System Administrator's Guide*.	No
LogEvents	Specifies whether received events should be written to a log file. Use this entry to troubleshoot Pipeline Manager event handling. Default = **False** See "Using Events to Start External Programs" in *BRM System Administrator's Guide*.	No
LogTransactions	Specifies if the balances affected during the CDR processing are logged. Default = **False**	No
PortalConfigDataModule	Specifies the connection to the DAT_PortalConfig module. This enables DAT_BalanceBatch to retrieve business parameter settings from the DAT_PortalConfig module. See "Using Business Parameter Settings From The BRM Database" in *BRM System Administrator's Guide*.	Yes
RowFetchSize	Specifies the number of rows of balance data to load from the BRM database for each database retrieving. Default = **50**	No
SelectiveSubBalLoad	Specifies whether to selectively load non-currency sub-balances at Pipeline Manager startup. Default = **True** See "Specifying Which Non-Currency Sub-Balances to Load at Startup" in *BRM Setting Up Pricing and Rating*.	No
Synchronized	Specifies whether to allow the first transaction to process and to make other transactions wait in the queue. Default = **False**	No
ThreadHashMapSize	Specifies the size of the hash map in each thread used for loading balance data from the BRM database. Default = **1024** See "Configuring Single-Threaded or Multithreaded Operation" in *BRM System Administrator's Guide*.	No

Table 37–3 (Cont.) DAT_BalanceBatch Registry Entries

Entry	Description	Mandatory
Threads	Specifies the number of threads for loading the balance data from the BRM database. The number of threads must be smaller than or equal to the number of connections. Default = **4** See "Configuring Single-Threaded or Multithreaded Operation" in *BRM System Administrator's Guide*.	No
UseFlexibleConsumptionRule	Specifies whether to use the resource consumption rules defined at the plan level. **True**: Uses the consumption rules defined for each resource in a balance group. If a consumption rule is not defined, this module uses the rules defined in the **/config/beid** object. If a consumption rule isn't defined in a balance group or the **/config/beid** object, this module uses the rule defined in the **multi_bal** instance of the **/config/business_params** object. **False**: Uses the system-wide consumption rule defined in the **multi_bal** instance of the **/config/business_params** object only. Default = **True** See "How Batch Rating Applies Consumption Rules" in *BRM Setting Up Pricing and Rating*.	No
VirtualTime	Specifies whether this module uses system time or virtual time. Default = **False** Set to **True** *only* if you are performing tests and have used the **pin_virtual_time** utility to set a virtual time. If you set this entry to **True**, make sure you copy the **pin.conf** file from the *BRM_Home*/**sys**/**test** directory to the *Pipeline_Home* directory. *BRM_Home* is the directory where you installed BRM components. The **pin.conf** file contains this entry: **- - pin_virtual_time pin_virtual_time_file**	No

Sample Registry Entry

```
BalanceDataModule
  {
    ModuleName = DAT_BalanceBatch
    Module
    {
      IntegrateConnection = ifw.DataPool.Login
      InfranetConnection  = ifw.DataPool.LoginInfranet
      AccountDataModule   = ifw.DataPool.CustomerData
      ListenerDataModule  = ifw.DataPool.Listener
      DiscountDataModule  = ifw.DataPool.DiscountData
      BalanceDirectory    = ./samples/wireless/data/balance
      UseFlexibleConsumptionRule = True
      CustomEvents
      {
        CycleRollover20days
      }
    }
  }
```

Semaphore File Entries

Table 37–4 lists the DAT_BalanceBatch Semaphore file entries.

Table 37–4 DAT_BalanceBatch Semaphore File Entries

Entry	Description
BalanceGroupId	Specifies the ID field of the balance group POID entry. The balance data referenced by **BalanceGroupId** is written into the file specified by the **DataFileName** entry.
DataFileName	Specifies the file name that contains balance data. If the **BalanceGroupId** entry is not present, DAT_BalanceBatch writes all balance data in memory into the file.
LogEvents	Specifies whether events should be stored in a log file. You can also use this entry in the startup registry. See "Using Events to Start External Programs" in *BRM System Administrator's Guide*.
ReloadCreditThresholdParam	Reloads the value from the **CreditThresholdChecking** business parameter. See "Reloading data into a Pipeline Manager module" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry

```
ifw.DataPool.BalanceDataModule.Module.DataFileName = balData.txt
ifw.DataPool.BalanceDataModule.Module.BalanceGroupId = 70015
ifw.DataPool.BalanceDataModule.Module.LogEvents = True
ifw.DataPool.BalanceDataModule.Module.ReloadCreditThresholdParam{}
```

For more information, see "Semaphore file syntax" in *BRM System Administrator's Guide*.

DAT_BalanceRealtime

The DAT_BalanceRealtime module runs in a real-time discounting pipeline. It retrieves the current balance from the BRM database and supplies the data for real-time discounting.

> **Note:** Unlike the DAT_BalanceBatch module, the DAT_BalanceRealtime module does not load balance data in memory when you start Pipeline Manager. Instead, it gets balance data in real time from the BRM database by using the NET_EM module.

See the following documents:

- Configuring a Real-Time Discounting Pipeline
- FCT_Discount

Dependencies

The DAT_BalanceRealtime module requires the NET_EM module. It makes a connection to the NET_EM module automatically; you don't need to configure the connection.

Registry Entries

There are no registry entries for the DAT_BalanceRealtime module. You only need to enter the module in the registry DataPool section.

Sample Registry Entry

```
BalanceDataModule
  {
    ModuleName = DAT_BalanceRealtime
    Module
    {
      #
    }
  }
```

Semaphore File Entries

DAT_BalanceRealtime does not support semaphore updates.

DAT_Calendar

The DAT_Calendar module provides holiday calendar data for the FCT_MainRating module.

See the following documents:

- "Rating by date and time with Pipeline Manager" in *BRM Setting Up Pricing and Rating*
- FCT_MainRating

Dependencies

Requires a connection to the Database Connect (DBC) module. See "Database Connect (DBC)".

Registry Entries

Table 37–5 lists the DAT_Calendar registry entries.

Table 37–5 DAT_Calendar Registry Entries

Entry	Description	Mandatory
DataConnection	Specifies the database connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes

Sample Registry Entry

```
Calendar
{
  ModuleName = DAT_Calendar
  Module
  {
    DataConnection = ifw.DataPool.Login
  }
}
```

Semaphore File Entries

Table 37–6 lists the DAT_Calendar Semaphore file entries.

Table 37–6 DAT_Calendar Semaphore File Entries

Entry	Description
Reload	Reloads data from the Pipeline Manager database. See "Reloading data into a Pipeline Manager module" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry

```
ifw.DataPool.CalendarDataModule.Module.Reload {}
```

For more information, see "Semaphore file syntax" in *BRM System Administrator's Guide*.

Events

Table 37–7 lists the DAT_Calendar events.

Table 37–7 DAT_Calendar Events

Event Name	Trigger	Sender	Parameter
EVT_RELOAD_SUCCESSFUL	Data reload was successful.	DAT_Calendar	None
EVT_RELOAD_FAILED	Data reload failed.	DAT_Calendar	None

Database Tables

The DAT_Calendar module uses the following database tables:

- IFW_CALENDAR
- IFW_HOLIDAY

To enter data in these tables, use Pricing Center. See "Creating Holiday Calendars" in *BRM Setting Up Pricing and Rating*.

For information about the fields in database tables, see the documentation in *Pipeline_Home*/**database**.

DAT_ConnectionMonitor

This module creates and monitors the idle timeout period for each connection and maintains the state for each client.

Registry Entries

Table 37–8 lists the DAT_ConnectionMonitor registry entries.

Table 37–8 DAT_ConnectionMonitor Registry Entries

Entry	Description	Mandatory
KeepAliveInterval	The idle timeout value in milliseconds, which specifies how long to wait for a message from the client before sending a Device Watchdog Request (DWR) message to the client. Default is **30000**.	Yes
KeepAliveQueue	Specifies the pipeline queue to which the dummy EDR for the DWR message is sent.	Yes
ShutdownInterval	The idle timeout value in milliseconds, which specifies how long to wait before shutting down after sending a Disconnect-Peer-Request (DPR) message to the client. Default is **1000**.	No
Threads	Number of threads in the pool. Default is **1**.	Yes

Sample Registry Entry

```
ConnectionMonitor
{
  ModuleName = DAT_ConnectionMonitor
  Module
  {
    Threads = 1
    KeepAliveInterval = 30000
    ShutdownInterval = 1000
    KeepAliveQueue = ifw.IPCQueues.INOutputQueue

  }
}
```

Semaphore File Entries

DAT_ConnectionMonitor does not support semaphore updates.

DAT_ConnectionPool

DAT_ConnectionPool module has a set of configured Connection Manager (CM) connections, which the "FCT_Opcode" module uses to connect to the CM and call the appropriate opcode.

For each CM, the DAT_ConnectionPool module maintains a queue for spare connections, determined by the size of the queue.

The DAT_ConnectionPool module balances the load by distributing the required load among the pipelines using the FCT_Opcode module and when there is a CM failure, redistributes the load among the active CMs.

> **Note:** If a connection fails, the processing pipeline connects to the spare connection of the CM to which it initially connected; if the CM is down, it tries to use a connection from a different CM queue.

If an idle processing pipeline is connected to an inactive CM, The DAT_ConnectionPool module sets the connection status to inactive and updates the pipeline

recycle flag, so that when the pipeline starts processing a request it can connect to an active CM.

The DAT_ConnectionPool module initializes the Global Data Dictionary (GDD) during startup by accessing the database.

Note: If CMs are not available, it uses a file containing the data dictionary flist to initialize the GDD.

Registry Entries

Table 37–9 lists the DAT_ConnectionPool registry entries.

Table 37–9 DAT_ConnectionPool Registry Entries

Entry	Description	Mandatory
InfranetDataDictionaryFileName	The file containing the data dictionary. If a CM is not available, DAT_ConnectionPool uses this file to start and initialize the GDD. If you don't specify a file name, the DAT_ConnectionPool uses the default file, **./gddDataFile.dat**, where it stored the data dictionary flists at the initial startup.	No
IdleConnectionBuffer	Size of the queue for spare connections.	Yes
FullQueueTimeout	The interval, in seconds, in which the worker thread pings the queue to check if there is space available in the queue for a connection, when the queue is full. Default is 10 seconds.	No
EmptyQueueTimeout	The interval, in seconds, in which the pipeline thread pings the queue to check if there is a connection available in the queue, when the queue is empty. It can happen during startup of the pipeline or when the CM connection is not working as expected, for example, the CM times out. Default is 1 second.	No
InfranetPool	Specifies the CMs in the connection pool. For each CM in the pool, define the following entries: ■ Host name (**Host =** *CM1_host)* ■ Port number (**Port =***CM1_port)* ■ Login name and password for logging into BRM (**LoginName =** *root.0.0.0.1* and ■ **LoginPassword =***password)* ■ Whether to log debug messages (**Logging =** *False*. Values are **True** and **False**. The default is **False**.) ■ CM response timeout in milliseconds (**SocketTimeOut =** *30000*.)	Yes

Sample Registry Entry

```
DataPool
{

   CMConnectionPool
  {
    ModuleName = DAT_ConnectionPool
    Module
```

```
    {
      InfranetDataDictionaryFileName = File_with_DD_objects
      IdleConnectionBuffer
      {
        Size = 2
       }
      FullQueueTimeout = number_of_seconds
      EmptyQueueTimeout = number_of_seconds
      InfranetPool
     {
       CM1
         {
           Host = CM1_host
           Port = CM1_port
           LoginName = root.0.0.0.1
           LoginPassword = password
           Logging = True
           SocketTimeOut = 30000
         }
      CM2
         {
           Host = CM2_host
           Port = CM2_port
           LoginName = root.0.0.0.1
           LoginPassword = password
           Logging = True
           SocketTimeOut = 30000
          }
      }
   }
}
```

Semaphore File Entries

DAT_ConnectionPool does not support semaphore updates.

DAT_Currency

The DAT_Currency module converts currency symbols to numeric values and retreives resource rounding rules, using data from **/config/beid** objects in the BRM database. See "Setting up Pipeline Manager Resources" in *BRM Setting Up Pricing and Rating*.

Dependencies

Requires a connection to the BRM database.

Registry Entries

Table 37–10 lists the DAT_Currency registry entries.

Table 37–10 DAT_Currency Registry Entries

Entry	Description	Mandatory
DataConnection	Specifies the database connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
InfranetConnection	Specifies the database connection to the BRM database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
ReuseOnFailure	Specifies whether the module should continue to use the old data if the **Reload** command fails. If **True**, the old data is used. If the entry is not used, the default is **False**.	No

Sample Registry Entry

```
DAT_Currency
{
  ModuleName = DAT_Currency
  Module
  {
    ReuseOnFailure = TRUE
    InfranetConnection  = ifw.DataPool.LoginInfranet
    DataConnection = ifw.DataPool.Login
  }
}
```

Semaphore File Entries

Table 37–11 lists the DAT_Currency Semaphore file entries.

Table 37–11 DAT_Currency Semaphore File Entries

Entry	Description
Reload	Reloads data from the Pipeline Manager database. See "Reloading data into a Pipeline Manager module" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry

```
ifw.DataPool.CurrencyDataModule.Module.Reload ()
```

For more information, see "Semaphore file syntax" in *BRM System Administrator's Guide*.

DAT_Dayrate

The DAT_Dayrate module provides special day rate data for the FCT_Dayrate module.

See the following documents:

- "About Special Day Rates" in *BRM Setting Up Pricing and Rating*
- FCT_Dayrate

Dependencies

Requires a connection to the Pipeline Manager database.

Registry Entries

Table 37–12 lists the DAT_Dayrate registry entries.

Table 37–12 DAT_Dayrate Registry Entries

Entry	Description	Mandatory
Buffer	Specifies the size of the internal data buffer.	Yes
DataConnection	Specifies the database connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes

Sample Registry Entry

```
Dayrate
{
  ModuleName = DAT_Dayrate
  Module
  {
    DataConnection = ifw.DataPool.Login
    Buffer = 5000
  }
}
```

Semaphore File Entries

Table 37–13 lists the DAT_Dayrate Semaphore file entries.

Table 37–13 DAT_Dayrate Semaphore File Entries

Entry	Description
Reload	Reloads data from the Pipeline Manager database. See "Reloading Data into a Pipeline Manager Module" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry

```
ifw.DataPool.DayRateDataModule.Module.Reload {}
```

For more information, see "Semaphore File Syntax" in *BRM System Administrator's Guide*.

Events

Table 37–14 lists the DAT_Dayrate events.

Table 37–14 DAT_Dayrate Events

Event Name	Trigger	Sender	Parameter
EVT_RELOAD_SUCCESSFUL	Data reload was successful.	DAT_Dayrate	None
EVT_RELOAD_FAILED	Data reload failed.	DAT_Dayrate	None

Database Tables

The DAT_Dayrate module uses the following database tables:

- IFW_SPECIALDAYRATE
- IFW_SPECIALDAY_LNK

To enter data in these tables, use Pricing Center. See "Setting Up Global Special Day Rates" in *BRM Setting Up Pricing and Rating*.

For information about the fields in database tables, see the documentation in *Pipeline_Home*/**database**.

DAT_Discount

The DAT_Discount module provides discount model data for the FCT_Discount module. See "FCT_Discount".

Dependencies

Requires a connection to:

- The Pipeline Manager database.
- The BRM database.
- The DAT_AccountRealtime or DAT_AccountBatch module.
- The DAT_ModelSelector module.
- The DAT_PortalConfig module.

Registry Entries

Table 37–15 lists the DAT_Discount registry entries.

Table 37–15 DAT_Discount Registry Entries

Entry	Description	Mandatory
AccountDataModule	Specifies the connection to the DAT_AccountRealtime or DAT_AccountBatch module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes
InfranetConnection	Specifies the database connection to the BRM database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes

Table 37–15 (Cont.) DAT_Discount Registry Entries

Entry	Description	Mandatory
EvalScriptFiles	Specify name-value pairs for one or more iScript files. The name is a unique string used to identify the script if there is an error. The value is the relative or absolute path of the file. The iScript files specified in this entry contain functions that can be referenced via **EVAL** tokens in discount expressions. Any number of files can be specified.	No
IntegrateConnection	Specifies the database connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
PortalConfigDataModule	Specifies the connection to the DAT_PortalConfig module. This enables DAT_Discount to retrieve business parameter settings from the DAT_PortalConfig module. See "Using Business Parameter Settings From The BRM Database" in *BRM System Administrator's Guide*.	Yes

Sample Registry Entry

```
ifw
{ ....
  DataPool
  { ....
    DiscountModelDataModule
    {
      ModuleName = DAT_Discount
      Module
      {
        IntegrateConnection = ifw.DataPool.Login
        InfranetConnection = ifw.DataPool.LoginInfranet
        AccountDataModule = ifw.DataPool.CustomerData
        #Customizable iScript files supporting EVAL function
        EvalScriptFiles
        {
           scriptFile1 = ../iScriptLib/iScriptLib_Samples/ISC_GetMostCalledInfo.isc
           scriptFile2 = ../iScriptLib/iScriptLib_Samples/ISC_GetLastSixMonthCharge.isc
        }
      }
    }
  }
}
```

Semaphore File Entries

Table 37–16 lists the DAT_Discount Semaphore file entries.

Table 37–16 DAT_Discount Semaphore File Entries

Entry	Description
Reload	Reloads data from the Pipeline Manager database. See "Reloading data into a Pipeline Manager module" in *BRM System Administrator's Guide*. **Note:** Discounting data is reloaded only when the discount configuration contains a new non-currency resource.
ReloadEvalScripts	Reloads and recompiles the iScript files specified in the **EvalScriptFiles** registry entry.
DiscountModel	The discount models whose data you want written to the output. This entry is used for troubleshooting purposes and must be used in conjunction with the **DataFileName** semaphore. ■ **ALL**: Writes all discount codes (such as model codes, rule codes, step codes, and so on) and related configuration information for all discount models in your system. ■ *Discount_model_code*: Writes the discount codes and related configuration information associated with the specified discount model code (as entered in Pricing Center). **Note:** You cannot specify multiple discount model codes. You can specify only a single code or ALL.
DataFileName	Where discount model information should be written. This entry is used for troubleshooting purposes and must be used in conjunction with the **DiscountModel** semaphore. ■ To write the information to a file, specify a file name. By default, the file is created in the *Pipeline_Home* directory. ■ To write the information to the terminal, leave the value of this entry blank.

Sample Semaphore File Entry

- To reload data from the database and to reload and recompile iScript files:

```
ifw.DataPool.DiscountModelDataModule.Module.Reload {}
ifw.DataPool.DiscountModelDataModule.Module.ReloadEvalScripts=True
```

- To write all discount model configuration information to a file named **DiscountConfig.log**:

```
ifw.DataPool.DiscountModelDataModule.Module.DiscountCode = ALL
ifw.DataPool.DiscountModelDataModule.Module.DataFileName = DiscountConfig.log
```

- To write the configuration information for a discount model with the code **DM10%off** to the terminal:

```
ifw.DataPool.DiscountModelDataModule.Module.DiscountCode = DM10%OFF
ifw.DataPool.DiscountModelDataModule.Module.DataFileName {}
```

For more information, see "Semaphore file syntax" in *BRM System Administrator's Guide*.

Database Tables

The DAT_Discount module uses the following database tables:

- IFW_DISCOUNTMODEL
- IFW_DSCMDL_VER
- IFW_DSCMDL_CNF

- IFW_DSCTRIGGER
- IFW_DSCCONDITION
- IFW_DISCOUNTMASTER
- IFW_DISCOUNTDETAIL
- IFW_DISCOUNTRULE
- IFW_DISCOUNTSTEP
- IFW_DSCBALIMPACT

To enter data in these tables, use Pricing Center. See "Grouping Discount Components into Discount Models".

For information about the fields in database tables, see the documentation in *Pipeline_Home*/**database**.

Note: For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

DAT_ExchangeRate

The DAT_ExchangeRate module provides currency exchange rate data for the FCT_ExchangeRate module.

See the following documents:

- "Defining Currency Exchange Rates" in *BRM Setting Up Pricing and Rating*
- FCT_ExchangeRate

Dependencies

Requires a connection to the Pipeline Manager database.

Registry Entries

Table 37–17 lists the DAT_ExchangeRate registry entries.

Table 37–17 DAT_ExchangeRate Registry Entries

Entry	Description	Mandatory
DataConnection	Specifies the database connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
ReuseOnFailure	Specifies if the module should continue to use the old data if the **Reload** command fails. If **True**, the old data is used. If the entry is not used, the default is **False**.	No

Sample Registry Entry

```
ExchangeRateData
{
    ModuleName = DAT_ExchangeRate
    Module
    {
```

```
      DataConnection = ifw.DataPool.Login
      ReuseOnFailure = True
    }
}
```

Semaphore File Entries

Table 37–18 lists the DAT_ExchangeRate Semaphore file entries.

Table 37–18 DAT_ExchangeRate Semaphore File Entries

Entry	Description
Reload	Reloads data from the Pipeline Manager database. See "Reloading data into a Pipeline Manager module" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry

```
ifw.DataPool.ExchangeRateDataModule.Module.Reload {}
```

For more information, see "Semaphore file syntax" in *BRM System Administrator's Guide*.

Events

Table 37–19 lists the DAT_ExchangeRate events.

Table 37–19 DAT_ExchangeRate Events

Event Name	Trigger	Sender	Parameter
EVT_RELOAD_SUCCESSFUL	Data reload was successful.	DAT_ExchangeRate	None
EVT_RELOAD_FAILED	Data reload failed.	DAT_ExchangeRate	None

Database Tables

The DAT_ExchangeRate module uses the IFW_EXCHANGE_RATE database table. To enter data in this table, use Pricing Center. See "Defining Currency Exchange Rates" in *BRM Setting Up Pricing and Rating*.

For information about the fields in database tables, see the documentation in *Pipeline_Home*/**database**.

> **Note:** For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

DAT_InterConnect

The DAT_InterConnect module caches InterConnect and roaming related configuration data. This information is used by FCT_CarrierIcRating module.

> **Note:** Use the OpFlagExt iScript extension in iScripts to call this module directly to obtain the network operator taxation value flag.

See the following documents:

- "About Rating Roaming Events" in *BRM Configuring Roaming in Pipeline Manager*
- "About Processing Home Subscribers' Roaming Usage" in *BRM Configuring Roaming in Pipeline Manager*
- FCT_CarrierIcRating

Dependencies

Requires a connection to the Pipeline Manager database.

Registry Entries

Table 37–20 lists the DAT_InterConnect registry entries.

Table 37–20 DAT_InterConnect Registry Entries

Entry	Description	Mandatory
DataConnection	Specifies the database connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
LoadPoiAreas	If **True**, load data from the IFW_POIAREA_LNK table for reseller interconnection network models.	No
ReuseOnFailure	Specifies if the module should continue to use the old data if the **Reload** command fails. If **True**, the old data is used. If the entry is not used, the default is **False**.	Yes

Sample Registry Entry

```
InterConnect
{
  ModuleName = DAT_InterConnect
  Module
  {
    DataConnection = ifw.DataPool.Login
    ReuseOnFailure = False
  }
}
```

Semaphore File Entries

Table 37–21 lists the DAT_InterConnect Semaphore file entries.

Table 37–21 DAT_InterConnect Semaphore File Entries

Entry	Description
Reload	Reloads data from the Pipeline Manager database. See "Reloading data into a Pipeline Manager module" in *BRM System Administrator's Guide*.
LoadPoiAreas	Specifies if the module should load data from the IFW_POIAREA_LNK table for reseller interconnection network models.
ReuseOnFailure	Specifies if the module should continue to use the old data if the **Reload** command fails. If **True**, the old data is used. If the entry is not used, the default is **False**.

Sample Semaphore File Entry

```
ifw.DataPool.InterConnect.Module.Reload {}
```

```
ifw.DataPool.InterConnectDataModule.Module.ReuseOnFailure = True
ifw.DataPool.InterConnectDataModule.Module.LoadPoiAreas = True
```

For more information, see "Semaphore file syntax" in *BRM System Administrator's Guide*.

Database Tables

The DAT_InterConnect module uses the following database tables:

- IFW_NETWORKOPER
- IFW_NETWORKMODEL
- IFW_ICPRODUCT
- IFW_ICPRODUCT_RATE
- IFW_ICPRODUCT_GRP
- IFW_ICPRODUCT_CNF

Data for interconnect rating is stored in the following tables:

- IFW_SWITCH
- IFW_POI
- IFW_TRUNK
- IFW_TRUNK_CNF
- IFW_POIAREA_LNK

For information about the fields in database tables, see the documentation in *Pipeline_Home*/**database**.

To enter data in these tables, use Pricing Center.

Note: For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

DAT_ItemAssign

The DAT_ItemAssign module returns the item POID for an item tag to the FCT_ItemAssign and FCT_Billing Record modules. See:

- "Setting Up Batch Rating To Assign Items Based On Event Attributes" in *BRM Configuring and Running Billing*
- "Creating Custom Bill Items" in *BRM Configuring and Running Billing*
- FCT_Reject.

Dependencies

Requires a connection to the BRM database and the following modules:

- DAT_AccountBatch
- DAT_PortalConfig

Registry Entries

Table 37–22 lists the DAT_ItemAssign registry entries.

Table 37–22 DAT_ItemAssign Registry Entries

Entry	Description	Mandatory
AccountDataModule	Specifies the connection to the DAT_AccountBatch module. See "Connecting a Pipeline Manager Module to Another Module" in *BRM System Administrator's Guide* and "DAT_AccountBatch".	Yes
InfranetConnection	Specifies the connection to the BRM database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
ItemPoidReservedRangeUnitSize	Specifies the maximum number of POIDs to be reserved. Default =**10000**	No
PortalConfigDataModule	Specifies the connection to the DAT_PortalConfig module. This enables the DAT_ItemAssign module to retrieve business parameter settings from the DAT_PortalConfig module. See "Using Business Parameter Settings from the BRM Database" in *BRM System Administrator's Guide*.	Yes

Sample Registry Entry

```
Flexible item assignment data module

ItemAssignDataModule
{
  ModuleName = DAT_ItemAssign
  Module
  {
   InfranetConnection = ifw.DataPool.LoginInfranet
   AccountDataModule = ifw.DataPool.CustomerData
   ItemPoidReservedUnitSize = 10000
   PortalConfigDataModule = ifw.DataPool.PortalConfigDataModule
  }
}
```

Semaphore File Entries

Table 37–23 lists the DAT_ItemAssign Semaphore file entries.

Table 37–23 DAT_ItemAssign Semaphore File Entries

Entry	Description
PrintData	Generates a log file that contains the item tag-to-type mapping information. See "Verifying Item-Tag-to-Item-Type Mapping" in *BRM Configuring and Running Billing*.
Reload	Reloads data from the Pipeline Manager database. See "Reloading Data into a Pipeline Manager Module" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry

```
ifw.DataPool.ItemAssignDataModule.Module.PrintData=TagTypeMap.txt
ifw.dataPool.ItemAssignDataModule.Module.Reload {}
```

For more information, see "Semaphore File Syntax" in *BRM System Administrator's Guide*.

DAT_Listener

The DAT_Listener module dequeues business events from a database queue and then sends the data to the DAT_AccountBatch and DAT_Discount modules.

The DAT_Listener module also posts acknowledgment events to the acknowledgment queue in response to business events sent by **pin_rerate**. See "About comprehensive rerating using pin_rerate" in *BRM Setting Up Pricing and Rating*.

DAT_Listener module controls whether the Pipeline Manager processes business events or CDRs by interleaving the two processes. You can configure the DAT_Listener module for concurrent or interleaved processing. See "Configuring Interleaved Processing" in *BRM Installation Guide*.

See the following documents:

- "About Sending Account Data to Pipeline Manager" in *BRM Installation Guide*
- DAT_AccountBatch
- DAT_Discount

Dependencies

Requires a connection to the database containing the account synchronization queue.

The Listener section of the registry file must be listed after the Pipeline Manager and BRM database connection sections. Otherwise, the Pipeline Manager fails to start.

Registry Entries

Table 37–24 lists the DAT_Listener registry entries.

Table 37–24 DAT_Listener Registry Entries

Entry	Description	Mandatory
AckQueueNameAMM	Specifies the name of the acknowledgment queue for posting AMM-related acknowledgment events.	Yes This entry is mandatory for AMM operations. For more information, see "About Notifying AMM About EDR Processing" in *BRM System Administrator's Guide*.
AckQueueName	Specifies the name of the acknowledgment queue for posting rerating related acknowledgment events. Default = **RERATING_ACK_QUEUE**	Yes This entry is mandatory for rerating using **pin_rerate**. See "About comprehensive rerating using pin_rerate" in *BRM Setting Up Pricing and Rating*.
BatchSize	The maximum number of events to be dequeued in each dequeuing operation. Default = 10	No
EventsPath	Specifies the directory location of the file that stores the event information retrieved by DAT_Listener if the LogEvents entry is set to **True**. Default location is the directory where the ifw is launched.	No
EventsPrefix	Specifies the prefix of the name of the file that stores the event information retrieved by DAT_Listener if the LogEvents entry is set to **True**. Default =**listenerLog**	No

EventThreadAllocation	Defines the number of threads (in addition to one default thread) to use for dequeuing specific business events. For example:
	`EventThreadAllocation`
	`{`
	`  RecycleRequest = 1`
	`  OpenNewActgCycle = 2`
	`}`
	uses 4 threads: one for RecycleRequest business events, two for OpenNewActgCycle business events, and one default thread for dequeuing all other types of business events.
	For more information, see "About Using Multiple Threads to Dequeue Events" in *BRM Installation Guide*.
		No

InfranetConnection	Specifies the connection to a database with the account synchronization queue. Default = **ifw.DataPool.LoginInfranet**	Yes

Table 37–24 (Cont.) DAT_Listener Registry Entries

Entry	Description	Mandatory
LogEvents	Specifies whether received events should be written to a log file. Default = **False**	No
NumOfRetries	Specifies the number of times the DAT_Listener module retries to connect to the database queue. Default = 10	No
QueueLibrary	Specifies whether the DAT_Listener module is configured for Oracle AQ. Set QueueLibrary to **OracleQueue**.	Yes
QueueName	Specifies the name of the database queue from which the DAT_Listener module retrieves events. Default = **IFW_SYNC_QUEUE**	Yes
RetryInterval	Specifies the time in seconds that the DAT_Listener module waits before trying to reconnect to the database specified in **InfranetConnection**. Default = **5**	No

Registry Entries for Interleaved Processing

The following are registry entries used for interleaved processing.

Important: The default values for interleaved processing are also the minimum required values. If you specify a value less than the default for any entry, that value is ignored and the minimum default value is used.

Table 37–25 lists the DAT_Listener registry entries for interleaved processing.

Table 37–25 DAT_Listener Registry Entries for Interleaved Processing

Entry	Description	Mandatory
CheckInterval	Specifies (in seconds) how frequently the DAT_Listener module checks the number of events waiting in the queue. If this entry is not present, the default frequency check is used. **Important:** This entry takes precedence over **MaxNumEvents**, **MinNumEvents**, **MaxEventProcessTime**, and **MaxCDRProcessTime**. For example, if **MaxEventProcessTime** is set to 3600 seconds, and **CheckInterval** is set to 7200 seconds, events are processed for 7200 seconds. Default = **60**	No
EnableInterLeaving Statistics	Specifies whether to log only interleaving statistical data. If set to **False**, all processing messages are logged. Default = **False**	No
InterleavingReqd	Specifies whether interleaved processing is enabled: **True** = Enabled **False** = Not enabled When set to **False** or not specified, interleaved processing is not performed; CDRs and events are processed simultaneously. Default = **False**	No
MaxCDRProcessTime	Specifies the maximum number of seconds that CDRs are processed. When the pipeline has been processing CDRs for this amount of time, the DAT_Listener module stops CDR processing and starts business event processing regardless of how many business events are in the queue. Default and minimum allowed = **300**	If **MaxEventProcessTime** is specified and **InterleavingReqd** is set to TRUE, yes. Otherwise, no.
MaxEventProcessTime	Specifies the maximum number of seconds that business events are processed. When the pipeline has been processing business events for this amount of time, the DAT_Listener module stops business event processing and starts CDR processing regardless of how many business events are in the queue. Default and minimum allowed = **60**	If **MaxCDRProcessTime** is specified and **InterleavingReqd** is set to TRUE, yes. Otherwise, no.

Table 37–25 (Cont.) DAT_Listener Registry Entries for Interleaved Processing

Entry	Description	Mandatory
MaxNumEvents	Specifies the maximum number of business events allowed in the queue. When the number of events in the queue reaches or exceeds this amount, DAT_Listener stops pipeline CDR processing and starts business event processing. Default and minimum allowed = **900**	Yes, if **InterleavingReqd** is set to **True**. Otherwise, no. Requires that you also specify **MinNumEvents** and **MaxNumEvents** is greater than **MinNumEvents**.
MinNumEvents	Specifies the minimum number of business events allowed in the queue. When the number of events in the queue reaches or drops below this amount, the DAT_Listener stops business event processing and starts CDR processing. Default and minimum allowed = **300**	Yes, if **InterleavingReqd** is set to **True**. Otherwise, no.
ProcessAllEvents	Specifies whether to process all business events in the queue when Pipeline Manager is started: **True** = Processes all business events in the queue before activating interleaved processing. **False** = Interleaved processing is activated at startup. Business events are processed according to the interleaving settings. If set to **True** at startup, after processing all business events, this entry is reset to **False**. To process all business events at startup, you must reset this entry to **True** each time you restart Pipeline Manager. Default = **False**	No

Sample Registry Entry

```
Listener
{
    ModuleName = DAT_Listener
    Module
    {
       InfranetConnection = ifw.DataPool.LoginInfranet
       QueueLibrary = OracleQueue
       QueueName = IFW_SYNC_QUEUE
       NumOfRetries = 1
       RetryInterval = 5
       LogEvents = TRUE

       InterleavingReqd = true
       MaxNumEvents = 900
       MinNumEvents = 300
       CheckInterval = 60
       EnableInterLeavingStatistics = false
       ProcessAllEvents = true
       MaxEventProcessTime = 60
       MaxCDRProcessTime = 300
    }
}
```

Semaphore File Entries

Table 37–26 lists the DAT_Listener Semaphore file entries.

Table 37–26 DAT_Listener Semaphore File Entries

Entry	Description
AckQueueNameAMM	Specifies the name of the acknowledgment queue for posting AMM-related acknowledgment events. To add or modify this entry without having to stop the pipeline, use the **Disconnect** semaphore to disconnect the Listener before setting it. After specifying the queue name, use the **Connect** semaphore to reconnect the Listener to the pipeline for the new value to take effect.
CheckInterval	Specifies (in seconds) how frequently the DAT_Listener module checks the number of events waiting in the queue. If this entry is not present, the default frequency check is used.
Connect{}	Reconnects the DAT_Listener module event dequeuing threads to the Account Synchronization queue.
Disconnect{}	Disconnects the DAT_Listener module event dequeuing threads from the Account Synchronization queue. The module checks the dequeuing threads before disconnecting them: ■ If a thread is in the middle of processing an event, the module waits until the pipeline finishes processing events, and then suspends and disconnects the thread from the queuing database. ■ If a thread *is not* in the middle of processing an event, the module suspends and disconnects the thread from the queuing database.
EnableDequeueStatistics	Specifies whether to log dequeue statistics in the process log. **TRUE** = Log dequeue statistics **FALSE** = Do not log dequeue statistics (Default) **Note:** When set to **TRUE**, the size of the process log increases. Additionally, there may be a performance impact due to file input and output processing. Use this entry for diagnostic purposes only and should not be used otherwise.
EnableInterLeavingStatistics	Specifies whether to log only interleaving statistical data. If set to **False**, all processing messages are logged.
LogEvents	Specifies whether received events should be written to a log file. Default = **False**
MaxCDRProcessTime	Specifies the maximum number of seconds that CDRs are processed. When the pipeline has been processing CDRs for this amount of time, the DAT_Listener module stops CDR processing and starts business event processing regardless of how many business events are in the queue. Required when **MaxEventProcessTime** is specified. Requires that you also specify **MaxNumEvents**, **MinNumEvents**, and **MaxEventProcessTime**.

Table 37–26 (Cont.) DAT_Listener Semaphore File Entries

Entry	Description
MaxEventProcessTime	Specifies the maximum number of seconds that business events are processed. When the pipeline has been processing business events for this amount of time, the DAT_Listener module stops business event processing and starts CDR processing regardless of how many business events are in the queue. Required when **MaxCDRProcessTime** is specified. Requires that you also specify **MaxNumEvents**, **MinNumEvents**, and **MaxCDRProcessTime**.
MaxNumEvents	Specifies the maximum number of business events allowed in the queue. When the number of events in the queue reaches this amount, the DAT_Listener module stops pipeline CDR processing and starts business event processing. Required when **MinNumEvents**, **MaxEventProcessTime**, or **MaxCDRProcessTime** is specified. Requires that you also specify **MinNumEvents**.
MinNumEvents	Specifies the minimum number of business events allowed in the queue. When the number of events in the queue reaches this amount, the DAT_Listener module stops business event processing and starts CDR processing. Required when **MaxNumEvents**, **MaxEventProcessTime**, or **MaxCDRProcessTime** is specified. Requires that you also specify **MaxNumEvents**.

Sample Semaphore File Entry

```
ifw.DataPool.Listener.Module.CheckInterval=180
ifw.DataPool.Listener.Module.EnableInterLeavingStatistics=true
ifw.DataPool.Listener.Module.Disconnect{}
ifw.DataPool.Listener.Module.Connect{}
```

For more information, see "Semaphore file syntax" in *BRM System Administrator's Guide*.

DAT_ModelSelector

When a model selector is used to rate or discount an EDR, the DAT_ModelSelector module evaluates the model selector rules to determine the correct price or discount model. The rules are evaluated in the order they are ranked in the model selector.

The following rating and discounting modules get the model information from DAT_ModelSelector:

- FCT_MainRating gets the price model from DAT_ModelSelector. See "FCT_MainRating".
- FCT_DiscountAnalysis gets the discount model from DAT_ModelSelector. See "FCT_DiscountAnalysis".

See the following documents:

- "About Price Model Selectors" in *BRM Setting Up Pricing and Rating*
- About Discount Model Selectors

Dependencies

Requires a connection to the Database Connect (DBC) module. See "Database Connect (DBC)".

The module uses event notification to refresh customized product data. You must configure a connection to DAT_Listener if you plan to use this feature. See "DAT_Listener".

Registry Entries

Table 37–27 lists the DAT_ModelSelector registry entries.

Table 37–27 DAT_ModelSelector Registry Entries

Entry	Description	Mandatory
IntegrateConnection	Specifies the connection to the Pipeline Manager database. This typically points to the login registry section. For example: `IntegrateConnection = ifw.DataPool.Login` See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
ListenerDataModule	Specifies the connection to the DAT_Listener module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide* and "DAT_Listener".	No

Sample Registry Entry

```
ModelSelectorDataModule
    {
      ModuleName = DAT_ModelSelector
      Module
      {
     ListenerDataModule  = ifw.DataPool.Listener
        IntegrateConnection = ifw.DataPool.Login
        LogEvents           = True
      }
```

Semaphore File Entries

Table 37–28 lists the DAT_ModelSelector Semaphore file entries.

Table 37–28 DAT_ModelSelector Semaphore File Entries

Entry	Description
Reload	Reloads data from the Pipeline Manager database. See "Reloading data into a Pipeline Manager module" in *BRM System Administrator's Guide*.
DiscountModel	The discount models whose data you want written to the output. This entry is used for troubleshooting purposes and must be used in conjunction with the **DataFileName** semaphore. ■ **ALL**: Writes all discount codes (such as model codes, rule codes, step codes, and so on) and related configuration information for all discount models in your system. ■ *Discount_model_code*: Writes the discount codes and related configuration information associated with the specified discount model code (as entered in Pricing Center). **Note:** You cannot specify multiple discount model codes. You can specify only a single code or **ALL**.
DataFileName	Where discount model information should be written. This entry is used for troubleshooting purposes and must be used in conjunction with the **DiscountModel** semaphore. ■ To write the information to a file, specify a file name. By default, the file is created in the *Pipeline_Home* directory. ■ To write the information to the terminal, leave the value of this entry blank.

For more information, see "Semaphore file syntax" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry

To reload data from the database and to reload and recompile iScript files:

```
ifw.DataPool.ModelSelectorDataModule.Module.Reload {}
```

To write all model selector configuration information to a file named **ModelSelectorConfig.log**:

```
ifw.DataPool.ModelSelectorDataModule.Module.ModelSelectorCode = ALL

ifw.DataPool.ModelSelectorDataModule.Module.DataFileName = ModelSelectorConfig.log
```

To write the configuration information for a model selector with the code **DMS10%off** to the terminal:

```
ifw.DataPool.ModelSelectorDataModule.Module.ModelSelectorCode = DMS10%off
ifw.DataPool.ModelSelectorDataModule.Module.DataFileName {}
```

Database Tables

The DAT_ModelSelector module uses the following database tables:

- IFW_MODEL_SELECTOR. This table stores all model selector information in the Pipeline Manager database. It has a type field to indicate whether a model selector is for discounting or rating.

  To enter data in this table, use Pricing Center. See "About Price Model Selectors" in *BRM Setting Up Pricing and Rating* and "About Discount Model Selectors".

- IFW_SELECTOR_RULESET. This table maps model selector rules to specific model selectors. Rules associated with a model selector are ranked in order of priority. To enter data in this table, use Pricing Center. See "About Price Model

Selectors" in *BRM Setting Up Pricing and Rating* and "About Discount Model Selectors".

- IFW_SELECTOR_RULE. This table stores information for each model selector rule, including the code, name, and rule links. To enter data in this table, use Pricing Center. See "About Price Model Selectors" in *BRM Setting Up Pricing and Rating* and "About Discount Model Selectors".
- IFW_SELECTOR_RULE_LNK. This table maps a model selector rule to its detail or block. To enter data in this table, use Pricing Center. See "About Price Model Selectors" in *BRM Setting Up Pricing and Rating* and "About Discount Model Selectors".
- IFW_SELECTOR_DETAIL. This table stores each model selector's rule details; the EDR field and value. To enter data in this table, use Pricing Center. See "About Price Model Selectors" in *BRM Setting Up Pricing and Rating* and "About Discount Model Selectors".
- IFW_SELECTOR_BLOCK. This table stores block information for a model selector rule. This table is for future use.
- IFW_SELECTOR_BLOCK_LNK. This table maps a block to a selector detail or to another block. This table is for future use.

For information about the fields in database tables, see the documentation in *Pipeline_Home*/**database**.

Note: For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

DAT_NOSP

The DAT_NOSP module provides data for mapping network source and destinations to new values for the FCT_NOSP module, used for multi-segment rating.

See the following documents:

- Identifying the Network Operator/Service Provider
- About Multi-Segment Rating
- FCT_NOSP

Dependencies

DAT_NOSP module supports both file and database option.

If configured to get data from the database, the DAT_NOSP module requires a connection to the Pipeline Manager database.

Registry Entries

Table 37–29 lists the DAT_NOSP registry entries.

Table 37–29 DAT_NOSP Registry Entries

Entry	Description	Mandatory
DataConnection	Specifies the database connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes, if the data is stored in the database. Otherwise is not used.
FileName	Specifies the path and file name of the initialization file. See "Creating an NO/SP Data File". You can use this entry in a semaphore file.	Yes, if the data is stored in a file. Otherwise is not used.
ReuseOnFailure	Specifies if the module should continue to use the old data if the **Reload** command fails. If **True**, the old data is used. If the entry is not used, the default is **False**.	Yes
Source	Specifies where the data is stored: ■ **File** ■ **Database**	Yes

Sample Registry Entry for the Database Interface

```
NospData
{
  ModuleName = DAT_NOSP
  Module
  {
    Source = Database
    DataConnection = ifw.DataPool.Login
    ReuseOnFailure = FALSE
  }
}
```

Sample Registry Entry for the File Interface

```
NOSP
{
  ModuleName = DAT_NOSP
  Module
  {
    ReuseOnFailure = FALSE
    Source = File
    FileName = ./cfg/NOSP_Config1.dat
  }
}
```

Format of the file:

```
RANK;OLD_SOURCE;OLD_DESTINATION;A_PREFIX;NEW_SOURCE;NEW_DESTINATION;
```

For example,

```
ALL_RATE;1;ABC;BCD;0987;XYZ;YZA;
```

Semaphore File Entries

Table 37–30 lists the DAT_NOSP Semaphore file entries.

Table 37–30 DAT_NOSP Semaphore File Entries

Entry	Description
FileName	Specifies the path and file name of the initialization file.
Reload	Reloads data from the Pipeline Manager database. Only used if data is stored in the database. See "Reloading data into a Pipeline Manager module" in *BRM System Administrator's Guide*.

For more information, see "Semaphore file syntax" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry for the Database Interface

```
ifw.DataPool.NOSP.Module.Reload {}
```

Sample Semaphore File Entry for the File Interface

```
ifw.DataPool.NOSP.Module.FileName = ./cfg/NOSP_Config2.dat
```

Database Tables

The DAT_NOSP module uses the following tables:

- IFW_NOSP
- IFW_GROUP

For information about the fields in database tables, see the documentation in *Pipeline_Home*/**database**.

DAT_NumberPortability

The DAT_NumberPortability module provides number portability data to the FCT_NumberPortability module.

See the following documents:

- "Managing number portability" in *BRM Telco Integration*
- FCT_NumberPortability

Registry Entries

Table 37–31 lists the DAT_NumberPortability registry entries.

Table 37–31 DAT_NumberPortability Registry Entries

Entry	Description	Mandatory
CountryCode	Specifies the country code, for example **49** for Germany. This is needed for normalization of CLIs. See "Configuring Normalization for Number Portability".	Yes
FileName	Specifies the name of the number portability file. See "Creating a Number Portability Data File".	Yes
InternationalAccessCode	Specifies the international access code. This is needed for normalization of CLIs. Default = 00 See "Configuring Normalization for Number Portability".	No
InternationalAccessCodeSign	Specifies the international access code sign. This is needed for normalization of CLIs. Default = + See "Configuring Normalization for Number Portability".	No
NationalAccessCode	Specifies the national access code, for example **0** for Germany. This is needed for normalization of CLIs. See "Configuring Normalization for Number Portability".	Yes
ReuseOnFailure	Specifies if the module should continue to use the old data if the **Reload** command fails. If **True**, the old data is used. If the entry is not used, the default is **False**.	No
SearchMethod	Specifies which search method to use. ▪ **0** specifies to use best match. ▪ **1** specifies to use exact match. ▪ **2** specifies to use first prefix match. Default = **0** See "Configuring Number Portability Search".	No

Sample Registry Entry

```
NumberPortabilityData
{
  ModuleName = DAT_NumberPortability
  Module
  {
    FileName = ./data/primary_np.data
    CountryCode = 49
    NationalAccessCode = 0
    SearchMethod = 0
  }
}
```

Semaphore File Entries

Table 37–32 lists the DAT_NumberPortability Semaphore file entries.

Table 37–32 DAT_NumberPortability Semaphore File Entries

Entry	Description
AdditionalNumPortData	Specifies the name of the ASCII file that contains the newly ported numbers to reload. **Important:** This parameter must not be used with the Reload semaphore entry. Otherwise, an error message is logged and nothing is updated.
PrintData	Specifies the name of the ASCII file in which to print all newly ported numbers. Important: If this entry is specified with the **Reload** or **AdditionalNumPortData** semaphore entries, the **PrintData** entry is processed last.
Reload	Reloads data from the number portability data file. See "Reloading data into a Pipeline Manager module" in *BRM System Administrator's Guide*. Important: You cannot use this entry at the same time as the **AdditionalNumPortData** semaphore entry.

Sample Semaphore File Entry

```
ifw.DataPool.NumberPortability.Module.Reload {}
ifw.DataPool.NumberPortability.Module.AdditionalNumPortData=./data/primary_np.data
ifw.DataPool.NumberPortability.Module.PrintData=./data/records.nport.dump
```

For more information, see "Semaphore file syntax" in *BRM System Administrator's Guide*.

Events

Table 37–33 lists the DAT_NumberPortability Events.

Table 37–33 DAT_NumberPortability Events

Event Name	Trigger	Sender	Parameter
EVT_ADD_NUM_PORT_DATA_SUCCESSFUL	Adding new number Portability data succeeded.	DAT_NumberPortability	None
EVT_ADD_NUM_PORT_DATA_FAILED	Adding new number portability data failed.	DAT_NumberPortability	None

DAT_PortalConfig

The DAT_PortalConfig module loads data from the **/config/event_order_criteria**, **/config/business_params**, and **/config/credit_profile** objects in the BRM database.

For more information, see the following:

- "About Automatic Rerating of Out-Of-Order Events" in *BRM Setting Up Pricing and Rating*
- "Using Business Parameter Settings From The BRM Database" in *BRM System Administrator's Guide*
- "About credit limit and threshold checking during batch rating" in *BRM Managing Customers*

Dependencies

This module requires a connection to the Database Connect (DBC) module. See "Database Connect (DBC)".

Important: Due to the dependency of other data modules on DAT_PortalConfig, the DAT_PortalConfig registry entries must appear before all other data module entries in the registry file.

Registry Entries

Table 37–34 lists the DAT_PortalConfig registry entries.

Table 37–34 DAT_PortalConfig Registry Entries

Entry	Description	Mandatory
InfranetConnection	Specifies the connection to the DBC module.	Yes

Sample Registry Entry

```
PortalConfigDataModule
{
  ModuleName       = DAT_PortalConfig
  Module
  {
    InfranetConnection = ifw.DataPool.LoginInfranet
  }
}
```

Semaphore File Entries

Table 37–35 lists the DAT_PortalConfig Semaphore file entries.

Table 37–35 DAT_PortalConfig Semaphore File Entries

Entry	Description
CBPPrintData	Prints the /config/business_params data stored in the DAT_PortalConfig module's memory. If a filename is not provided, the module dumps the data into a file named **DefaultCBPDataFile**_*Timestamp*.**lst**. See "Printing business parameter settings stored in DAT_PortalConfig memory" in *BRM System Administrator's Guide*.
CreditProfilePrintData	Prints the **/config/credit_profile** data stored in the DAT_PortalConfig module's memory. If a filename is not provided, the module dumps the data into a file named **DefaultConfigCreditProfileDataFile**_*Timestamp*.**lst**. See "Printing business parameter settings stored in DAT_PortalConfig memory" in *BRM System Administrator's Guide*.
CBPReload	Reloads /config/business_params data. See "Refreshing business parameter settings stored in DAT_PortalConfig memory" in *BRM System Administrator's Guide*.
CreditProfileReload	Reloads **/config/credit_profile** data.

Table 37–35 (Cont.) DAT_PortalConfig Semaphore File Entries

Entry	Description
PrintData	Prints all the data stored in the DAT_PortalConfig module's memory. If a filename is not provided, the module dumps the data to the standard output console.
OODReload	Reloads **/config/event_order_criteria** data.
OODPrintData	Prints the **/config/event_order_criteria** data stored in the DAT_PortalConfig module's memory. If a filename is not provided, the module dumps the data into a file named **DefaultOODDataFile**_*Timestamp*.**lst**.

Sample Semaphore File Entry

```
ifw.DataPool.PortalConfigDataModule.Module.CreditProfile.Reload{}
ifw.DataPool.PortalConfigDataModule.Module.Reload{}
ifw.DataPool.PortalConfig.Module.CBPPrintData=BRM/config/prntCBPdata
ifw.DataPool.PortalConfig.Module.OODPrintData=BRM/config/prntOODdata
```

Events

Table 37–36 lists the DAT_PortalConfig events.

Table 37–36 DAT_PortalConfig Events

Event Name	Trigger	Sender	Parameter
EVT_RELOAD_SUCCESSFUL	Reload was successful.	DAT_PortalConfig	None
EVT_RELOAD_FAILED	Reload failed.	DAT_PortalConfig	None

Database Tables

The DAT_PortalConfig module uses the following database tables:

- CONFIG_T
- CONFIG_EVENT_ORDER_CRITERIA_T

DAT_PrefixDesc

The DAT_PrefixDesc module provides data for mapping phone number prefixes to descriptions, used by the FCT_PrefixDesc module.

See the following documents:

- Creating Call Destination Descriptions
- FCT_PrefixDesc

Dependencies

If data is stored in the database, the DAT_PrefixDesc module requires a connection to the Pipeline Manager database.

Registry Entries

Table 37–37 lists the DAT_PrefixDesc registry entries.

Table 37–37 DAT_PrefixDesc Registry Entries

Entry	Description	Mandatory
CLIBase	Specifies if the zone tree values should be hexadecimal or decimal. You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes, if the data is stored in the database. Otherwise is not used.
PrefixDesc.File	Specifies the file prefix for the files that contain prefix descriptions. See "Creating a Prefix/Description Data File". You can use this entry in a semaphore file.	Yes, if the data is stored in a file. Otherwise is not used.
ReuseOnFailure	Specifies if the module should continue to use the old data if the **Reload** command fails. If **True**, the old data is used. If the entry is not used, the default is **False**.	Yes
Source	Specifies where the module gets data: ■ **File** ■ **Database**	Yes

Sample Registry Entry for the Database Interface

```
PrefixDescDataModule
{
  ModuleName = DAT_PrefixDesc
  Module
  {
    Source = Database
    DataConnection = ifw.DataPool.Login
    ReuseOnFailure = false
    CLIBase = 10
  }
}
```

Sample Registry Entry for the File Interface

```
PrefixDescData
{
  ModuleName = DAT_PrefixDesc
  Module
  {
    Source = File
    ReuseOnFailure = false
    CLIBase = 10
    PrefixDesc
    {
      File = ../daten/forgn_names.dat
      File = ../daten/onkz_names.dat
    }
  }
}
```

Semaphore File Entries

Table 37–38 lists the DAT_PrefixDesc Semaphore file entries.

Table 37–38 DAT_PrefixDesc Semaphore File Entries

Entry	Description
CLIBase	Specifies if the zone tree values should be hexadecimal or decimal. Valid values are 10(DEC) and 16(HEX).
PrefixDesc.File	Specifies the file prefix for the files that contain prefix descriptions.
Reload	Reloads the data. See "Reloading data into a Pipeline Manager module" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry

```
ifw.DataPool.PrefixDescDataModule.Module.Reload {}
```

For more information, see "Semaphore file syntax" in *BRM System Administrator's Guide*.

Events

Table 37–39 lists the DAT_PrefixDesc events.

Table 37–39 DAT_PrefixDesc Events

Event name	Trigger	Sender	Parameter
EVT_RELOAD_FAILED	Update semaphore	DAT_PrefixDesc	None
EVT_RELOAD_SUCCESSFUL	Update semaphore	DAT_PrefixDesc	None

Database Tables

The DAT_PrefixDesc module uses the IFW_DESTINDESC database table.

For information about the fields in database tables, see the documentation in *Pipeline_Home*/**database**.

To enter data in this table, use Pricing Center. See "Creating Call Destination Descriptions".

DAT_PriceModel

The DAT_PriceModel module provides price model data for the FCT_MainRating module.

See the following documents:

- About Pipeline Rating
- FCT_MainRating

Dependencies

Requires a connection to the Pipeline Manager database.

This module uses event notification to refresh customized product data. You must configure a connection to "DAT_Listener" if you plan to use this feature.

This module uses the **TailormadeProductsSearch** business parameter to skip lock on a price model. If you do not use tailor-made products, and intend to skip lock on a price model, configure a connection to the DAT_PortalConfig module.

Registry Entries

Table 37–40 lists the DAT_PriceModel registry entries.

Table 37–40 DAT_PriceModel Registry Entries

Entry	Description	Mandatory
DataConnection	Specifies the connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
Listener	Specifies the connection to the DAT_Listener module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide* and "DAT_Listener".	No
LogEvents	Specifies whether notification events received by the module are written to the process log file. Default = **False** See "Troubleshooting event handling" in *BRM System Administrator's Guide*.	No
PortalConfigData Module	Specifies the connection to the DAT_PortalConfig module and enables DAT_PriceModel to retrieve business parameter settings from the DAT_PortalConfig module. See "Using Business Parameter Settings from the BRM Database" in *BRM System Administrator's Guide*.	No

Sample Registry Entry

```
PriceModel
{
  ModuleName = DAT_PriceModel
  Module
  {
    DataConnection = ifw.DataPool.Login
    Listener = ifw.DataPool.Listener
  }
}
```

Semaphore File Entries

Table 37–41 lists the DAT_PriceModel Semaphore file entries.

Table 37–41 DAT_PriceModel Semaphore File Entries

Entry	Description
Reload	Reloads the data from the database. See "Reloading data into a Pipeline Manager module" in *BRM System Administrator's Guide*.
PrintAllPriceModels	Prints all price models in the configuration.
PrintOnePriceModel <PriceModel ID>	Prints the price model ID.
PrintRangeOfPriceModels <PriceModel fromID> <PriceModel toID>	Prints all the price models where ID is in the range.

For more information, see "Semaphore file syntax" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry

```
ifw.DataPool.PriceDataModule.Module.Reload {}
```

Events

Table 37–42 lists the DAT_PriceModel events.

Table 37–42 DAT_PriceModel Events

Event Name	Trigger	Sender	Parameter
EVT_RELOAD_SUCCESSFUL	Reload was successful.	DAT_PriceModel	None
EVT_RELOAD_FAILED	Reload failed.	DAT_PriceModel	None

Database Tables

The DAT_PriceModel module uses the following database tables:

- IFW_PRICEMODEL. This table stores price model data. To enter data in this table, use Pricing Center. See "Creating pipeline rate plans and price models" in *BRM Setting Up Pricing and Rating*.
- IFW_PRICEMDL_STEP. This table stores price model step data. To enter data in this table, use Pricing Center.
- IFW_RESOURCE. This table stores resource configuration data. To enter data in this table, use Pricing Center.

For information about the fields in database tables, see the documentation in *Pipeline_Home*/**database**.

Note: For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

DAT_Rateplan

The DAT_Rateplan module provides rate plan data for the FCT_MainRating module.

See the following documents:

- About Configuring Pipeline Rating
- FCT_MainRating

Dependencies

Requires a connection to the Pipeline Manager database.

This module uses event notification to refresh customized product data. You must configure a connection to "DAT_Listener" if you plan to use this feature.

This module uses the **TailormadeProductsSearch** business parameter to skip lock on rate plans. If you do not use tailor-made products, and intend to skip lock on rate plans, configure a connection to the DAT_PortalConfig module.

Registry Entries

Table 37–43 lists the DAT_Rateplan registry entries.

Table 37–43 DAT_Rateplan Registry Entries

Entry	Description	Mandatory
DataConnection	Specifies the connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
Listener	Specifies the connection to the DAT_Listener module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide* and "DAT_Listener".	No
LogEvents	Specifies whether notification events received by the module are written to the process log file. Default = **False** See "Troubleshooting event handling" in *BRM System Administrator's Guide*.	No
PortalConfigData Module	Specifies the connection to the DAT_PortalConfig module and enables DAT_RatePlan to retrieve business parameter settings from the DAT_PortalConfig module. See "Using Business Parameter Settings from the BRM Database" in *BRM System Administrator's Guide*.	No
RowFetchSize	Specifies the number of rows of data to retrieve from the BRM database. Default = **1000**	**RowFetchSize**

Sample Registry Entry

```
Rateplan
{
  ModuleName = DAT_Rateplan
  Module
  {
    DataConnection = ifw.DataPool.Login
    Listener = ifw.DataPool.Listener
  }
}
```

Semaphore File Entries

Table 37–44 lists the DAT_Rateplan Semaphore file entries.

Table 37–44 DAT_Rateplan Semaphore File Entries

Entry	Description
Reload	Reloads the rating configuration data. See "Reloading data into a Pipeline Manager module" in *BRM System Administrator's Guide*.
PrintAllRateplans	Prints all rate plans in the configuration.
PrintOneRateplan <RatePlan ID>	Prints the rate plan ID.
PrintRangeOfRateplans <RatePlan fromID> <RatePlan toID>	Prints all the rate plans where ID is in the range.

For more information, see "Semaphore file syntax" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry

```
ifw.DataPool.RateplanDataModule.Module.Reload {}
```

Events

Table 37–45 lists the DAT_Rateplan events.

Table 37–45 DAT_Rateplan Events

Event Name	Trigger	Sender	Parameter
EVT_RELOAD_SUCCESSFUL	Reload was successful.	DAT_Rateplan	None
EVT_RELOAD_FAILED	Reload failed.	DAT_Rateplan	None

Database Tables

The DAT_Rateplan module uses rate plan data from the following database tables:

- IFW_RATEPLAN
- IFW_RATEPLAN_VER
- IFW_RATEPLAN_CNF

To enter data in these tables, use Pricing Center. See "About pipeline rate plans" in *BRM Setting Up Pricing and Rating*.

The DAT_Rateplan module uses RUM data from the following database tables:

- IFW_RUM
- IFW_RUMGROUP
- IFW_RUMGROUP_LNK

To enter data in these tables, use Pricing Center. See "About Defining Ratable Usage Metrics (RUMs)" in *BRM Setting Up Pricing and Rating*.

For information about the fields in database tables, see the documentation in *Pipeline_Home*/**database**.

> **Note:** For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

DAT_Recycle

The DAT_Recycle module is used by standard recycling and Suspense Manager EDR to recycle EDRs. It connects to the DAT_Listener module and waits for business events that call for EDRs to be recycled.

This module creates a parameter file that allows the "EXT_InEasyDB" module to read suspended usage records associated with a recycle job. It also provides an interface for the "INP_Recycle" module to provide status updates about the EDR stream.

Dependencies

Requires a connection to the "DAT_Listener" module.

Registry Entries

Table 37–46 lists the DAT_Recycle registry entries.

Table 37–46 DAT_Recycle Registry Entries

Entry	Description	Mandatory
ControlPath	Specifies the path for SQL, parameter, job and restart files. **./database/Oracle/Scripts/Suspense**	Yes
Listener	Specifies the connection to the DAT_Listener module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide* and "DAT_Listener".	Yes
LogEvents	Specifies whether notification events received by the module are written to the process log file. Default = **False** See "Troubleshooting Event Handling" in *BRM System Administrator's Guide*.	Yes
ParameterFile	Specifies the name of the parameter file which contains optional key/value entries.	Yes
ProcessCount	Specifies the threshold job count in the **QueueFileName** file. When the threshold is reached, the DAT_Recycle cleans up the queue. If not specified, the default value is **50**.	No
QueueFileName	Specifies the name of the file that stores recycle job IDs to be processed.	Yes
QueueFilePath	The path to the queue file specified for **QueueFileName**.	Yes

Sample Registry Entry

```
#------------------------------------------------------------
# Recycling Data
#------------------------------------------------------------
RecyclingData
{
  ModuleName               = DAT_Recycle
  Module
  {
    Listener = ifw.DataPool.Listener
    LogEvents = True
    ControlPath = ./database/Oracle/Scripts/Suspense
    ParameterFile = parameter.isc
    QueueFileName = RecycleJobIds_wireless.dat
    QueueFilePath = ./data
    ProcessCount = 50
  }
}
```

Semaphore File Entries

DAT_Recycle does not support semaphore updates.

DAT_ResubmitBatch

The DAT_ResubmitBatch module supports batch suspension and resubmission.

DAT_ResubmitBatch subscribes to "DAT_Listener" for ResubmitBatchRequest event. Upon receiving this, it gets information for all the batches corresponding to this

ResubmitBatchRequest from the BRM database. It then moves all these batches to their respective pipeline input directories.

A ResubmitBatchRequest is propagated to the ifw through ifw_sync when a user resubmits a suspended batch with the SMC. During resubmission, a notification event (**/event/notification/suspense/batch_resubmit**) is generated with the admin action job id. This notification event is propagated to the ifw through ifw_sync in form of ResubmitBatchRequest.

Dependencies

This module requires connections to the following:

- BRM database.
- Pipeline Manager database.
- DAT_Listener module. See "DAT_Listener".

Registry Entries

Table 37–47 lists the DAT_ResubmitBatch registry entries.

Table 37–47 DAT_ResubmitBatch Registry Entries

Entry	Description	Mandatory
DataConnection	Specifies the connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
Listener	Specifies the connection to the DAT_Listener module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide* and "DAT_Listener".	Yes
LogEvents	Logs each request received from the listener when true. Default = **True** See "Troubleshooting Event Handling" in *BRM System Administrator's Guide*.	No
QueueFileName	Name of file for file based queue. For example: **QueueFileName** = **ResubmitJobIds.dat**	No
QueueFilePath	Path of file for file based queue	No
QueueFileCleanupThreshold	For already processed events cleanup.	No
PipelineCategory	The Pipeline Category for the records. The module should only process records of its own pipeline category For example: **PipelineCategory= CDRPipeline**	Yes
TempDirectoryPath	Temporary directory path, used as a staging directory for resubmitted batches. It should not be used for any other purpose. For example: TempDirectoryPath = **./data/tmp**	No

Sample Registry Entry

```
ResubmitBatch
  {
    ModuleName = DAT_AccountBatch
    Module
    {
      DataConnection       = ifw.DataPool.LoginInfranet
      Listener             = ifw.DataPool.Listener
      LogEvents            = True
      QueueFileName        = ResubmitJobIds.dat
      QueueFilePath        = ./dataQueue
      FileCleanupThreshold = 50
      PipelineCategory     = CDRPipeline
      TmpDirectoryPath     = ./data/tmp
    }
  }
```

Semaphore File Entries

DAT_ResubmitBatch does not support semaphore updates.

DAT_ScenarioReader

The DAT_ScenarioReader module provides aggregation scenario data for the FCT_AggreGate module.

See the following documents:

- Setting Up Pipeline Aggregation
- FCT_AggreGate

Dependencies

Requires a connection to the Pipeline Manager database.

Registry Entries

Table 37–48 lists the DAT_ScenarioReader registry entries.

Table 37–48 DAT_ScenarioReader Registry Entries

Entry	Description	Mandatory
Calendar	Specifies the calendar that is used for holiday evaluation. Default = No calendar	No
DataCollection	Specifies a connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes

Sample Registry Entry

```
ScenarioReader
{
  ModuleName = DAT_ScenarioReader
  Module
  {
    DataConnection = ifw.DataPool.Database
```

```
    Calendar = 2
  }
}
```

Semaphore File Entries

Table 37–49 lists the DAT_ScenarioReader Semaphore file entries.

Table 37–49 DAT_ScenarioReader Semaphore File Entries

Entry	Description
Reload	Reloads aggregation scenarios. See "Reloading Data into a Pipeline Manager Module" in *BRM System Administrator's Guide*.

For more information, see "Semaphore File Syntax" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry

```
ifw.DataPool.ScenarioReader.Module.Reload {}
```

Messages and Requests

Table 37–50 lists the DAT_ScenarioReader messages and requests.

Table 37–50 DAT_ScenarioReader Messages and Requests

Message/Request	Description	Sending/Receiving
REQ_EVENTHANDLER_NAME	Get the Event Handler.	Send to controller.

Events

Table 37–51 lists the DAT_ScenarioReader events.

Table 37–51 DAT_ScenarioReader Events

Event Name	Trigger	Sender	Parameter
EVT_RELOAD_SUCCESSFUL	Data reload was successful.	DAT_ScenarioReader	None
EVT_RELOAD_FAILED	Data reload failed.	DAT_ScenarioReader	None

Database Tables

The DAT_ScenarioReader module uses the following database tables:

- IFW_SCENARIO. This table stores the aggregation scenario parameters. Some values can be overwritten by using the FCT_AggreGate registry.
- IFW_EDRC_FIELD. This table defines the EDR container fields for the aggregation scenarios. Each scenario uses exactly one EDR container description.
- IFW_CONDITION. This table stores the conditions that exclude an EDR or parts of an EDR from the aggregation process.
- IFW_GROUPING. This tables stores the scenario groupings that group aggregated results into subgroups. You can summarize the values within a grouping into subclasses.

- IFW_AGGREGATION. This tables stores aggregation functions and specifies how to handle the results.
- IFW_GROUPING_CNF. This table links data classes to a grouping.
- IFW_CLASS. This table defines the grouping classes. Each class consists of several class items.
- IFW_CLASSITEM. This table defines class items. All grouping values matching a class item are summarized and the class item code is added to the result.
- IFW_CLASS_LNK. This table links items and classes.
- IFW_CLASSCON. This table defines the class conditions. They determine which class to use when more than one class is associated to a grouping. A class condition specifies the dependency between a class from one grouping and a class item from another grouping.
- IFW_CLASSCON_LNK. This table links class conditions to one or more class items.

To enter data in these tables, use Pricing Center.

For information about the fields in database tables, see the documentation in *Pipeline_Home*/**database**.

Note: For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

DAT_TimeModel

The DAT_TimeModel module provides time model, time zone, and day code data for the FCT_Mainrating module.

See the following documents:

- "Rating By Date and Time with Pipeline Manager" in *BRM Setting Up Pricing and Rating*
- FCT_MainRating

Dependencies

Requires a connection to the Pipeline Manager database.

Registry Entries

Table 37–52 lists the DAT_TimeModel registry entries.

Table 37–52 DAT_TimeModel Registry Entries

Entry	Description	Mandatory
DataConnection	Specifies a connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes

Sample Registry Entry

```
TimeModel
```

```
{
  ModuleName = DAT_TimeModel
  Module
  {
    DataConnection = ifw.DataPool.Login
  }
}
```

Semaphore File Entries

Table 37–53 lists the DAT_TimeModel Semaphore file entries.

Table 37–53 DAT_TimeModel Semaphore File Entries

Entry	Description
Reload	Reloads data from the Pipeline Manager database. See "Reloading Data into a Pipeline Manager Module" in *BRM System Administrator's Guide*.

For more information, see "Semaphore File Syntax" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry

```
ifw.DataPool.TimeDataModule.Module.Reload {}
```

Events

Table 37–54 lists the DAT_TimeModel events.

Table 37–54 DAT_TimeModel Events

Event Name	Trigger	Sender	Parameter
EVT_RELOAD_SUCCESSFUL	Data reload was successful.	DAT_TimeModel	None
EVT_RELOAD_FAILED	Data reload failed.	DAT_TimeModel	None

Database Tables

The DAT_TimeModel module uses the following database tables:

- IFW_TIMEMODEL
- IFW_TIMEMODEL_LNK
- IFW_DAYCODE
- IFW_TIMEINTERVAL
- IFW_TIMEZONE

For information about the fields in database tables, see the documentation in *Pipeline_Home*/**database**.

To enter data in these tables, use Pricing Center.

Note: For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

DAT_USC_Map

The DAT_USC_Map module provides usage scenario mapping data. See "Setting Up Usage Scenario Mapping" in *BRM Setting Up Pricing and Rating*.

The DAT_USC_Map module retrieves mapping data from an ASCII file or from the Pipeline Manager database. This data is used by the FCT_USC_Map module to perform usage scenario mapping. See "FCT_USC_Map".

You define usage scenario mapping rules in Pricing Center.

Dependencies

If the usage scenario mapping is stored in the database, this module requires a connection to the Pipeline Manager database.

Registry Entries

Table 37–55 lists the DAT_USC_Map registry entries.

Table 37–55 DAT_USC_Map Registry Entries

Entry	Description	Mandatory
DataConnection	Specifies a connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes, if the data is stored in the database. Otherwise not used.
LoadZoneDescription	Specifies whether to load the zone description into memory. Default = **False**	No
LoadZoneEntryName	Specifies whether to load the zone name into memory. Default = **False**	Yes
OptimizeFor	Specifies whether mapping should be optimized for **Speed** (the default) or **Memory**.	No
Source	Specifies where the USC mapping data is stored. The possible values are a **File** or **Database**.	Yes
USCMapFile	If **Source = File**, specifies file name and path that contains the USC mapping data. See "Creating a Usage Scenario Map File" in *BRM Setting Up Pricing and Rating*. You can use this entry in a semaphore file.	Yes, if the data is stored in a file. Otherwise not used.

Table 37–55 (Cont.) DAT_USC_Map Registry Entries

Entry	Description	Mandatory
PreCompiledDataDir	Compiles USC mapping data and saves the data to the specified directory. Default = **./compiled_usc_data** Files are stored in the format *USCzoneModelName*.**pc**. Make sure that the directory exists under the specified path. The compiled files are created in the first run of the pipeline. Before each subsequent run, they are validated and recompiled if necessary. See "Precompiling Usage Scenario Mapping Data" in *BRM System Administrator's Guide*.	No
NumberOfThreads	Specifies the number of threads to use when loading and saving the precompiled mapping data. Default = **1** See "Increasing the Number of Threads Used to Load Mapping Data" in *BRM System Administrator's Guide*.	No
UscGroups	Specifies the USC groups for which to load rules. Enclose the values in curly braces. For example: `UscGroups {TEL TEL_ROAMING TEL_INTL}` The default is to load all USC groups in the system. Use the semaphore when mapping rules are stored in the database (Source = **Database**). See "Filtering the Mapping Data to Compile and Load" in *BRM System Administrator's Guide*.	No

Sample Registry Entry

```
USCDataModule
{
  ModuleName = DAT_USC_Map
  Module
  {
     Source               = DataBase
     DataConnection       = ifw.DataPool.Login
     LoadZoneDescription  = False
     LoadZoneEntryName    = False
     OptimizeFor          = MEMORY
     PreCompiledDataDir   = ./compiled_usc_data
     NumberOfThreads      = 5
     UscGroups            = {TEL TEL_ROAMING}
   }
}
```

Semaphore File Entries

Table 37–56 lists the DAT_USC_Map Semaphore file entries.

Table 37–56 DAT_USC_Map Semaphore File Entries

Entry	Description
LoadZoneDescription	Specifies whether to load the zone description into memory. Default = **False**
LoadZoneEntryName	Specifies whether to load the zone name into memory. Valid values are **True** and **False**.
PrintAllUscMapData	Prints all the USC map data.
PrintUscMapDataForZoneModel	Prints the data for a given zone model ID.
PreCompiledDataDir	Compiles USC mapping data and saves the data to the specified directory. Default = **./compiled_usc_data** Files are stored in the format *USCzoneModelName*.**pc**. Make sure that the directory exists under the specified path. The compiled files are created in the first run of the pipeline. Before each subsequent run, they are validated and recompiled if necessary.
NumberOfThreads	Specifies the number of threads to use when loading and saving the precompiled mapping data. Default = **1**
UscGroups	Specifies the USC groups for which to load rules. If not set, all USC groups are loaded.
Reload	Command used to reload data into memory from the database.
USCMapFile	If **Source = File**, specifies file name and path that contains the USC mapping data.

For more information, see "Semaphore File Syntax" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry

```
ifw.DataPool.USCDataModule.Module.UscGroups {TEL TEL_ROAMING}
ifw.DataPool.USCDataModule.Module.Reload {}
```

Database Tables

If the mapping data is stored in the Pipeline Manager database, The DAT_USC_Map module uses the IFW_USC_MAP database table. This table stores mapping rules for usage scenario maps.

For information about the fields in database tables, see the documentation in *Pipeline_Home*/**database**.

For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

DAT_Zone

The DAT_Zone module provides zone data for the FCT_MainRating and the FCT_Zone modules. This module stores the real-time service class to Pipeline Manager service code mapping information in memory. When it processes realtime data, it returns the service code for a given service class.

See the following documents:

- "Setting up Zones for Batch Pipeline Rating" in *BRM Setting Up Pricing and Rating*
- "Setting up Zones by using Pricing Center" in *BRM Setting Up Pricing and Rating*.

Dependencies

Requires a connection to the Pipeline Manager database.

Registry Entries

Table 37–57 lists the DAT_Zone registry entries.

Table 37–57 DAT_Zone Registry Entries

Entry	Description	Mandatory
CliMode	Specifies if the zoning tree should be created in decimal (DEC) or hexadecimal (HEX) mode. Default = **DEC**	No
DataConnection	Specifies a connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes, if the module gets data from the database.
DistCalcMode	Specifies the mode for calculating the distance between two area codes: ■ **DISC**. The configured coordinates are Cartesian coordinates. The distance is calculated using the Pythagorean theorem. ■ **GLOBE**. The coordinates are global. The distance is calculated using slower goniometric functions. You can use this entry in a semaphore file. Default = **DISC**	No
FileName	Specifies the path and file name for the zone model master file, if the module gets data from a file. You can use this entry in a semaphore file. See "Creating Zone Data Files" in *BRM Setting Up Pricing and Rating*.	Yes, if the module gets data from a file
GeoFileName	Specifies the path and file name for the area code coordinate link file, if the module gets data from a file. You can use this entry in a semaphore file. See "Creating Zone Data Files" in *BRM Setting Up Pricing and Rating*.	Yes, if the module gets data from a file
LoadZoneDescription	Specifies whether to load the zone descriptions into memory. Default = **False** You can use this entry in a semaphore file.	No
LoadZoneEntryName	Specifies whether to load the zone names into memory. Default = **False** You can use this entry in a semaphore file.	No

Table 37–57 (Cont.) DAT_Zone Registry Entries

Entry	Description	Mandatory
MaxAge	Specifies the maximum age of zone entries. If the value is **0** or null, all zone entries are loaded. You can use this entry in a semaphore file. Default = **0**	No
RealTime	Specifies whether the module should process real-time events. If **True**, the module processes real-time events, if **False**, the module processes batch events.	Yes
ReuseOnFailure	Specifies if the module should continue to use the old data if the **Reload** command fails: **True** = use the old data. **False** = do not use the old data. Default = **False**	Yes
Source	Specifies whether the module gets data from a file or the database.	Yes
ZoneModels	Specifies the source of the zone model codes: ■ If the source is a file, contains a list of zone model codes with the corresponding path and file name for the configuration file. ■ If the source is the database, contains a list of zone model codes that shall be used. You can use this entry in a semaphore file. See "Creating Zone Data Files" in *BRM Setting Up Pricing and Rating*.	No, if the module gets data from the database.

Sample Registry for the Database Interface

```
Module
{
  ReuseOnFailure = FALSE
  MaxAge = 0
  Source = Database
  DataConnection = ifw.DataPool.Login
  LoadZoneDescription = False
  LoadZoneDescription = False
  ZoneModels
  {
    BASIC
    PROFI
    SPECIAL
  }
}
```

Sample Registry for the File Interface

Standard zone:

```
Module
{
   ReuseOnFailure = FALSE
   MaxAge = 90
```

```
   Source = File
   FileName = ./cfg/ZoneModelConfig.dat
   ZoneModels
   {
      ZM_ADD = /data9/INTEGRATE/TEST/config/ZM_ADD.dat
   }
}
```

Geographical zone:

```
Module
{
   ReuseOnFailure = FALSE
   MaxAge = 90
   Source = File
   FileName = ./cfg/ZoneModelConfig.dat
   GeoFileName = ./cfg/GeoAreaLink.dat
   ZoneModels
   {
      ZM_GEO = /data9/INTEGRATE/TEST/config/ZM_GEO.dat
   }
}
```

Sample Registry for Real-Time Zoning

```
Module
{
  ReuseOnFailure = FALSE
  Source = DataBase
  MaxAge = 0
  DistCalcMode = DISC
  DataConnection = ifw.DataPool.Login
  LoadZoneDescription = False
  LoadZoneEntryName = False
  RealTime = True
  ZoneModels
  {

  }
}
```

Semaphore File Entries

Table 37–58 lists the DAT_Zone Semaphore file entries.

Table 37–58 DAT_Zone Semaphore File Entries

Entry	Description
DistCalcMode	Specifies the mode for calculating the distance between two area codes: ■ **DISC**. The configured coordinates are cartesian coordinates. The distance is calculated using the Pythagorean theorem. ■ **GLOBE**. The coordinates are global. The distance is calculated using slower goniometric functions.
FileName	Specifies the path and file name for the zone model master file, if the module gets data from a file. See "Creating Zone Data Files" in *BRM Setting Up Pricing and Rating*.
GeoFileName	Specifies the path and file name for the area code coordinate link file, if the module gets data from a file. See "Creating Zone Data Files" in *BRM Setting Up Pricing and Rating*.
LoadZoneDescription	Specifies whether to load the zone descriptions into memory. **Note:** When this entry is updated through a semaphore, the reload semaphore must also be passed to reload the zone descriptions.
LoadZoneEntryName	Specifies whether to load the zone names into memory. **Note:** When this entry is updated through a semaphore, the reload semaphore must also be passed to reload the zone names.
MaxAge	Specifies the maximum age of zone entries. If the value is 0 or null, all zone entries are loaded.
Reload	Reloads the zoning data. See "Reloading Data into a Pipeline Manager Module" in *BRM System Administrator's Guide*.
ZoneModels	Specifies the source of the zone model codes: ■ If the source is a file, contains a list of zone model codes with the corresponding path and file name for the configuration file. ■ If the source is the database, contains a list of zone model codes that shall be used. See "Creating Zone Data Files" in *BRM Setting Up Pricing and Rating*.

For more information, see "Semaphore File Syntax" in *BRM System Administrator's Guide*.

Sample Semaphore File Entry for the Database Interface

```
ifw.DataPool.ZoneDataModule.Module.ZoneModels.BASIC.Reload {}
ifw.DataPool.ZoneDataModule.Module.ZoneModels.SPECIAL.Reload {}
```

Sample Semaphore File Entry for the File Interface

```
ifw.DataPool.ZoneDataModule.Module.ZoneModels.
ZM_ADD = /data9/INTEGRATE/test/config/ZM_ADD-new.dat

ifw.DataPool.ZoneDataModule.Module.ZoneModels.
ZM_MOBILE = /data9/INTEGRATE/test/config/ZM_MOBILE-new.dat
```

Events

Table 37–59 lists the DAT_Zone events.

Table 37–59 DAT_Zone Events

Event Name	Trigger	Sender	Parameter
EVT_RELOAD_SUCCESSFUL	Data reload was successful.	DAT_Zone	None
EVT_RELOAD_FAILED	Data reload failed.	DAT_Zone	None

Database Tables

The DAT_Zone module uses data from the following database tables:

- IFW_ZONEMODEL
- IFW_IMPACT_CAT
- IFE_STANDARD_ZONE
- IFW_GEO_MODEL
- IFW_GEO_ZONE
- IFW_GEOAREA_LNK

For information about the fields in database tables, see the documentation in *Pipeline_Home*/**database**.

Note: For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

To enter data in these tables, use Pricing Center. See "Setting up Zone Models" in *BRM Setting Up Pricing and Rating*.

38

Pipeline Manager iRules

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager iRules.

IRL_EventTypeSplitting

The IRL_EventTypeSplitting iRule sends EDRs to separate output streams based on service codes.

See "Sending EDRs to Pipeline Output Streams".

To run the iRule, configure the FCT_IRules module. See:

- FCT_IRules
- "About configuring iRules" in *BRM System Administrator's Guide*

Dependencies

This module must run after the FCT_ServiceCodeMap module and before the FCT_Reject module.

This is typically the last module before the FCT_Reject module.

For more information, see "Function Module Dependencies".

Sample Registry

```
EventSplitting
{
 ModuleName = FCT_IRules
 Module
 {
    Active = True
    Source = Database
    DataConnection = ifw.DataPool.DataConnection
    Rules {}
 }
}
```

EDR Container Fields

Table 38–1 lists the IRL_EventTypeSplitting EDR Container fields.

Table 38–1 IRL_EventTypeSplitting EDR Container Fields

Alias field name Default field name	Type	Access	Description
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Internal service code.

IRL_EventTypeSplitting_tt

The IRL_EventTypeSplitting_tt iRule sends EDRs to separate output streams based on the In-Memory Database (IMDB) Cache logical partition and service code.

See "Sending EDRs to Pipeline Output Streams".

To run the iRule, configure the FCT_IRules module. See:

- FCT_IRules
- "About configuring iRules" in *BRM System Administrator's Guide*

Dependencies

This module must run after the FCT_ServiceCodeMap module and before the FCT_Reject module.

This is typically the last module before the FCT_Reject module.

For more information, see "Function Module Dependencies".

Sample Registry

```
ServiceOutputSplit
{
 ModuleName = FCT_IRules
 Module
 {
    Active = True
    Source = Files
    Rules
    {
    }
    Descriptions
    {
        ServiceCodeSplit = ./iScriptLib/iScriptLib_Standard/IRL_
EventTypeSplitting_tt.irl
    }
 }
}
```

EDR Container Fields

Table 38–2 lists the IRL_EventTypeSplitting_tt EDR Container fields.

Table 38–2 IRL_EventTypeSplitting_tt EDR Container Fields

Alias field name Default field name	Type	Access	Description
LOGICAL_PARTITION_ID DETAIL.LOGICAL_PARTITION_ID	String	Read	Logical Partition ID.
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Internal service code.

IRL_LeastCostPerEDR

The IRL_LeastCostPerEDR iRule flags all EDRs that satisfy the criteria for BRM least=1 cost rating. For more information, see "About Least Cost Rating".

You set up the criteria that an EDR must meet to qualify for least=1 cost rating in the **IRL_LeastCostPerEDR.irl** and **IRL_LeastCostPerEDR.data** files. See "Specifying the Rules to Qualify for Least Cost Rating".

To run the iRule, configure the FCT_IRules module. See:

- FCT_IRules
- "About Configuring iRules" in *BRM System Administrator's Guide*

Dependencies

This module must run:

- Before the "FCT_CustomerRating" module, because FCT_CustomerRating uses the flag set by this module to decide whether to create charge packets for all products.
- After the "FCT_Filter_Set" module, because the rules you set up in **IRL_LeastCostRating.data** frequently use filter sets as one criteria for least=1 cost rating.

For more information, see "Function Module Dependencies".

Registry Entries

Table 38–3 lists the IRL_LeastCostEDR registry entries.

Table 38–3 IRL_LeastCostEDR Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. ■ **True** = Active ■ **False** = Inactive	Yes
LeastCostCheck	Specifies the path to the **IRL_LeastCostPerEDR.irl** file. See "Specifying the Rules to Qualify for Least Cost Rating".	Yes
Source	Specifies whether the least= cost rating data is stored in a file or in a database table. ■ **File** = The iRules data is stored in a file. ■ **Database** = The iRules data is stored in a database table.	Yes

Sample Registry

```
LeastCostPerEDR
{
   ModuleName = FCT_IRules
   Module
   {
      Active = False
      Source = Database
      DataConnection = ifw.DataPool.DataConnection
              Rules {}
   }
   }
```

EDR Container Fields

Table 38–4 lists the IRL_LeastCostEDR EDR Container fields.

Table 38–4 IRL_LeastCostEDR EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.CUST_A.LEAST_COST	Integer	Write	Toggles least= cost rating on and off. A value of **1** means do not apply least cost rating; a value of **2** specifies to apply least= cost rating.
DETAIL.CUST_A.MARKET_SEGMENT	String	Read	Specifies the filter set associated with the EDR.
DETAIL.CUST_A.PRODUCT_PRIORITY	String	Read	Contains a list of the priorities for all products that are associated with the same service and event.
DETAIL.CUST_A.INTERN_FOUND_PP_INDEX	String	Write	Contains the index of the highest priority rating product. This is the product with the highest rate priority for an event. In the case of two products with matching priorities, the product with the first start time is selected.

IRL_PipelineSplitting

The IRL_PipelineSplitting iRule is used in the pre-recycling pipeline to send EDRs to different output streams depending on their original pipeline names. The EDRs are then routed to their original pipelines for recycling.

The **PipelineSplitting.irl** file specified in the registry references a data file called **PipelineSplitting.data**, which you must modify based on your pipeline names. The default contents of the file are:

```
ALL_RATE;PreRecycleOutput
ALL_RATE_2;PreRecycleOutput_2
.*;PreRecylceOutput
```

See "Configuring a Pre-Recycling Pipeline".

For more information about iRules, see "Creating iScripts and iRules" in *BRM Developer's Guide*.

To run the iRule, configure the FCT_IRules module. See:

- FCT_IRules
- "About configuring iRules" in *BRM System Administrator's Guide*

Sample Registry

```
PipelineSplit
{
  ModuleName = FCT_IRules
  Module
  {
    Active = True
      Source = Database
      DataConnection = ifw.DataPool.DataConnection
      Rules
      {
      }
  }
}
```

IRL_PromotionalSavingPerEDR

The IRL_PromotionalSavingPerEDR iRule flags all EDRs that satisfy the criteria for a promotional savings calculation.

You set up the rules to qualify for the promotional savings calculation in the promotional savings iRules files (**IRL_PormotionalSavingPerEDR.irl** and **IRL_PromotionalSavingPerEDR.data**).

For more information, see "About Calculating the Promotional Savings".

To run the iRule, configure the FCT_IRules module. See:

- FCT_Zone
- "About configuring iRules" in *BRM System Administrator's Guide*

Dependencies

This module must run before the IRL_LeastCost module and the FCT_CustomerRating module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 38–5 lists the IRL_PromotionalSavingPerEDR registry entries.

Table 38–5 IRL_PromotionalSavingPerEDR Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. ■ **True** = Active ■ **False** = Inactive	Yes
PromotionalSaving	Specifies the path to the **IRL_PromotionalSavingPerEDR.irl** file. See "Specifying the Rules to Qualify for Promotional Savings".	No
Source	Specifies whether the iRules data is stored in a file or database table. ■ **File** = The iRules data is stored in a file. ■ **Database** = The iRules data is stored in a database table.	Yes

Sample Registry

```
PromotionalSavingPerEDR
{
   ModuleName = FCT_IRules
   Module
   {
      Active = True
      Source = Database
      DataConnection = ifw.DataPool.DataConnection
          Rules {}
   }
}
```

EDR Container Fields

Table 38–6 lists the IRL_PromotionalSavingPerEDR EDR Container fields.

Table 38–6 IRL_PromotionalSavingPerEDR EDR Container Fields.

Alias field name Default field name	Type	Access	Description
DETAIL.CUST_A.PROMOTIONAL_SAVING	Integer	Write	Toggles promotional savings on and off. A value of **1** means do not apply promotional savings. A value of **2** applies promotional savings.
DETAIL.CUST_A.MARKET_SEGMENT	String	Read	Specifies the filter set associated with an EDR.
DETAIL.CUST_A.PRODUCT_PRIORITY	String	Read	Contains a list of the priorities for all products that are associated with the same service and event.
DETAIL.CUST_A.INTER_RATING_PRODUCTS	String	Write	Contains the product rating indexes. This is a comma-separated list of all rating products' indexes associated with the same service and event, and their priorities.

IRL_UsageType

The IRL_UsageType iRule assigns usage types to EDRs.

See "Mapping usage types" in *BRM Setting Up Pricing and Rating*.

To run the iRule, configure the FCT_IRules module. See:

- FCT_IRules
- "About configuring iRules" in *BRM System Administrator's Guide*

Dependencies

This module must be run after the FCT_Account module and before the FCT_USC_Map module.

For more information, see "Function Module Dependencies".

Sample Registry

```
IRules
{
  ModuleName = FCT_IRules
  Module
  {
     Active = True
     Source = Database
     DataConnection = integrate.DataPool.DataConnection
     Rules {}
  }
}
```

EDR Container Fields

Table 38–7 lists the IRL_UsageType EDR Container fields.

Table 38–7 IRL_UsageType EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.CUST_A.ERA.PROFILE	String	Read	Contains the profile for customer A
DETAIL.CUST_B.ERA.PROFILE	String	Read	Contains the profile for customer B
DETAIL.CUST_A.PRODUCT.ERA.PROFILE	String	Read	Contains the product profile for customer A
DETAIL.CUST_B.PRODUCT.ERA.PROFILE	String	Read	Contains the product profile for customer B
DETAIL.CUST_A.ERA.PA.KEY	String	Read	Contains the customer A profile attribute key
DETAIL.CUST_A.ERA.PA.VALUE	String	Read	Contains the customer A profile attribute value
DETAIL.CUST_B.ERA.PA.KEY	String	Read	Contains the customer B profile attribute key
DETAIL.CUST_B.ERA.PA.VALUE	String	Read	Contains the customer B profile attribute value

Table 38–7 (Cont.) IRL_UsageType EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.CUST_A.PRODUCT.ERA.PA.KEY	String	Read	Contains the customer A product profile attribute key
DETAIL.CUST_A.PRODUCT.ERA.PA.VALUE	String	Read	Contains the customer A product profile attribute key
DETAIL.CUST_B.PRODUCT.ERA.PA.KEY	String	Read	Contains the customer B product profile attribute key
DETAIL.CUST_B.PRODUCT.ERA.PA.VALUE	String	Read	Contains the customer B product profile attribute key
DETAIL.CUST_A.INTERN_FOUND_PP_INDEX	String	Read	Contains the internal found purchases product index.
DETAIL.USAGE_DIRECTION	String	Read	Contains the usage direction.
DETAIL.CONNECT_SUB_TYPE	String	Read	Contains the connection subtype
DETAIL.CUST_A.ACCOUNT_NO	String	Read	Contains the customer A account number.
DETAIL.CUST_B.ACCOUNT_NO	String	Read	Contains the customer B account number
DETAIL.CUST_A.SYSTEM_BRAND	String	Read	Contains the system brand for customer A
DETAIL.CUST_B.SYSTEM_BRAND	String	Read	Contains the system brand for customer B
DETAIL.CUST_A.BILL_CYCLE	String	Read	Contains the customer A bill cycle
DETAIL.B_NUMBER	String	Read	Contains the B number for call
DETAIL.ASS_GSMW_EXT.CELL_ID	String	Read	Contains the cell ID for GSM call
DETAIL.CHARGING_START_TIMESTAMP	String	Read	Contains the start time for call
DETAIL.CUST_A.PRODUCT.RATEPLAN_NAME	String	Read	Contains the rate plan for customer A product
DETAIL.CUST_B.PRODUCT.RATEPLAN_NAME	String	Read	Contains the rate plan for customer B product

iRuleValidation

iRuleValidation is an instance of the FCT_IRules module used to validate the data in individual CIBER fields in the EDR container. iRuleValidation uses the **CIBER_VAL.xml** file that specifies the rules and rule items for validating CIBER fields.

You must load the rules in the **CIBER_VAL.xml** file into the Pipeline Manager database before starting Pipeline Manager. See "Importing and Exporting Validation Rules" in *BRM Developer's Guide*.

For information about validating CIBER records for incollect processing, see "About Validating Roaming Usage Data" in *BRM Configuring Roaming in Pipeline Manager*.

To run the iRule, configure the FCT_IRules module. See:

- FCT_IRules

- "About configuring iRules" in *BRM System Administrator's Guide*

Dependencies

Run this iRule before the ISC_TapSplitting module.

For more information , see "Function Module Dependencies".

Sample Registry

```
iRuleValidation
{
  ModuleName = FCT_IRules
  Module
  {
     Active = True
     Source = Database
     DataConnection = ifw.DataPool.DataConnection
     Rules
     {
        CIBER_VAL
     }
  }
}
```

39

Pipeline Manager iScripts

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager iScripts.

ISC_AddCBD

The ISC_AddCBD iScript prepares EDRs for rerating in the backout pipeline.

> **Note:** The ISC_AddCBD iScript is a deprecated module but remains in BRM for backward compatibility.

For information, see "About Rerating Pipeline-Rated Events" in *BRM Setting Up Pricing and Rating*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

This module runs in its own backout pipeline for rerating.

For more information, see "Function Module Dependencies".

Sample Registry

```
AddCBD
{
  ModuleName = FCT_IScript
  Module
  {
    Active = TRUE
    Source = FILE
    Scripts
    {
      AddCBD
      {
        FileName = ./iScriptLib/iScriptLib_Standard/ISC_AddCBD.isc
      }
    }

}
```

Modified Output Container Fields

The ISC_AddCBD iScript creates one associated charge breakdown record of type 981 with charge packets of type 680.

EDR Container Fields

Table 39–1 lists the ISC_AddCBD EDR container fields.

Table 39–1 ISC_AddCBD EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_PIN.ACCOUNT_POID	String	Read	Contains the BRM account POID.
DETAIL.ASS_PIN.BP.PIN_INFO_STRING	String	Read	Contains the string that stores aggregated balance packet information.
DETAIL.ASS_PIN.BP.PIN_AMOUNT	Decimal	Read/Write	Contains the monetary balance impact.
DETAIL.ASS_PIN.BP.PIN_DISCOUNT	Decimal	Read/Write	Contains the discount value
DETAIL.ASS_CBD.ACCOUNT_CODE	String	Write	Contains the account code related to charge packet being constructed.
DETAIL.ASS_CBD.RECORD_TYPE	String	Write	Contains the record type of charge packet.
DETAIL.ASS_CBD.CP	Data Block	Write	Charge packet data block.
DETAIL.DISCOUNT_KEY	String	Write	Contains the discount key.
DETAIL.ASS_CBD.CP.DP	Data Block	Write	Charge breakdown discount packet data block.

Required Input EDR Container Fields

The associated BRM billing record type 900 and the balance impacts of type 600 must be present.

ISC_ApplyTax

The ISC_ApplyTax iScript is used in the reprice pipeline during Incollect processing. This iScript retrieves the taxation flag value for the specific network operator from the in-memory cache. If the taxation flag is set to **on**, the tax amount is passed to the subscriber; otherwise, the tax amount is ignored.

For more information, see "Choosing Whether To Apply Taxes For Roaming" in *BRM Configuring Roaming in Pipeline Manager*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

This iScript requires the DAT_InterConnect module and the iScript extension IXT_OpFlag, which provide network operator configuration information that is accessed by ISC_ApplyTax.

For more information, see "Function Module Dependencies".

Sample Registry

```
ApplyTaxIScript
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = File
      Scripts
      {
         ApplyTaxIScript
         {
            FileName = ./iScriptLib/iScriptLib_Roaming/ISC_ApplyTax.isc
         }
      }
   }
}
```

EDR Container Fields

Table 39–2 lists the ISC_ApplyTax EDR container fields.

Table 39–2 ISC_ApplyTax EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE	Decimal	Read/Write	Contains the charge amount value.
DETAIL.ASS_ROAMING_EXT.SENDER	String	Read	Contains the sender PLMN of the Transferred Account Procedure (TAP) file.

ISC_BACKOUTTypeSplitting

The ISC_BACKOUTTypeSplitting iScript is used by the backout pipeline for backout-only rerating. It determines if the EDRs are flagged for backout-only rerating and sends the EDRs to different output streams based on the event types.

See "About Configuring The Backout Pipeline For Back-Out-Only Rerating" in *BRM Setting Up Pricing and Rating*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Sample Registry

```
BackOutputSplit
```

```
{
     ModuleName = FCT_IScript
     Module
     {
          Active = TRUE
          Source = File
          Scripts
          {
               SplitScript
               {
                    FileName = ./iScriptLib/iScriptLib_Standard/ISC_BACKOUTTypeSplitting.isc
               }
          }
     }
}
```

ISC_CiberInputValidation

The ISC_CiberInputValidation iScript performs record-level validations of CIBER records.

For information about validating CIBER records, see "About Validating Roaming Usage Data" in *BRM Configuring Roaming in Pipeline Manager*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

Because erroneous CIBER records can be discarded, this module must run before the FCT_Discard module.

For more information, see "Function Module Dependencies".

Sample Registry

```
ISC_CiberInputValidation
{
  ModuleName = FCT_IScript
  Module
  {
     Active = TRUE
     Source = FILE
     Scripts
     {
       CiberInputValidation
       {
            FileName = ./iScriptLib/iScriptLib_Standard/ISC_CiberInputValidation.isc
       }
    }
  }
}
```

ISC_CiberOutputMapping

The ISC_CiberOutputMapping iScript adds charge data to the ASSOCIATED_CIBER_EXTENSION block of the EDR. If the EDR does not contain an ASSOCIATED_CIBER_EXTENSION block, this iScript adds one.

See "About Settling Roaming Charges" in *BRM Configuring Roaming in Pipeline Manager*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

This module must run after the FCT_MainRating module and the ISC_PostRating iScript.

For more information, see "Function Module Dependencies".

Sample Registry

```
ISC_CiberOutputMapping
{
  ModuleName = FCT_IScript
  Module
  {
    Active = TRUE
    Source = FILE
    Scripts
    {
       CiberOutputMapping
       {
          AirRum = AIR
          TollRum = TOL
          OCCRum = OCC
          FileName = ./iScriptLib/iScriptLib_Standard/ISC_CiberOutputMapping.isc
       }
    }
  }
}
```

EDR Container Fields

Table 39–3 lists the ISC_CiberOutputMapping EDR container fields.

Table 39–3 ISC_CiberOutputMapping EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_CURRENCY	String	Read	Specifies the currency type.
CURRENCY_TYPE	String	Write	Specifies the currency type.
DETAIL.ASS_CBD.CP.RUM	String	Read	Checked to see if the EDR contains a toll charge.

Table 39–3 (Cont.) ISC_CiberOutputMapping EDR Container Fields

Alias field name Default field name	Type	Access	Description
TOLL_NETWORK_CARRIER_ID	String	Write	If the EDR contains a toll charge, this is set to **99001**, otherwise it is set to **00000**.
AIR_CHARGE Set only for CIBER record type 22 (specified in DETAIL.ASS_CIBER_EXT.CIBER_RECORD_TYPE).	String	Write	If the value of AIR_CHARGABLE_TIME in the CIBER extension block is **0**, AIR_CHARGE is also set to **0**. Otherwise, it is set to the sum of the DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE in each charge packet for which all of the following statements are true: ■ RUM equals the value specified in the **AirRum** parameter in the registry ■ CHARGED_CURRENCY_TYPE equals "R" (rated) or "H" (home) ■ PRICEMODEL_TYPE equals "S" or "A" **Note:** If the value of DETAIL.ASS_CIBER_EXT.SPECIAL_FEATURES_USED in the CIBER extension block is F, then AIR_CHARGE is always set to **0.0**. The Special Features Used value is set in the CIBER input grammar.
AIR_RATE_PERIOD Set only for CIBER record type 22 (specified in DETAIL.ASS_CIBER_EXT.CIBER_RECORD_TYPE).	String	Write	If the value of AIR_CHARGE is **0**, this field is set to **00**. Otherwise, this field is set to **01**. **Note:** If the value of DETAIL.ASS_CIBER_EXT.SPECIAL_FEATURES_USED in the CIBER extension block is F, then AIR_CHARGE is always set to **0.0**, and therefore AIR_RATE_PERIOD is set to **00**. The value of AIR_RATE_PERIOD is generally derived from the time interval code in the charge packet where RUM equals the value of the **AirRum** registry parameter. The time interval code is part of the time model that you define. Therefore, the fields used to derive the CIBER extension AIR_RATE_PERIOD field is specific to your customization.

Table 39–3 (Cont.) ISC_CiberOutputMapping EDR Container Fields

Alias field name Default field name	Type	Access	Description
AIR_MULTIRATE_PERIOD Set only for CIBER record type 22 (specified in DETAIL.ASS_CIBER_EXT.CIBER_RECORD_TYPE).	N/A	Write	If the value of AIR_CHARGE is **0**, this field is set to **0**. Otherwise, this field is set to **1**. **Note:** If the value of DETAIL.ASS_CIBER_EXT.SPECIAL_FEATURES_USED in the CIBER extension block is F, then AIR_CHARGE is always set to **0.0**, and therefore AIR_MULTIRATE_PERIOD is set to **0**. The value of AIR_MULTIRATE_PERIOD is generally derived from the time interval code in the charge packet where RUM equals the value of the **AirRum** registry parameter. The time interval code is part of the time model that you define. Therefore, the fields used to derive the CIBER extension AIR_RATE_PERIOD field is specific to your customization.
TOLL_CHARGE Set only for CIBER record type 22 (specified in DETAIL.ASS_CIBER_EXT.CIBER_RECORD_TYPE).	N/A	Write	If the value of AIR_CHARGABLE_TIME in the CIBER extension block is **0**, TOLL_CHARGE is also set to **0**. Otherwise, it is set to the sum of the DETAIL.ASS_CBD.CHARGED_AMOUNT_VALUE in each charge packet for which all of the following statements are true: ■ RUM equals the value specified in the **AirRum** parameter in the registry ■ CHARGED_CURRENCY_TYPE equals "R" (rated) or "H" (home) ■ PRICEMODEL_TYPE equals "S" or "A"

Table 39–3 (Cont.) ISC_CiberOutputMapping EDR Container Fields

Alias field name Default field name	Type	Access	Description
TOLL_RATE_PERIOD (for CIBER record type 22 only)	N/A	Write	If the value of TOLL_CHARGE is **0**, this field is set to **00**. Otherwise, this field is set to **01**. The value of TOLL_RATE_PERIOD is generally derived from the time interval code in the charge packet where RUM equals the value of the **TollRum** registry parameter. The time interval code is part of the time model that you define. Therefore, the fields used to derive the CIBER extension TOLL_RATE_PERIOD field are specific to your customization.
TOLL_MULTIRATE_PERIOD (for CIBER record type 22 only)	N/A	Write	If the value of TOLL_CHARGE is **0**, this field is set to **0**. Otherwise, this field is set to **1**. The value of TOLL_MULTIRATE_PERIOD is generally derived from the time interval code in the charge packet where RUM equals the value of the **TollRum** registry parameter. The time interval code is part of the time model that you define. Therefore, the fields used to derive the CIBER extension TOLL_MULTIRATE_PERIOD field are specific to your customization.
TOLL_RATE_PERIOD and TOLL_MULTIRATE_PERIOD (for CIBER record type 32 and 42 only) Sets these Charge NO 1 related fields: ■ **Charge NO 1** ■ **Charge NO 1 Indicator** ■ **Charge NO 1 Rate Period** ■ **Charge NO 1 Multi-Rate Period**	N/A	Write	**Charge NO 1** is set to the value of DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE. The values for the other Charge NO 1 related fields are based on the CIBER record type: Record type 32: ■ **Indicator** is set to **17** ■ **Rate Period** is set to **01** ■ **Multi-Rate Period** is set to **1** Record type 42: ■ **Indicator** is set to **12** ■ **Rate Period** is set to **04** ■ **Multi-Rate Period** is set to **1**

ISC_CiberRejectReason

The ISC_CiberRejectReason iScript sets a reason code in the CIBER extension block for records that are rejected for one of the following reasons:

- They are duplicates.
- There is no roaming agreement with the network operator.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript

- "About Configuring iScripts" in *BRM System Administrator's Guide*

Sample Registry

```
ISC_CiberRejectReason
{
  ModuleName = FCT_IScript
  Module
  {
     Active = TRUE
     Source = FILE
     Scripts
     {
        CiberRejectReason
        {
           FileName = ./iScriptLib/iScriptLib_CiberValidation/ISC_CiberRejectReason.isc
        }
     }
  }
}
```

ISC_ConsolidatedCP

The ISC_ConsolidatedCP iScript is used in the Incollect settlement pipeline and the Outcollect settlement pipeline. This iScript removes all non '00' impact category charge packets.

The TAP input grammar creates individual charge packets as well as consolidated charge packets. However, the FCT_BillingRecord module considers all charge packets for creating balance packets. For this reason, the individual charge packets (non '00' impact category charge packets) are removed and only consolidated charge packets are considered so that the balance amounts in the balance packets are correct.

This iScript also assigns the G/L ID to each consolidated charge packet based on the **GL_CODE** registry entry.

For more information, see "About Settling Roaming Charges" in *BRM Configuring Roaming in Pipeline Manager* and "About Processing Rejected Outcollect TAP Files And Records" in *BRM Configuring Roaming in Pipeline Manager*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Registry Parameters

Table 39–4 lists the ISC_ConsolidatedCP registry parameter.

Table 39–4 ISC_ConsolidatedCP Registry Parameter

Registry Parameter	Description	Mandatory
GL_CODE	Specifies the G/L ID used for tracking the balance impacts of roaming usage events.	Yes

Sample Registry

```
ConsolidatedCP
{
```

```
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = File
      Scripts
      {
         ConsolidatedCPIScript
         {
            FileName = ./iScriptLib/iScriptLib_Roaming/ISC_ConsolidatedCP.isc
            GL_CODE = 1514
         }
      }
   }
}
```

EDR Container Fields

Table 39–5 lists the ISC_ConsolidatedCP EDR container fields.

Table 39–5 ISC_ConsolidatedCP EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_CBD.CP.IMPACT_CATEGORY	String	Read	Contains the impact category of the charge packet.
DETAIL.ASS_CBD.CP.RUM_ID	Long	Write	Contains the RUM ID.
DETAIL.ASS_CBD.CP.USAGE_GL_ ACCOUNT_CODE	String	Write	Contains the G/L account.
DETAIL.ASS_CBD.TP.RELATED_CHARGE_ INFO_ID	Integer	Write	Contains the corresponding charge packet index.

ISC_DupRAPRecords

The ISC_DupRAPRecords iScript is used in the RAP processing pipeline. It duplicates severe and fatal TAP records so that the records can be routed to multiple output streams.

For more information, see "About Processing Rejected Outcollect TAP Files And Records" in *BRM Configuring Roaming in Pipeline Manager*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Registry Parameters

Table 39–6 lists the ISC_DupRAPRecords registry parameter.

Table 39–6 ISC_DupRAPRecords Registry Parameter

Registry Parameter	Description	Mandatory
NULLStream	Specifies the DevNull output stream. See "OUT_DevNull".	Yes

Sample Registry

```
DupRAPRecords
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = File
      Scripts
      {
         DupRAPRecords
         {
            FileName = ./iScriptLib/iScriptLib_Roaming/ISC_DupRAPRecords.isc
            NULLStream = DevNull
         }
      }
   }
}
```

EDR Container Fields

Table 39–7 lists the ISC_DupRAPRecords EDR container fields.

Table 39–7 ISC_DupRAPRecords EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ERROR_REJECT_TYPE	String	Write	FCT_Reject uses this to reject the detail to a stream other than the standard reject stream.
DETAIL.ASS_ROAMING_EXT.RAP_RECORD_TYPE	String	Read	RAP record type.

ISC_EDRToTAPOUTMap

The ISC_EDRToTAPOUTMap iScript is used to populate standard values to fields in the output TAP file based on its corresponding value in the EDR container. This is because for some fields EDR might have different internal representations and the TAP specification may ask for different representation for the same fields.

This iScript has to be customized by the user for mapping fields and specifying what is the internal EDR representation for the TAP fields specified value given in the standards document.

Note: The ISC_EDRToTAPOUTMap iScript is a deprecated module but remains in BRM for backward compatibility.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Sample Registry

```
ISC_EDRToTAPOUTMap
```

```
{
  ModuleName = FCT_IScript
  Module
  {
     Active = TRUE
     Source = FILE
     Scripts
     {
       EDRToTAPOUTMap
       {
            FileName = ./iScriptLib/iScriptLib_Standard/ISC_EDRToTAPOUTMap.isc
       }
     }
  }
}
```

EDR Container Fields

Table 39–8 lists the ISC_EDRToTAPOUTMap EDR container fields.

Table 39–8 ISC_EDRToTAPOUTMap EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_CBD.CP.DAY_CODE	String	Read / Write	External Day Code as estimated and used by the rating process. Sample mapping given in the script: (Can be customized as per user's requirement) WEEKDAY => N WEEKEND => P ALLDAYS => I
DETAIL.ASS_CBD.CP.TIME_INTERVAL_CODE	String	Read / Write	External Time Interval Code as estimated and used by the rating process. Sample mapping given in the script (Can be customized as per user's requirement): 20012 - O 20021 - I 20031 - P 20032 - O 20033 - O 20001 - P 0002 - S 20003 - S 20004 - S 20011 - P
DETAIL.ASS_CBD.CP.RUM	String	Read / Write	Classifies the charging part of a call in an intercarrier relationship, for interconnection or roaming. Sample mapping given in the script (Can be customized as per user's requirement): DUR - D SND - V REC - W EVT - E AIR - D OCC - F

ISC_FirstProductRealtime

This iScript sets the validity period of products that start on first usage when they are used for the first time to rate an event in the real-time rerating pipeline.

For more information, see "Configuring Rerating To Reset First-Usage Validity Periods" in *BRM Setting Up Pricing and Rating*.

Note: To process first-usage events in the batch rating pipeline, use the FCT_FirstUsageNotify module.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

Place this iScript in the real-time rerating pipeline before the FCT_DiscountAnalysis module.

For more information, see "Function Module Dependencies".

Sample Registry

```
FirstProductRealtime
{
    ModuleName = FCT_IScript
    Module
    {
        Active = True
        Source = File
        Scripts
        {
            FirstProductRealtime
            {
                FileName = ./iScriptLib/iScriptLib_Standard/ISC_FirstProductRealtime.isc
            }
        }
    }
}
```

EDR Container Fields

Table 39–9 lists the ISC_FirstProductRealtime EDR container fields.

Table 39–9 ISC_FirstProductRealtime EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.CUST_A.INTERN_FOUND_PP_INDEX	Decimal	Read	Contains an index of the customer's purchased products that were used for rating.
DETAIL.CUST_A.PRODUCT.FIRST_USAGE_INDICATOR	Decimal	Read	Specifies whether the product is configured to start when first used and the first-usage validity period status.
DETAIL.CUST_A.ACCOUNT_PARENT_ID	String	Read	Specifies the customer account POID.
DETAIL.CUST_A.PRODUCT.OFFERING_POID	String	Read	Specifies the account's product POID.
DETAIL.CUST_A.PRODUCT.SERVICE_ID	String	Read	Specifies the product's service POID.
DETAIL.CUST_A.PRODUCT.SERVICE_TYPE	String	Read	Specifies the product's service type.
DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Specifies the EDR's start timestamp.

Table 39–9 (Cont.) ISC_FirstProductRealtime EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.EVENT_ID	Decimal	Read	Specifies the event POID.
DETAIL.EVENT_TYPE	String	Read	Specifies the event type.
DETAIL.UTC_TIME_OFFSET	String	Read	Specifies the UTC time offset.
DETAIL.REFRESH_BALANCE	Decimal	Write	Specifies whether the account's product validity period has been updated in the BRM database. If set, the discount module retrieves the updated balance information before evaluating discounts for the event.

ISC_GetCamelFlag

The ISC_GetCamelFlag iScript is used in the roaming outcollect processing to retrieve the CAMEL flag information for a roaming partner. The CAMEL flag information is used during EDR processing.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

The ISC_GetCamelFlag iScript must run before the FCT_PreRating module in the outcollect rating pipeline.

For more information, see "Function Module Dependencies".

Sample Registry

```
GetCamelFlag
{
     ModuleName = FCT_IScript
     Module
     {
          Active = True
          Source = FILE
          Scripts
          {
               CamelFlagIScript
               {
                    FileName =./iScriptLib/iScriptLib_Roaming/ISC_GetCamelFlag.isc
               }
          }
     }
}
```

ISC_GetResourceBalance

The ISC_GetResourceBalance iScript is used to get the memory balance of a resource. This iScript returns either the balance amount or '0' if the balance retrieval is successfull. If a failure occurs, it returns '-1'.

The iScript should be in the same function pool and added after FCT_ApplyBalance in the batch pipeline.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Sample Registry

```
NewBalanceAPI
{
     ModuleName = FCT_IScript
     Module
     {
          Active = True
          Source = FILE
          Scripts
          {
               NewBalanceAPI
               {
                    FileName =./iScriptLib/iScriptLib_Standard/ISC_RetrieveBalance.isc
               }
          }
     }
}
```

ISC_LeastCost

The ISC_LeastCost iScript performs the following:

- Calculates and finds the lowest charge for an EDR. See "About Least Cost Rating".
- Calculates the total savings between the charge for a promotional product and the charge for the lowest priority (base) product. See "About Calculating the Promotional Savings".

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

This module must run after FCT_CustomerRating.

For more information, see "Function Module Dependencies".

Registry Parameters

Table 39–10 lists the ISC_LeastCost registry parameters.

Table 39–10 ISC_LeastCost Registry Parameters

Registry Parameter	Description	Mandatory
Resource	Specifies the resource type that this module uses to calculate any savings amount.	Yes
Resource_ID	Specifies the resource IDs used to identify the resource used when calculating the savings amount.	Yes

Sample Registry

```
FCT_LeastCostRating
{
    ModuleName = FCT_IScript
    Module
    {
        Active = True
        Source = File
        Scripts
        {
            myScript
            {
                FileName = ./ISC_LeastCost.isc
                Resource = "Saving Charge Resource"
                Resource_ID = "1000100"
            }
        }
    }
}
```

EDR Container Fields

Table 39–11 lists the ISC_LeastCost EDR container fields.

Table 39–11 ISC_LeastCost EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE	Integer	Read/write	The amount charged to the customer.
DETAIL.ASS_CBD.DP.AMOUNT	Integer	Read	The amount of discount between the amount charged to the customer, and a greater amount that could have been charged.
DETAIL.ASS_CBD.CP.RATEPLAN_CODE	String	Read	The code identifying the least cost rating rate plan.
DETAIL.CUST_A.PRODUCT.RATEPLAN_NAME	String	Read	A description of the least cost rating rate plan code.
DETAIL.CUST_A.PRODUCT_PRIORITY	String	Read	Contains a list of the priorities for all products that are associated with the same service and event.
DETAIL.CUST_A.INTERN_RATING_PRODUCTS	String	Write	Contains the product rating indexes. This is a comma-separated list of all rating products' indexes associated with the same service and event, and their priorities.

Table 39–11 (Cont.) ISC_LeastCost EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_CBD.CP.RESOURCE	String	Write	The resource to impact for reporting promotional savings.
DETAIL.ASS_CBD.CP.RESOURCE_ID	Integer	Write	The ID of the resource to impact for promotional savings.
DETAIL.ASS_CBD.CP.RESOURCE_TYPE	Integer	Write	The savings charge packet to impact. This is **992** by default.

ISC_MapNetworkOperatorInfo

The ISC_MapNetworkOperatorInfo iScript maps the DETAIL.SOURCE_NETWORK field to the PIN_FLD_ORIGIN_NETWORK field and the DETAIL.DESTINATION_NETWORK field to the PIN_FLD_DESTINATION_NETWORK field of the opcode input block for the corresponding event.

See "Managing Number Portability" in *BRM Telco Integration*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

For more information, see "Function Module Dependencies".

Sample Registry

```
MapNetworkOperatorInfo
  {
   ModuleName = FCT_IScript
   Module
   {
     Active = TRUE
     Source = File
     Scripts
     {
       PreOpcode
       {
         FileName = ./iScriptLib/AAA/ISC_MapNetworkOperatorInfo.isc
       }
     }
   }
  }
```

ISC_Migration

Use the ISC_Migration iScript during account migration to automatically flag EDRs for suspension. The FCT_Suspense module can then route the EDRs to a separate suspense output stream.

For more information, see "Migrating accounts with the Pipeline Manager Running" in *BRM System Administrator's Guide*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Sample Registry

```
MigrationPlugIn
{
  ModuleName = FCT_IScript
  Module
  {
    Active = True
    Source = File
    Scripts
    {
      MigrationIScript
      {
        FileName = ./samples/wireless_splitter/ISC_Migration.isc
      }
    }
  }
}
```

ISC_MiscOutcollect

The ISC_MiscOutcollect iScript is used in the Outcollect rating pipeline. This module adds BASIC_SERVICE and SUPPLEMENTARY_SERVICE blocks to the EDR container for GSM services. This is done to ensure that the TAP file generated by the pipeline contains all the required information.

For GPRS services, this module adjusts the record number field in the GPRS extension block to **0**.

It also modifies the RUM field of the charge packets as follows:

- DUR is replaced by D.
- SND is replaced by V.
- REC is replaced by W.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Sample Registry

```
MiscAddInfo
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = File
      Scripts
      {
         MiscAddInfoIScript
         {
```

```
            FileName = ./iScriptLib/iScriptLib_Roaming/ISC_MiscOutCollect.isc
         }
      }
   }
}
```

EDR Container Fields

Table 39–12 lists the ISC_MiscOutcollect EDR container fields.

Table 39–12 ISC_MiscOutcollect EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_GSMW_EXT.BS_PACKET.RECORD_TYPE	String	Write	Contains the record type field.
DETAIL.ASS_GSMW_EXT.BS_PACKET.RECORD_NUMBER	Integer	Write	Contains the record number field.
DETAIL.ASS_GSMW_EXT.BS_PACKET.CHAIN_REFERENCE	String	Write	Contains the call reference.
DETAIL.ASS_GSMW_EXT.BS_PACKET.LONG_DURATION_INDICATOR	String	Write	Contains whether the call is a long duration or not.
DETAIL.ASS_GSMW_EXT.BS_PACKET.BASIC_SERVICE	String	Write	Contains the basic service of the call.
DETAIL.ASS_GSMW_EXT.BS_PACKET.QOS_REQUESTED	String	Write	Contains the QOS requested.
DETAIL.ASS_GSMW_EXT.BS_PACKET.QOS_USED	String	Write	Contains the QOS used.
DETAIL.ASS_GSMW_EXT.BS_PACKET.CHARGING_START_TIMESTAMP	Date	Write	Contains the call start time.
DETAIL.ASS_GSMW_EXT.BS_PACKET.CHARGING_END_TIMESTAMP	Date	Write	Contains the call end time.
DETAIL.ASS_GSMW_EXT.BS_PACKET.WHOLESALE_CHARGED_AMOUNT_VALUE	Decimal	Write	Contains the total charged amount value of the call.
DETAIL.ASS_GSMW_EXT.BS_PACKET.WHOLESALE_CHARGED_TAX_RATE	Decimal	Write	Contains the tax rate.
DETAIL.ASS_GSMW_EXT.BS_PACKET.SPEECH_VERSION_REQUESTED	String	Write	Contains the speech version requested.
DETAIL.ASS_GSMW_EXT.BS_PACKET.SPEECH_VERSION_USED,	String	Write	Contains the speech version used.
DETAIL.ASS_GSMW_EXT.SS_PACKET.RECORD_TYPE	String	Write	Contains the record type.
DETAIL.ASS_GSMW_EXT.SS_PACKET.RECORD_NUMBER	Integer	Write	Contains the record number.
DETAIL.ASS_GSMW_EXT.SS_PACKET.SS_EVENT	String	Write	Contains the event type.

Table 39–12 (Cont.) ISC_MiscOutcollect EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_GSMW_EXT.SS_PACKET.ACTION_CODE	String	Write	Contains the connect type.
DETAIL.ASS_CBD.CP.RUM	String	Read/Write	Contains the ratable unit of measurement.
DETAIL.ASS_GPRS_EXT.RECORD_NUMBER	Integer	Write	Contains the record number.

ISC_Monitoring

The ISC_Monitoring iScript records latencies for authentication, authorization, and accounting (AAA) requests.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

The ISC_Monitoring iScript depends on the ISC_StartTime iScript.

For more information, see "Function Module Dependencies".

Registry Parameters

Table 39–13 lists the ISC_Monitoring registry parameters.

Table 39–13 ISC_Monitoring Registry Parameters

Registry Parameter	Description	Mandatory
FileName	Specifies the location of the ISC_Monitoring iScript.	Yes
recordsPerFile	Specifies the number of records per event log file.	Yes
recordsPerWrite	Specifies the number of records per write operation.	Yes
eventLogDir	Specifies the directory for the event log file.	Yes

Sample Registry

```
MonitoringPlugIn
{
   ModuleName = FCT_IScript
   Module
     {
       Active = True
       Source = FILE
       Scripts
       {
          ReadIScript
          {
             FileName = ./iScriptLib/AAA/ISC_Monitoring.isc
             recordsPerFile = 1000 # number of records per event log file
             recordsPerWrite = 10 # number of records per write operation
             eventLogDir = ./log/dump # directory for series of event log file
          }
```

```
      }
   }
}
```

EDR Container Fields

Table 39–14 lists the ISC_Monitoring EDR container fields.

Table 39–14 ISC_Monitoring EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.MILLISEC_TIME	Integer	Read	Specifies the latency time in milliseconds.
DETAIL.OPCODE_NUM	Integer	Read	Number of the BRM opcode that performs the requested action.

ISC_NRTRDE_ErrorReport

The ISC_NRTRDE_ErrorReport iScript is used during roaming incollect processing by the NRTRDE processing pipeline. This iScript collects the validation errors in the EDRs and creates error records in the Pipeline Manager database. It also collects NRTRDE file processing information and creates file processing records in the Pipeline Manager database. Information stored in the validation and file processing records in the database are used for generating NRTRDE reports.

For more information, see the description on detecting roaming fraud using NRTRDE in *BRM Configuring Pipeline Rating and Discounting*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

The ISC_NRTRDE_ErrorReport iScript must run after the ISC_NrtrdeHeaderValidation iScript.

For more information, see "Function Module Dependencies".

Registry Parameters

The registry parameter is **FileName**. It is mandatory. **FileName** specifies the location of the iScript. The default location is:

`./iScriptLib/iScriptLib_Roaming/ISC_NRTRDE_ErrorReport.isc`

Sample Registry

```
NRTRDE_ErrorReport
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = FILE
```

```
      Scripts
      {
         NRTRDE_ErrorReport
        {
            FileName = ./iScriptLib/iScriptLib_Roaming/ISC_NRTRDE_ErrorReport.isc
            DatabaseConnection = ifw.DataPool.Login
        }
      }
   }
}
```

ISC_NRTRDE_EventSplit

The ISC_NRTRDE_EventSplit iScript is used by roaming outcollect processing to duplicate and route EDRs to the corresponding roaming partner's NRTRDE output streams based on the roaming partner's NRTRDE flag.

For more information, see the description on detecting roaming fraud using NRTRDE in *BRM Configuring Pipeline Rating and Discounting*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

The ISC_NRTRDE_EventSplit iScript must run after the FCT_EnhancedSplitting module in the outcollect rating pipeline.

For more information, see "Function Module Dependencies".

Registry Parameters

Table 39–15 lists the ISC_NRTRDE_EventSplit registry parameters.

Table 39–15 ISC_NRTRDE_EventSplit Registry Parameters

Registry Parameter	Description	Mandatory
FileName	Specifies the location of the ISC_NRTRDE_EventSplit iScript. The default location is: **`./iScriptLib/iScriptLib_Roaming/ISC_NRTRDE_EventSplit.isc`**	Yes
NRTRDE_STREAM_ PATTERN	Specifies the name of the NRTRDE stream pattern. This parameter appends the PLMN to the NRTRDE stream pattern to retrieve the output stream name.	Yes

Sample Registry

```
NRTRDESplit
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = File
      Scripts
      {
```

```
         NRTRDESplit
         {
            FileName = ./iScriptLib/iScriptLib_Roaming/ISC_NRTRDE_EventSplit.isc
            NRTRDE_STREAM_PATTERN = NRTRDEOutput_
         }
      }
   }
}
```

ISC_NrtrdeHeaderValidation_v2_01

The ISC_NrtrdeHeaderValidation_v2_01 iScript is used during roaming incollect processing by the NRTRDE processing pipeline. This iScript validates the information in the header record of the TD35 file based on the TD35 specifications.

For more information, see the description on detecting roaming fraud using NRTRDE in *BRM Configuring Pipeline Rating and Discounting*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

The ISC_NrtrdeHeaderValidation_v2_01 iScript must run before any other modules in the NRTRDE processing pipeline.

For more information, see "Function Module Dependencies".

Registry Parameters

The registry parameter is **FileName**. It is mandatory. **FileName** specifies the location of the iScript. The default location is:

./iScriptLib/iScriptLib_Tap3Validation/ISC_NrtrdeHeaderValidation_v2_01.isc

Sample Registry

```
TapValidationIScripts
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = FILE
      Scripts
      {
         NrtrdeHeaderValidation
         {
             FileName = ./iScriptLib/iScriptLib_Tap3Validation/ISC_NrtrdeHeaderValidation_v2_01.isc
         }
      }
   }
}
```

ISC_ObjectCacheTypeOutputSplitter

Use the ISC_ObjectCacheTypeOutputSplitter iScript to enter a value in an EDR to create two identical output files from a single input EDR and write them to separate output streams.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

To use this iScript, you must have object cache residency distinction enabled in your system.

Requires a connection to the DAT_BalanceBatch module.

For more information, see "Function Module Dependencies".

Sample Registry

```
ObjectCacheTypeOutputSplitter_Script
{
  ModuleName = FCT_IScript
  Module
  {
    Active = True
    Source = File

    Scripts
    {
      ObjectCacheTypeOutputSplit
      {
        FileName = ./iScriptLib/iScriptLib_Standard/ISC_ObjectCacheTypeOutputSplitter.isc
      }
    }
  }
}
```

ISC_OverrideRateTag

The ISC_OverrideRateTag iScript is used in the outcollect settlement pipelines to populate the RATE_TAG field with the value of the NRTRDE flag in the balance impact.

The value of the RATE_TAG field is used by high-usage reports to filter out NRTRDE operator data from the report.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

The ISC_OverrideRateTag iScript must run after the FCT_BillingRecord module in the outcollect settlement pipeline.

For more information, see "Function Module Dependencies".

Sample Registry

```
OverrideRateTag
{
     ModuleName = FCT_IScript
     Module
     {
          Active = True
          Source = FILE
          Scripts
          {
               OverrideRateTagIScript
               {
                    FileName =./iScriptLib/iScriptLib_Roaming/ISC_OverrideRateTag.isc
               }
          }
     }
}
```

ISC_OverrideSuspenseParams

The ISC_OverrideSuspenseParams iScript overrides some fields in the Suspense Extension block of the EDR container that is set by Suspense Manager.

During Outcollect processing, this iScript in the RAP Processing pipeline overrides the PIPELINE_NAME suspense field. For severe TAP records, PIPELINE_NAME is set to the name of the outcollect rating pipeline. For fatal TAP records, PIPELINE_NAME is set to the Suspense Batch output stream.

For more information, see "About Processing Rejected Outcollect TAP Files And Records" in *BRM Configuring Roaming in Pipeline Manager*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Registry Parameters

Table 39–16 lists the ISC_OverrideSuspenseParams registry parameters.

Table 39–16 ISC_OverrideSuspenseParams Registry Parameters

Registry Parameter	Description	Mandatory
TAPFilePrefix	Specifies the prefix of the TAP file. ■ **CD** for files containing chargeable data ■ **TD** for files containing test data	Yes
TAPOutCollectPipeline	Specifies the name of the outcollect rating pipeline.	Yes
TAPCorrectionPipeline	Specifies the path of the directory from which the original TAP file was sent to the network operator.	Yes
TAPSentArchivePath	Specifies the path of the directory from which the original TAP file was sent to the network operator.	Yes
SBRStream	Specifies the output stream that generates the suspense batch file.	Yes

Sample Registry

```
OverrideSuspenseParams
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = File
      Scripts
      {
         OverrideSuspenseParams
         {
            FileName = ./iScriptLib/iScriptLib_Roaming/ISC_OverrideSuspenseParams.isc
            TAPFilePrefix = CD
            TAPOutCollectPipeline = TAPOutCollectPipeline
            TAPCorrectionPipeline = TAPCorrectionPipeline

            #the following path must be absolute
            TAPSentArchivePath = ./data/outcollect/tapout/sent
            SBRStream = SBROutput
         }
      }
   }
}
```

EDR Container Fields

Table 39–17 lists the ISC_OverrideSuspenseParams EDR container fields.

Table 39–17 ISC_OverrideSuspenseParams EDR Container Fields

Alias field name **Default field name**	**Type**	**Access**	**Description**
DETAIL.ASS_ROAMING_EXT.TAP_FILE_SEQ_NO	Integer	Read	Sequence number of the TAP file.
DETAIL.ASS_ROAMING_EXT.RAP_RECORD_TYPE	String	Read	RAP record type.
DETAIL.BATCH_ID	String	Write	Records the TAP batch ID.
DETAIL.ORIGINAL_BATCH_ID	String	Write	Records the TAP batch ID.
DETAIL.ASS_SUSPENSE_EXT.SUSPENDED_FROM_BATCH_ID	String	Write	Records the TAP batch ID.
DETAIL.ASS_SUSPENSE_EXT.SOURCE_FILENAME	String	Write	Records the TAP file name.
DETAIL.ASS_SUSPENSE_EXT.PIPELINE_NAME	String	Write	For a severe error, records the TAP outcollect rating pipeline name. For a fatal error, records the TAP correction pipeline name.

Table 39–17 (Cont.) ISC_OverrideSuspenseParams EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_SUSPENSE_EXT.RECYCLE_KEY	String	Write	Records the RAP file sequence number.
DETAIL.ASS_ROAMING_EXT.TAP_FILE_PATH	String	Write	The path of the directory from which the original TAP file was sent to the network operator (Only for fatal errors).
DETAIL.ASS_ROAMING_EXT.SUSPENSION_TIME	Date	Write	(Only for fatal errors) Suspension time of the TAP file.

ISC_PopulateOpcodeandUtilBlock_Diameter

The ISC_PopulateOpcodeandUtilBlock_Diameter iScript adds an opcode block in the EDR. In the Dispatcher pipeline, the EDR is duplicated to handle failover. The ISC_PopulateOpcodeandUtilBlock_Diameter iScript improves the performance by reducing the time taken to duplicate the EDR. Also, it provides the flexibility of performing minor validations. This iScript populates the opcode block and other relevant fields like OPCODE_NODE and OPCODE_NUMBER, which are required by the TimeoutRouter pipeline.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

This iScript must be called before calling FCT_Opcode.

For more information, see "Function Module Dependencies".

Sample Registry

```
ProcessPipeline
  {
    PopulateOpcodeAndUtilBlock
  {
    ModuleName? = FCT_IScript
    Module
     {
       Active = TRUE
       Source = File
       Scripts
       {
         PreOpcode?
         {
           FileName? = ./iScriptLib/AAA/ISC_PopulateOpcodeAndUtilBlock.isc
      }
   }
}
```

ISC_PostRating

The ISC_PostRating iScript adds all the retail and wholesale charges and puts them in the DETAIL.RETAIL_CHARGED_AMOUNT_VALUE and DETAIL.WHOLESALE_CHARGED_AMOUNT_VALUE fields.

See "Billing Consolidation with CIBER Roaming and Revenue Assurance".

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

This module must run:

- After rating modules FCT_CustomerRating, FCT_PreRating, and FCT_MainRating

  or

- After the FCT_ExchangeRate module

For more information, see "Function Module Dependencies".

Sample Registry

```
PostRating
{
    ModuleName = FCT_IScript
    Module
    {
        Active = True
        Source = File

        Scripts
        {
           PostRating
           {
              FileName = ./ISC_PostRating.isc

              RetailRecordType = 981
              RetailResource = DEM
              RetailPricemodelType = S
              RetailCurrencyType = R

              WholesaleRecordType = 990
              WholesaleResource = DEM
              WholesalePricemodelType = S
              WholesaleCurrencyType = R
           }
        }
    }
}
```

EDR Container Fields

Table 39–18 lists the ISC_PostRating input EDR container fields.

Table 39–18 ISC_PostRating Input EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_CBD.RECORD_TYPE	String	N/A	Contains the charge breakdown record type (retail or wholesale).
DETAIL.ASS_CBD.CP.RESOURCE	String	N/A	Contains the charge breakdown resource type.
DETAIL.ASS_CBD.CP.PRICEMODEL_TYPE	String	N/A	Contains the charge breakdown price model.
DETAIL.ASS_CBD.CP.CHARGED_CURRENCY_TYPE	String	N/A	Contains the charge breakdown currency type for charged amount.
DETAIL.ASS_CBD.CP.IMPACT_CATEGORY	String	N/A	Contains the charge breakdown impact category.
DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_CURRENCY	String	N/A	Contains the charge breakdown currency type for charged amount.
DETAIL.ASS_CBD.CP.CHARGED_TAX_TREATMENT	String	N/A	Contains the charge breakdown tax treatment type.
DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE	Decimal	N/A	Contains the charge breakdown charged amount.

Table 39–19 lists the ISC_PostRating output EDR container fields.

Table 39–19 ISC_PostRating Output EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.RETAIL_IMPACT_CATEGORY	String	N/A	Contains the retail impact category.
DETAIL.RETAIL_CHARGED_AMOUNT_VALUE	Decimal	N/A	Contains the retail charged amount value.
DETAIL.RETAIL_CHARGED_AMOUNT_CURRENCY	String	N/A	Contains the retail charged amount currency.
DETAIL.RETAIL_CHARGED_TAX_TREATMENT	String	N/A	Contains the retail charged amount tax treatment.
DETAIL.WHOLESALE_IMPACT_CATEGORY	String	N/A	Contains the wholesale impact category.
DETAIL.WHOLESALE_CHARGED_AMOUNT_VALUE	Decimal	N/A	Contains the wholesale charged amount value.
DETAIL.WHOLESALE_CHARGED_AMOUNT_CURRENCY	String	N/A	Contains the wholesale charged amount currency.
DETAIL.WHOLESALE_CHARGED_TAX_TREATMENT	String	N/A	Contains the wholesale charged amount tax treatment.

ISC_ProfileAnalyzer

For each EDR, the ISC_ProfileAnalyzer iScript compares the telephone numbers or other relevant data in EDRs with the ERAs that the customer's service owns. ISC_ProfileAnalyzer stores each ERA label that matches the relevant EDR container field.

The iScript adds the label names to the EDR container field DETAIL.PROFILE_LABEL_LIST. If there are multiple names, the names are separated by a comma by

default. You can change the separator character in the registry. If a label name contains a comma, you need to change the separator character.

See "Pipeline rating for friends and family ERAs" in *BRM Setting Up Pricing and Rating*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

ISC_ProfileAnalyzer depends on the ERA values populated by FCT_Account. It must be run after FCT_Account and before any rating modules in the pipeline.

For more information, see "Function Module Dependencies".

Sample Registry

```
ProfileAnalyzer
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = FILE
      Scripts
      {
        ProfileAnalyzer
        {
        ServiceType = TEL
        ProfileName = FRIENDS_FAMILY
        LabelSeparator = ,
        FileName = ./iScriptLib/iScriptLib_Standard/ISC_ProfileAnalyzer.isc
        }
     }
   }
}
```

EDR Container Fields

The ISC_ProfileAnalyzer iScript uses the EDR container fields listed in Table 39–20:

Table 39–20 ISC_ProfileAnalyzer EDR Container Fields

Alias field name **Default field name**	**Type**	**Access**	**Description**
PROFILE_LABEL_LIST DETAIL. PROFILE_LABEL_LIST	String	Write	Contains unique profile labels when the profile attributes match an EDR container field used for comparison to find F&F list.
DETAIL.CUST_A.PRODUCT. ERA.LABEL	String	Read	Contains a label associated with the service: for example, MYFAMILY.
DETAIL.CUST_A.SHARED_PROFILE_LIST. ERA.LABEL	String	Read	Contains a label associated with the service profile from a shared profile.
ERA.PROFILE	String	Read	Contains the profile name: for example, "Friends&Family."

Table 39–20 (Cont.) ISC_ProfileAnalyzer EDR Container Fields

Alias field name Default field name	Type	Access	Description
CUST_A. SHARED_PROFILE_LIST	Block	Read	Contain the profiles that the service owns or shares as a member of a profile sharing group.
CUST_B.SHARED_PROFILE_LIST	Block	Read	Contain the profiles that the service owns or shares as a member of a profile sharing group.
CUST_A. SHARED_PROFILE_LIST .ERA	Block	Read	Contains shared ERA information.
CUST_B.SHARED_PROFILE_LIST.ERA	Block	Read	N/A
ERA.NAME	String	Read	Contains the profile attribute name.
ERA.VALUE	String	Read	Contains the profile attribute value.

ISC_ProfileLabel

The ISC_ProfileLabel iScript is used when rating CDRs based on ERAs. It determines whether the ERA profiles specified in the **ProfileName** registry entry match the EDR field value and populates the DETAIL.PROFILE_LABEL_LIST field with the ERA labels of the matching ERAs, and the DETAIL.USAGE_TYPE field with appropriate usage type for the ERA.

For more information, see "Improving Pipeline Rating Performance For Events With ERAs" in *BRM Setting Up Pricing and Rating*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

The ISC_ProfileLabel iScript must run after the FCT_Account module and before any rating modules.

Requires a connection to DAT_AccountBatch. This iScript works only in batch rating.

For more information, see "Function Module Dependencies".

Registry Parameters

Table 39–21 lists the ISC_ProfileLabel registry parameters.

Table 39–21 ISC_ProfileLabel Registry Parameters

Registry Parameter	Description	Mandatory
ProfileName	Specifies the name of the ERA profile to analyze.	Yes
LabelSeparator	ERA labels in the DETAIL.PROFILE_LABEL_LIST EDR field are separated using this delimiter. The default is a comma (,).	No
FileName	Specifies the location of the ISC_ProfileLabel iScript. The default location is **./iScriptLib/iScriptLib_Standard/ISC_ProfileLabel.isc**.	Yes

Sample Registry

```
ProfileLabel
{
  ModuleName = FCT_IScript
  Module
  {
    Active = True
    Source = FILE
    Scripts
    {
      ProfileLabel
      {
      ProfileName = FRIENDS_FAMILY
      LabelSeparator = ,
      FileName = ./iScriptLib/iScriptLib_Standard/ISC_ProfileLabel.isc
      }
    }
  }
}
```

EDR Container Fields

Table 39–22 list the ISC_ProfileLabel EDR container fields.

Table 39–22 ISC_ProfileLabel EDR Container Fields

Alias field name Default field name	Type	Access	Description
CUST_A_INTERN_PP_INDEX DETAIL.CUST_A.INTERN_FOUND_PP_INDEX	Integer	Read	Contains an index of the customer's purchased products identified by the FCT_Account module.
CUST_A.ACCOUNT_PARENT_ID DETAIL.CUST_A.ACCOUNT_PARENT_ID	String	Read	Account ID of the service for which usage is getting rated.
CUST_A.PRODUCT.SERVICE_ID DETAIL.CUST_A.PRODUCT.SERVICE_ID	String	Read	Service ID for which usage is getting rated.
B_NUMBER DETAIL.B_NUMBER	String	Read	Called number of the event.
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the event's starting timestamp. The timezone information of this timestamp is stored in the BDR_UTC_TIME_OFFSET field.
BDR_UTC_TIME_OFFSET DETAIL.UTC_TIME_OFFSET	String	Read	Contains the UTC time offset that normalizes the charging start timestamp to the UTC tim zone. All validity timestamps in the BRM customer data are stored in normalized UTC time. The format is +/– *HHMM*.
PROFILE_LABEL_LIST DETAIL.PROFILE_LABEL_LIST	String	Write	Contains ERA labels that contain a value that matches the EDR field value.
USAGE_TYPE DETAIL.USAGE_TYPE	String	Write	Contains the usage type for the ERA.

ISC_RAP_0105_InMap

The ISC_RAP_0105_InMap iScript copies TAP data from staging fields in the EDR container to business fields in the EDR container. This iScript is used during roaming outcollect processing. See "About Processing Visiting Subscribers' Roaming Usage" in *BRM Configuring Roaming in Pipeline Manager* for more information.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

This should be the first module in the **FunctionPool**.

For more information, see "Function Module Dependencies".

Sample Registry

```
BusinessMapping
{
  ModuleName = FCT_IScript
  Module
  {
     Active = True
     Source = File
     Scripts
     {
       BusinessMapping
       {
           FileName = ./iScriptLib/iScriptLib_Roaming/ISC_RAP_0105_InMap.isc
       }
     }
  }
}
```

ISC_RemoveASSCBD

The ISC_RemoveASSCBD iScript is used in the Outcollect rating pipeline to remove associated charge breakdown packets associated with RAP records that are recycled to the pipeline during RAP file processing.

For more information, see "About Processing Rejected Outcollect TAP Files And Records" in *BRM Configuring Roaming in Pipeline Manager*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Sample Registry

```
RemoveASSCBD
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
```

```
      Source = File
      Scripts
      {
         RemoveASSCBDIScript
         {
            FileName = ./iScriptLib/iScriptLib_Roaming/ISC_RemoveASSCBD.isc
         }
      }
   }
}
```

EDR Container Fields

Table 39–23 lists the ISC_RemoveASSCBD EDR container fields.

Table 39–23 ISC_RemoveASSCBD EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_CBD.RECORD_NUMBER	Integer	Read/Write	Contains the record number.

ISC_RollbackSettlement

During roaming incollect and outcollect settlement processing, the ISC_RollbackSettlement iScript checks for errors in the EDR. When there is an error, it notifies the Transaction Manager to roll back the transactions in the settlement pipeline. The Transaction Manager then notifies the FCT_BatchSuspense module to suspend the entire input file.

For information, see "About processing rejected outcollect TAP files and records" in *BRM Configuring Roaming in Pipeline Manager* and "About Settling Roaming Charges" in *BRM Configuring Roaming in Pipeline Manager*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Sample Registry

```
RollbackSettlement
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = File
      Scripts
      {
         RollbackIScript
         {
            FileName = ./iScriptLib/iScriptLib_Roaming/ISC_RollbackSettlement.isc
         }
      }
   }
}
```

ISC_SetAndValidateBatchInfo

The ISC_SetAndValidateBatchInfo iScript populates and validates the batch related fields for the EDR container.

The iScript validates the HEADER.BATCH_ID. If it does not exist, the entire batch is rejected. If it exists, then it copies the HEADER.BATCH_ID to DETAIL.BATCH_ID.

If DETAIL.EVENT_ID is missing in an EDR, the EDR is rejected.

See "Using iScripts To Derive Grouping Fields" in *BRM Collecting Revenue Assurance Data*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

This iScript must be placed at the beginning of the pipeline so that the batch ID is inserted before any further processing of the mediation batches. It should be used only if you do not use Suspense Manager.

For more information, see "Function Module Dependencies".

Registry Entries

Table 39–24 lists the ISC_SetAndValidateBatchInfo registry entries.

Table 39–24 ISC_SetAndValidateBatchInfo Registry Entries

Entry	Description	Mandatory
ValidateOriginalBatchId	■ If **True**, and if DETAIL.ORIGINAL_BATCH_ID is missing, the EDR is rejected. ■ If **False**, it copies the HEADER.BATCH_ID in DETAIL.ORIGINAL_BATCH_ID.	Yes
KeepBatchIds	■ If **True**, this iScript does not modify the values in DETAIL.ORIGINAL_BATCH_ID and DETAIL.BATCH_ID. ■ If **False**, and if **ValidateOriginalBatchId** is **True**, this iScript assigns the values in HEADER.BATCH_ID to DETAIL.BATCH_ID. ■ If **False**, and if **ValidateOriginalBatchId** is **False**, this iScript assigns the value in HEADER.BATCH_ID to DETAIL.ORIGINAL_BATCH_ID and DETAIL.BATCH_ID.	Yes

Sample Registry

```
SetAndValidateBatchInfo
   {
    ModuleName = FCT_IScript
     Module
     {
       Active = True
       Source = FILE
       Scripts
        {
          SetAndValidateBatchInfo
        {
          FileName = ./iScriptLib/iScriptLib_Standard/ISC_SetAndValidateBatchInfo.isc
```

```
          ValidateOriginalBatchId = TRUE
          KeepBatchIds = TRUE
        }
      }
    }
}
```

ISC_SetEDRStatus

The ISC_SetEDRStatus iScript sets the EDR status to **Success**, **Suspense**, **Duplicate**, **Discard**, or **Skipped** for each EDR.

See "Using iScripts To Derive Grouping Fields" in *BRM Collecting Revenue Assurance Data*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

This iScript must be used before FCT_AggreGate and before the scenario that collects audit data grouped on the EDRStatus field.

For more information, see "Function Module Dependencies".

Sample Registry

```
SetEDRStatus
  {
    ModuleName = FCT_IScript
     Module
      {
       Active = True
        Source = FILE
         Scripts
      {
        SetEDRStatus
      {
        FileName = ./iScriptLib/iScriptLib_Standard/ISC_SetEDRStatus.isc
      }
    }
  }
}
```

ISC_SetOutputStream

The ISC_SetOutputStream iScript sets the Output Stream to **TelOut**, **SMSOut**, **GPRSOut**, **RejectOut**, or **DuplicateOut** for each EDR.

See "Using iScripts To Derive Grouping Fields" in *BRM Collecting Revenue Assurance Data*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

This iScript must be used after FCT_Reject, because it is dependent on the fields set in FCT_Reject, and before the scenario that collects audit data grouped on the OutputStream field.

For more information, see "Function Module Dependencies".

Sample Registry

```
SetOutputStream
  {
   ModuleName = FCT_IScript
   Module
  {
    Active = True
    Source = FILE
    Scripts
  {
     SetOutputStream
  {
      FileName = ./iScriptLib/iScriptLib_RevenueAssurance/ISC_SetOutputStream.isc
    }
  }
  }
}
```

ISC_SetRevenueFigures

The ISC_SetRevenueFigures iScript collects the previous and current charged and discount amount for a configured resource ID.

See "Using iScripts To Derive Grouping Fields" in *BRM Collecting Revenue Assurance Data*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

This iScript must run after rating and discounting and before FCT_AggreGate.

For more information, see "Function Module Dependencies".

Registry Parameters

Table 39–25 lists the ISC_SetRevenueFigures registry parameter.

Table 39–25 ISC_SetRevenueFigures Registry Parameter

Registry Parameter	Description	Mandatory
ResourceId	Specifies the resource for which you want to calculate the discount value.	Yes

Sample Registry

```
SetRevenueFigures
```

```
{
 ModuleName = FCT_IScript
 Module
 {
  Active = True
  Source = FILE
  Scripts
  {
   SetRevenueFigures
   {
   FileName = ./iScriptLib/iScriptLib_Standard/ISC_SetRevenueFigures.isc
   ResourceId = 978
   }
  }
 }
}
```

ISC_SetRevenueStream

The ISC_SetRevenueStream iScript sets the Revenue Stream to **Retail**, **Wholesale**, **Roaming**, or **Unknown** for each EDR.

See "Using iScripts To Derive Grouping Fields" in *BRM Collecting Revenue Assurance Data*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

This iScript must be used before FCT_AggreGate and after post rating (after the EDRs are rated).

For more information, see "Function Module Dependencies".

Sample Registry

```
SetRevenueStream
{
  ModuleName = FCT_IScript
  Module
 {
  Active = True
  Source = FILE
  Scripts
  {
   SetRevenueStream
   {
    FileName = ./iScriptLib/iScriptLib_Standard/ISC_SetRevenueStream.isc
   }
  }
 }
}
```

ISC_SetSvcCodeRTZoning

For each EDR, the ISC_SetSvcCodeRTZoning iScript finds the service type and updates DETAIL.INTERN_SERVICE_CODE EDR field with the customized service code value.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Sample Registry

```
SetSvcCodeRTZoning
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = FILE
      Scripts
      {
         SetSvcCodeRTZoning
         {
            FileName = ./iScriptLib/iScriptLib_Standard/ISC_SetSvcCodeRTZoning.isc
         }
      }
   }
}
```

EDR Container Fields

Table 39–26 lists the ISC_SetSvcCodeRTZoning EDR container fields.

Table 39–26 ISC_SetSvcCodeRTZoning EDR Container Field

Alias field name Default field name	Type	Access	Description
DETAIL.INTERN_SERVICE_CODE	String	Write	Contains the service type of the current EDR.

ISC_StartTime

The ISC_StartTime iScript is used to request the start time. For example, the ISC_Monitoring iScript uses the start time provided by ISC_StartTime for measuring latencies for authentication, authorization, and accounting (AAA) requests.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Sample Registry

```
StartTime
{
  ModuleName = FCT_IScript
  Module
  {
```

```
    Active = True
    Source = FILE
    Scripts
    {
      StartTime
      {
      FileName = ./iScriptLib/iScriptLib_Standard/ISC_StartTime.isc
      }
    }
  }
}
```

EDR Container Fields

Table 39–27 lists the ISC_StartTime EDR container fields.

Table 39–27 ISC_StartTime EDR Container Field

Alias field name Default field name	Type	Access	Description
DETAIL.MILLISEC_TIME	Integer	Read	Specifies the latency time in milliseconds.

ISC_TapDetailValidation_v3_12

The ISC_TapDetailValidation_v3_12 iScript validates that the fields present in the detail record of the EDR container contain valid data.

> **Note:** The ISC_TapDetailValidation_v3_12 iScript is a deprecated module but remains in BRM for backward compatibility.

You run the ISC_TapDetailValidation_v3_12 iScript when incoming files that you receive from your roaming partner use the TAP format. When processing the incoming TAP files, BRM converts input data from TAP fields into corresponding fields of the EDR container description file.

For general information on validating TAP records, see "About Validating Roaming Usage Data" in *BRM Configuring Roaming in Pipeline Manager*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

Because an erroneous TAP record can be discarded before the record is split into multiple records, this module must run before the FCT_Discard and ISC_TapSplitting modules.

For more information, see "Function Module Dependencies".

Sample Registry

```
ISC_TapDetailValidation_v3_12
{
  ModuleName = FCT_IScript
```

```
  Module
  {
     Active = True
     Source = File
     Scripts
     {
       TapDetailValidation_v3_12
       {
           FileName = ./iScriptLib/iScriptLib/ISC_TapDetailValidation_v3_12.isc
       }
     }
  }
}
```

EDR Container Fields

Table 39–28 lists the ISC_TapDetailValidation_v3_12 EDR container fields.

Table 39–28 ISC_TapDetailValidation_v3_12 EDR Container Fields

Alias field name Default field name	Type	Access	Description
SERVER_TYPE_OF_NUMBER DETAIL.ASS_CAMEL_EXT.SERVER_ TYPE_OF_NUMBER	Integer	Read	CAMEL Invocation Fee TD57 item. Performs number-normalization.
CHARGEABLE_QUANTITY_VALUE DETAIL.ASS_CBD.CP.CHARGEABLE_ QUANTITY_VALUE	Decimal	Read	Chargeable Units TD57 item. Indicates the number of chargeable units. The value should be equal to or greater than zero.
IMPACT_CATEGORY DETAIL.ASS_CBD.CP.IMPACT_ CATEGORY	String	Read	Charge Type TD57 item. Identifies the type of charge. Possible values: ■ **00** Total charge for Charge Information (the invoiceable value) ■ **01** Airtime charge ■ **02** reserved ■ **03** Toll charge ■ **04** Directory assistance ■ **05 – 20** reserved ■ **21** VPMN surcharge ■ **69 – 99** reserved
TAX_TYPE DETAIL.ASS_CBD.TP.TAX_TYPE	String	Read	Tax Type TD57 item. Indicates the type of tax. Possible values: ■ **01** National ■ **02** Regional ■ **03** County ■ **04** Local/City

Table 39–28 (Cont.) ISC_TapDetailValidation_v3_12 EDR Container Fields

Alias field name Default field name	Type	Access	Description
VOLUME_RECEIVED DETAIL.VOLUME_RECEIVED	Decimal	Read	Data Volume Incoming TD57 item. Identifies the number of incoming octets (bytes) within an occurrence of GPRS Service Used or Content Service Used. The value should be equal to or greater than zero.
VOLUME_SENT DETAIL.VOLUME_SENT	Decimal	Read	Data Volume Outgoing TD57 item. Identifies the number of outgoing octets (bytes) within an occurrence of GPRS Service Used or Content Service Used. The value should be equal to or greater than zero.
WHOLESALE_IMPACT_CATEGORY DETAIL.WHOLESALE_IMPACT_CATEGORY	String	Read	Charge Type TD57 item. Identifies the wholesale type of charge.

ISC_TapHeaderTrailerValidation_v3_12

The ISC_TapHeaderTrailerValidation_v3_12 iScript validates that fields present in the header and trailer records of the EDR contain valid data.

Note: The ISC_TapHeaderTrailerValidation_v3_12 iScript is a deprecated module but remains in BRM for backward compatibility.

You run the ISC_TapHeaderTrailerValidation_v3_12 iScript when incoming files that you receive from your roaming partner use the TAP format. When processing the incoming TAP files, BRM converts input data from TAP fields into corresponding fields of the EDR container description file.

For general information on validating TAP records, see "About Validating Roaming Usage Data" in *BRM Configuring Roaming in Pipeline Manager*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

Because an erroneous TAP record can be discarded before the record is split into multiple records, this module must run before the FCT_Discard and ISC_TapSplitting modules.

For more information, see "Function Module Dependencies".

Sample Registry

```
ISC_TapHeaderTrailerValidation_v3_12
{
```

```
  ModuleName = FCT_IScript
  Module
  {
     Active = True
     Source = File
     Scripts
     {
       TapHeaderTrailerValidation_v3_12
       {
            FileName = ./iScriptLib/iScriptLib/ISC_TapHeaderTrailerValidation_v3_12.isc
       }
     }
  }
}
```

EDR Container Fields

Table 39–29 lists the ISC_TapHeaderTrailerValidation_v3_12 EDR container fields.

Table 39–29 ISC_TapHeaderTrailerValidation_v3_12 EDR Container Fields

Alias field name Default field name	Type	Access	Description
CHARGE_REFUND_INDICATOR DETAIL.ASS_CONT_EXT.SERVICE_USED_ LIST.CHARGE_REFUND_INDICATOR	Integer	Read	Charge Refund Indicator TD57 item. Indicates that Content Transaction is a refund. When present, changes the signs of any revenue within Content Service Used. **Value:** **1** Refund
TOTAL_CHARGE_REFUND TRAILER.TOTAL_CHARGE_VALUE_ LIST.TOTAL_CHARGE_REFUND	String	Read	Total Charge Refund TD57 item. Specifies the sum of all the charges associated with Charge Type when Charge Type represents a refund. The value must be greater than zero.
TOTAL_NUMBER_OF_RECORDS TRAILER.TOTAL_NUMBER_OF_RECORDS	Integer	Read	Specifies the total number of Basic Records in the file, excluding header and trailer. Used as a check value to determine that all records have been correctly transmitted or used.

ISC_TapSplitting

The ISC_TapSplitting iScript splits mobile originating and terminating EDRs when the CDR contains more than one basic service. ISC_TapSplitting creates a new EDR for each additional basic service.

For information about splitting MOC and MTC records, see "Generating Multiple TAP MOC and MTC Records".

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

Must run after the following modules:

- FCT_DuplicateCheck
- FCT_Discard

For more information, see "Function Module Dependencies".

Sample Registry

```
ISC_TapSplitting
{
  ModuleName = FCT_IScript
  Module
  {
     Active = True
     Source = File
     Scripts
     {
       TapSplitting
       {
            FileName = ./iScriptLib/iScriptLib/ISC_TapSplitting.isc
       }
      }
  }
}
```

ISC_TaxCalc

The ISC_TaxCalc iScript applies a flat tax to pipeline-rated events.

For information about calculating flat taxes, see "About Pipeline Taxation" in *BRM Calculating Taxes*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

Run this module after the FCT_MainRating module, but before the FCT_BillingRecord module.

For more information, see "Function Module Dependencies".

Registry Parameters

Table 39–30 lists the ISC_TaxCalc registry parameters.

Table 39–30 ISC_TaxCalc Registry Parameters

Registry Parameter	Description	Mandatory
FlatTaxRate	Specifies the default flat tax percentage. Pipeline Manager applies this tax rate when an event does not match any criteria in the **taxcodes_map** file. For example, set **FlatTaxRate** to **5** to apply a 5% tax on the charged amount.	Yes
TaxCode	Specifies the default tax code. Pipeline Manager uses this tax code when an event does not match any criteria in the **taxcodes_map** file.	Yes
TaxCodeMapFilePath	Specifies the path to the **taxcodes_map** file.	Yes
TaxType	Specifies the default tax type. Pipeline Manager applies this tax type when an event does not match any criteria in the **taxcodes_map** file.	Yes

Sample Registry

```
TaxPlugin
{
  ModuleName = FCT_IScript
  Module
  {
     Active = True
     Source = File
     Scripts
     {
         Tax
            FlatTaxRate = 5
            TaxCode = FLAT
            TaxType = 16
            TaxCodeMapFilePath = BRM_Home/sys/cm/taxcodes_map
            Filename = ./iScriptLib/ISC_TaxationLib/ISC_TaxCalc.isc
     }
  }
}
```

ISC_TAP_0312_Include

The ISC_TAP_0312_Include iScript copies TAP data from staging fields in the EDR container to business fields in the EDR container.

For more information about TAP data mapping, see:

- "About Processing Home Subscribers' Roaming Usage" in *BRM Configuring Roaming in Pipeline Manager*.
- "About Processing Visiting Subscribers' Roaming Usage" in *BRM Configuring Roaming in Pipeline Manager*

This iScript is part of the following iScripts:

- ISC_TAP_0312_InMap
- ISC_RAP_0105_InMap

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Sample Registry

```
BusinessMapping
{
  ModuleName = FCT_IScript
  Module
  {
     Active = True
     Source = File
     Scripts
     {
       BusinessMapping
       {
           FileName = ./iScriptLib/iScriptLib_Roaming/ISC_TAP_0312_Include.isc
       }
     }
  }
}
```

EDR Container Fields

Table 39–31 lists the ISC_TAP_0312_Include EDR container fields.

Table 39–31 ISC_TAP_0312_Include EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_LTE_EXT.REQUESTEDNUMBER	String	Read	Contains the number (TEL URI of the original destination) to which the customer requested to be connected.
DETAIL.ASS_LTE_ EXT.REQUESTEDPUBLICUSERID	String	Read	Contains the public user ID (SIP URI of the original destination) to which the customer requested to be connected.

ISC_TAP_0312_InMap

The ISC_TAP_0312_InMap iScript copies TAP data from staging fields in the EDR container to business fields in the EDR container.

For more information about TAP data mapping, see:

- "About Processing Home Subscribers' Roaming Usage" in *BRM Configuring Roaming in Pipeline Manager*.
- "About Processing Visiting Subscribers' Roaming Usage" in *BRM Configuring Roaming in Pipeline Manager*

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Registry Entries

Table 39–32 lists the ISC_TAP_0312_inMap registry entries.

Table 39–32 ISC_TAP_0312_InMap Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive: **True** = Active **False** = Inactive	Yes
Source	Specifies the source of the iScripts: ■ **File** ■ **Database**	Yes
FileName	Specifies the location of the iScript. The default location is **./iScriptLib/iScriptLib_Roaming/ISC_TAP_0312_InMap.isc**.	Yes

Sample Registry

```
BusinessMapping
{
  ModuleName = FCT_IScript
  Module
  {
     Active = True
     Source = File
     Scripts
     {
       BusinessMapping
       {
           FileName = ./iScriptLib/iScriptLib_Roaming/ISC_TAP_0312_InMap.isc
       }
     }
  }
}
```

ISC_TAP_0312_Validations

The ISC_TAP_0312_Validations iScript validates TAP input data.

For general information on validating TAP files and records, see "About Validating Roaming Usage Data" in *BRM Configuring Roaming in Pipeline Manager*.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Registry Entries

Table 39–33 lists the ISC_TAP_0312_Validations registry entries.

Table 39–33 ISC_TAP_0312_Validations Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive	Yes
Source	Specifies the source of the iScripts. ■ **File** ■ **Database**	Yes
FileName	Specifies the location of the iScript. The default location is **./iScriptLib/iScriptLib_Roaming/ISC_TAP_0312_Validations.isc**.	Yes

Sample Registry

```
TAP3Validations
{
  ModuleName = FCT_IScript
  Module
  {
     Active = True
     Source = File
     Scripts
     {
       TAP3Validations
       {
           FileName = ./iScriptLib/iScriptLib_Roaming/ISC_TAP_0312_Validations.isc       }
     }
  }
}
```

ISC_UsageClassSetting

The ISC_UsageClassSetting iScript is used in the Incollect and Outcollect settlement pipelines to populate the DETAIL.USAGE_CLASS field with the value of DETAIL.CONNECT_TYPE in the EDR container, which specifies the type of call. The value of DETAIL.USAGE_CLASS is stored in the **event_dlay_sess_tlcs** table and is later used by high-usage reports to determine premium calls.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Sample Registry

```
UsageClassMap
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = File
      Scripts
      {
         UsageClassSettingIScript
         {
```

```
            FileName = ./iScriptLib/iScriptLib_Roaming/ISC_UsageClassSetting.isc
         }
      }
   }
}
```

EDR Container Fields

Table 39–34 lists the ISC_UsageClassSetting EDR container fields.

Table 39–34 ISC_UsageClassSetting EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.USAGE_CLASS	String	Write	Internal usage class.
DETAIL.CONNECT_TYPE	String	Read	Contains the connect type of the call.

UpdateTapInfo_StopRapout

The UpdateTapInfo_StopRapout iScript updates the database with information on the Stop Return RAP file sent to the Visited Public Mobile Network (VPMN) operator.

This iScript is part of the Stop RAP Generator pipeline. To use the UpdateTapInfo_StopRapout script, configure the Stop RAP Generator pipeline. See the discussion of configuring the Stop RAP Generator pipeline in *BRM Configuring Roaming in Pipeline Manager* for more information.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

None

Sample Registry

```
UpdateTapInfo_StopRapout
{
  ModuleName = FCT_IScript
  Module
  {
    Active = True
    Source = FILE
    Scripts
    {
      UpdateTapInfo_StopRapout
      {
        FileName = ./iScriptLib/iScriptLib_Roaming/ISC_UpdateTapInfo_StopRapout.isc
        DatabaseConnection = ifw.DataPool.Login
      }
    }
  }
}
```

UpdateTapInfo_Tapin

The UpdateTapInfo_Tapin iScript updates the information in the database on incoming TAP files for use in generating Stop Return RAP files.

This iScript is part of the validation pipeline. To use the UpdateTapInfo_Tapin iScript, configure the validation pipeline. See the discussion of configuring the validation pipeline in *BRM Configuring Roaming in Pipeline Manager* for more information.

To run the iScript, configure the FCT_IScript module. See:

- FCT_IScript
- "About Configuring iScripts" in *BRM System Administrator's Guide*

Dependencies

This iScript should be configured before the FCT_Reject module, and after any iScripts that populate the mandatory fields in the header of an EDR.

For more information, see "Function Module Dependencies".

Sample Registry

```
UpdateTapInfo_Tapin
{
  ModuleName = FCT_IScript
  Module
  {
    Active = True
    Source = FILE
    Scripts
    {
      UpdateTapInfo_Tapin
      {
        FileName = ./iScriptLib/iScriptLib_Roaming/ISC_UpdateTapInfo_Tapin.isc
        DatabaseConnection = ifw.DataPool.Login
      }
    }
  }
}
```

40

Pipeline Manager Input and Output Modules

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager input and output modules.

EXT_InEasyDB

The EXT_InEasyDB module handles pipeline input from a database. See:

- About Getting Pipeline Input from a Database
- Configuring EDR Input Processing

This module runs automatically when you start Pipeline Manager.

Configure this module as a submodule of the INP_GenericStream module. See "INP_GenericStream".

To configure input from files, see "EXT_InFileManager".

Dependencies

Requires a connection to the Pipeline Manager database.

Registry Entries

Table 40–1 lists the EXT_InEasyDB registry entries.

Table 40–1 EXT_InEasyDB Registry Entries

Entry	Description	Mandatory
ControlPath	Specifies the path for SQL, parameter, job and restart files. **./database/Oracle/Scripts/Suspense**	Yes
DataConnection	Specifies the connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
FieldDelimiter	Specifies the character that separates the EDR fields.	Yes
FileName	Specifies the job and restart file name prefix.	Yes
InputDirEmptyTime	Specifies the time period (in seconds), the input directory must be empty before the EVT_INPUT_DIR_EMPTY event is sent.	No
InputPrefix	Specifies the prefix of the stream/output file name.	No
InputSuffix	Specifies the flag which tells the stream to generate date and time information in the stream/output file name.	Yes

Table 40–1 (Cont.) EXT_InEasyDB Registry Entries

Entry	Description	Mandatory
NumberOfRows	Specifies the array fetch size.	Yes
ParameterFile	Specifies the name of a file that contains iRule parameter values which can be used as placeholders in the SQL statements. They can be used in the update or startup registry.	Yes
Replace	Specifies the prefix/suffix should be replaced (**True**) or appended (**False**).	No
SqlDetail	Specifies the name of a file that contains the SQL select statement for generating an EDR detail record. The choices are: ▪ **StdRecycleDetail.sql** - used with the standard recycling feature. Used without changes. ▪ **RecycleDetail.sql** - used with the Suspense Manager service integration component. You need to customize this file for your implementation. For details, see "Configuring Suspense Manager".	Yes
SqlHeader	Specifies the name of a file that contains the SQL select statement for generating an EDR header (result must be exactly one row).	No
SqlOnFailure	Specifies the name of a file that contains the SQL statement which is executed if the output file is incorrect.	Yes
SqlOnSucess	Specifies the name of a file that contains the SQL statement that is executed if the output file is correct.	Yes
SqlTrailer	Specifies the name of a file that contains the SQL select statement for generating an EDR trailer (result must be exactly one row).	No

Sample Registry

```
Stream
{
   ControlPath = ./database/Oracle/Scripts/Suspense
   DataConnection = IntegRate.DataPool.Login
   FileName = DB
   FileNameExtension = true
  inputPrefix = sol42_
  inputSuffix = .dat
   FieldDelimiter = ;
   ParameterFile = parameter.isc
   SqlHeader = header.sql
   SqlDetail = detail.sql
   SqlTrailer = trailer.sql
   SqlOnSuccess = success.sql
   SqlOnFailure = failure.sql
   Replace = true
   SynchronizeWithOutput = true
   NumberOfRows = 1000
}
```

Event Messages

Table 40–2 lists the EXT_InEasyDB event messages.

Table 40–2 EXT_InEasyDB Event Messages

Message	Description	Send/Recv
MSG_STREAM_START	The database input stream is started.	Send: Input module Send: Format
MSG_STREAM_END	The database input stream is stopped.	Send: Input module Send: Format
MSG_STREAM_BEGIN	The database stream starts the processing of a new input file.	Send: Input module
MSG_STREAM_END	The current file has been completely processed.	Send: Input module
MSG_STREAM_STOP	The database input stream is stopped (inactive).	Send: Input module
CMD_RENAME_INPUT_STREAM	The input file is renamed.	Receive: Output module

Events

Table 40–3 lists the EXT_InEasyDB events.

Table 40–3 EXT_InEasyDB Events

Event	Trigger	Parameter
EVT_CURSOR_OPENED	Starts the processing of a restart/ job file.	File name
EVT_INPUT_DIR_EMPTY	Control directory is empty.	Name of the control directory

EXT_InFileManager

The EXT_InFileManager module performs file handling for pipeline input from files. See:

- About Getting Pipeline Input from Files
- Configuring EDR Input Processing

This module runs automatically when you start Pipeline Manager.

Configure this module as a submodule of the INP_GenericStream module. See "INP_GenericStream".

To configure input from a database, see "EXT_InEasyDB".

Registry Entries

Table 40–4 lists the EXT_InFileManager registry entries.

Table 40–4 EXT_InFileManager Registry Entries

Entry	Description	Mandatory
DonePath	Specifies the path for the processed files.	Yes
DonePrefix	Specifies the prefix for the processed files.	No
DoneSuffix	Specifies the suffix for the processed files.	No
ErrorPath	Specifies the path for incorrect files.	Yes
ErrorPrefix	Specifies the prefix for incorrect files.	No

Table 40–4 (Cont.) EXT_InFileManager Registry Entries

Entry	Description	Mandatory
ErrorSuffix	Specifies the suffix for incorrect files.	No
InputDirEmptyTimeout	Specifies the time period (in seconds), the input directory must be empty before the EVT_INPUT_DIR_EMPTY event is sent.	No
InputPath	Specifies the path for the input files.	Yes
InputPrefix	Specifies the prefix for the input files.	No
InputSuffix	Specifies the suffix for the input files.	No
Replace	Specifies the prefix and or suffix can be replaced or appended. Default = **True**.	No
TempPrefix	Specifies the prefix for temporary files.	No

Sample Registry Section

```
InputStream
  {
    ModuleName = EXT_InFileManager
    Module
    {
      InputDirEmptyTimeout = 10
      InputPath   = ./samples/wireless/data/in
      InputPrefix = test
      InputSuffix = .edr
      DonePath    = ./samples/wireless/data/done
      DonePrefix  = test
      DoneSuffix  = .done
      ErrorPath   = ./samples/wireless/data/err
      ErrorPrefix = test
      ErrorSuffix = .err
      TempPrefix  = tmp
      Replace     = TRUE
    }
  }
```

Event Messages

Table 40–5 lists the EXT_InFileManager event messages.

Table 40–5 EXT_InFileManager Event Messages

Message	Description	Send/Receive
REQ_INPUT_FILENAME	Request to send back the complete input file name (including path) corresponding to a specific stream name (as known by the TAM).	Receive
REQ_INPUT_TEMP_ FILENAME	Request to send back the temporary input file name (including path) corresponding to a specific stream name (as known by the TAM).	Receive
REQ_DONE_FILENAME	Request to send back the final input file name (including path) corresponding to a specific stream name (as known by the TAM), after successful processing.	Receive

Table 40–5 (Cont.) EXT_InFileManager Event Messages

Message	Description	Send/Receive
REQ_ERROR_FILENAME	Request to send back the final input file name (including path) corresponding to a specific stream name (as known by the TAM), after an unsuccessful processing.	Receive
REQ_RETURN_FILENAME	Request to send back the return file name (including path) corresponding to a specific stream name (as known by the TAM), when batch reject was requested.	Receive
REQ_RETURN_TEMP_FILENAME	Request to send back the temporary return file name (including path) corresponding to a specific stream name (as known by the TAM), when batch reject was requested.	Receive

EXT_OutFileManager

The EXT_OutFileManager module handles files for the OUT_Generic_Stream and OUT_Reject modules. See:

- Sending Output to a File
- Configuring EDR Output Processing

This module runs automatically when you start Pipeline Manager.

Registry Entries

Important: To ensure output file integrity, specify a unique combination of OutputPath, OutputSuffix, and OutputPrefix values for each output stream defined in the registry.

Table 40–6 lists the EXT_OutFileManager registry entries.

Table 40–6 EXT_OutFileManager Registry Entries

Entry	Description	Mandatory
AppendSequenceNumber	Specifies if the sequence number should be appended to the output file name or not. See "Applying a Prefix to the Sequence Number".	No
DeleteEmptyFile	Deletes the output file if only the header and trailer are written to the stream. **Note:** By default, this entry is set to **True**. Configure any processes that manipulate output files to wait approximately one minute before acting on a file. This delay allows the module to delete empty files.	No
Replace	Specifies if the prefix/suffix should be replaced (**TRUE**) or appended (**FALSE**). Default = **True**.	No

Table 40–6 (Cont.) EXT_OutFileManager Registry Entries

Entry	Description	Mandatory
SequencerPrefix	This entry is used to specify a prefix to the sequence number before it gets appended to the generated output file name. This entry is used only when **AppendSequencerNumber** is set to **True**. See "Applying a Prefix to the Sequence Number".	No
TempPrefix	Specifies the prefix for the output stream's temporary data file. See "Configuring the Temporary File Name". Default = **.**	No
TempDataPath	Specifies the path for the internal temporary file list. **Important:** Do not change this registry entry. It is used by the pipeline for internal data processing.	No
TempDataPrefix	Specifies the prefix for the internal temporary file list. **Important:** Do not change this registry entry. It is used by the pipeline for internal data processing.	No
TempDataSuffix	Specifies the suffix for the internal temporary file list. **Important:** Do not change this registry entry. It is used by the pipeline for internal data processing.	No
OutputPath	Specifies the path for the output files. See "Configuring File Prefixes and Suffixes".	Yes
OutputPrefix	Specifies the prefix for the output files. See "Configuring File Prefixes and Suffixes".	No
OutputSuffix	Specifies the suffix for the output files. See "Configuring File Prefixes and Suffixes".	No
UseInputStreamName	Specifies to use the input file name to build the output file name. See "Creating an Output File Name from the Input File Name".	No

Sample Registry

```
#------------------------------------------------------------------------
# The /service/telco/gsm/telephony output stream
#------------------------------------------------------------------------
TELOutput
{
  ModuleName = OUT_GenericStream
  Module
  {
    Grammar = ./formatDesc/Formats/Solution42/SOL42_V430_REL_OutGrammar.dsc
    DeleteEmptyStream = True
    OutputStream
      {
      ModuleName = EXT_OutFileManager
      Module
      {
        OutputPath = ./samples/wireless/data/telout
        OutputPrefix = test
        OutputSuffix = .out
        UseInputStreamName = [2,4;4,6;8,&]
        TempPrefix   = tmp.
```

```
        TempDataPath    = ./samples/wireless/data/telout
        TempDataPrefix  = tel.tmp.
        TempDataSuffix  = .data
        Replace = TRUE
        SequencerPrefix = "+"
        }
    }
  }
} # end of TELOutput
```

Messages and Requests

Table 40–7 lists the EXT_OutFileManager messages and requests.

Table 40–7 EXT_OutFileManager Messages and Requests

Message	Description	Send/Receive
REQ_EVENTHANDLER_NAME	Get the event handler.	Send

Events

Table 40–8 lists the EXT_OutFileManager events.

Table 40–8 EXT_OutFileManager Events

Event	Trigger	Parameter
EVT_OUTPUT_FILE_READY	The renaming from the temporary file to the output file.	Target file name

INP_GenericStream

The INP_GenericStream module provides the input interface to pipelines. See "Configuring EDR Input Processing".

Registry Entries

Table 40–9 lists the INP_GenericStream registry entries.

Table 40–9 INP_GenericStream Registry Entries

Entry	Description	Mandatory
DefaultOutput	The default output stream.	
Grammar	Path to the input grammar description file.	No
InputStream	The input submodule: ■ EXT_InEasyDB ■ EXT_InFileManager	

Sample Registry for INP_GenericStream

```
InputModule
{
  ModuleName  = INP_GenericStream
  Module
  {
```

```
    DefaultOutput = EdrOutput
    Grammar       = ./FMD/Formats/Solution42/SOL42_V430_InGrammar.dsc
    InputStream
      ModuleName = EXT_InFileManager
      Module
      {
        InputPath   = ../input/maxitel/in
        InputPrefix = Sol42
        InputSuffix = .edr
        DonePath    = ../input/maxitel/done
        DonePrefix  = Sol42
        DoneSuffix  = .done
        ErrorPath   = ../input/maxitel/err
        ErrorPrefix = Sol42
        ErrorSuffix = .err
        TempPrefix  = tmp
        Replace     = TRUE
      }
    }
  }
}
```

INP_Realtime

The INP_Realtime module converts data in flist format to the EDR container format. See "Configuring a Real-Time Discounting Pipeline".

You can use an iScript to overwrite the mappings from flist fields to EDR fields, and to add custom mappings.

For more information, see "About Customizing Mapping of Flist Fields to Rating EDR Container Fields".

Registry Entries

Table 40–10 lists the INP_Realtime registry entries.

Table 40–10 INP_Realtime Registry Entries

Entry	Description	Mandatory
DefaultOutput	Specifies the default output module. This is always the OUT_ Realtime module.	Yes
MappingScript	iScript file name.	No
OpcodeMapping	Specifies the XML file that describes the input flist field to EDR container field mapping. BRM provides the following default flist-to-EDR mappings: ■ Discounting: **discount_event.xml** ■ Rerating: **rate_event.xml** ■ Zoning: **zonemap_event.xml** You can customize these files to change how flists are mapped to EDR container fields. See "About Customizing Mapping of Flist Fields to Rating EDR Container Fields".	Yes

Sample Registry Entry

In this example, **rate_event.xml** maps rerate requests in flist format to EDR container fields.

```
InputModule
{
  ModuleName = INP_Realtime
  Module
  {
     DefaultOutput = PcmOutput
     OpcodeMapping = ./formatDesc/Formats/Realtime/rate_event.xml
     MappingScript = ./inflist.isc
  }
```

INP_Recycle

The INP_Recycle module is used by standard recycling and Suspense Manager in the pre-recycling pipeline. It reads suspended usage records from the BRM database, restores original EDRs, applies edits to them, and pushes EDRs into the pre-recycling pipeline.

- For standard recycling, see "Configuring a Pre-Recycling Pipeline".
- For Suspense Manager, see "Configuring a Pre-Recycling Pipeline".

Dependencies

Requires connections to:

- DAT_Recycling module
- BRM database

Registry Entries

Table 40–11 lists the INP_Recycle registry entries.

Table 40–11 INP_Recycle Registry Entries

Entry	Description	Mandatory
DefaultOutput	Specifies the default output stream.	Yes
InfranetConnection	Specifies a connection to the BRM database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
InputStream	Configures the EXT_InEasyDB module. See "EXT_InEasyDB".	Yes
RecyclingDataModule	Specifies a connection to the DAT_Recycling module. See "Connecting A Pipeline Manager Module To Another Module" in *BRM System Administrator's Guide*.	Yes

Sample Registry

```
InputModule
{
 ModuleName            = INP_Recycle
  RecyclingDataModule   = ifw.DataPool.RecyclingData
  Module
```

```
  {
    DefaultOutput = TELOutput
    RecyclingDataModule = ifw.DataPool.RecyclingData
    InfranetConnection = ifw.DataPool.LoginInfranet
    InputStream
    {
      ModuleName = EXT_InEasyDB
      Module
      {
        ControlPath           = ./data/input/db
        DataConnection        = ifw.DataPool.LoginInfranet
        FileName              = DB
        FileNameExtension     = true
        InputPrefix           = sol42_
        InputSuffix           = .dat
        FieldDelimiter        = \t
        RecordDelimiter       = \n
        ParameterFile         = parameter.isc
        # optional parameter:
        SqlHeader             = RecycleHeader.sql
        SqlDetail             = StdRecycleDetail.sql
        # optional parameter:
        # SqlTrailer            = trailer.sql
        SqlOnSuccess          = success.sql
        SqlOnFailure          = failure.sql
        Replace               = true
        SynchronizeWithOutput = true
        NumberOfRows          = 1000
      }
    } # end of InputStream
  }
} # end of InputModule
```

EDR Container Fields

Table 40–12 lists the INP_Recycle EDR container fields.

Table 40–12 INP_Recycle EDR Container Fields

Alias field name Default field name	Type	Access	Description
OVERRIDE_REASONS DETAIL.ASS_SUSPENSE_OVERRIDE_REASONS	String	Write	The suspense reasons to ignore during recycling. Used by the other pipeline modules for rating call records in spite of the pipeline validation rules they violate.
PIPELINE_NAME DETAIL.ASS_SUSPENSE_EXT.PIPELINE_NAME	String	Write	Name of the pipeline originally used for the EDR. Read from the database. Used by the IRL_ PipelineSplitting module.

Table 40–12 (Cont.) INP_Recycle EDR Container Fields

Alias field name Default field name	Type	Access	Description
SUSPENSE_ID DETAIL.ASS_SUSPENSE_EXT.SUSPENSE_ID	Integer	Write	The POID ID of the **/suspended_usage** object for this EDR. Used by Suspended Event Loader when updating suspended usage records.
BATCH_ID DETAIL.ORIGINAL_BATCH_ID	String	Write	The original routing switch batch ID. Used for revenue assurance.
PROCESS_STATUS DETAIL.INTERN_PROCESS_STATUS	Integer	Write	Indicates whether the EDR is being recycled (**1**) or test recycled (**2**).

INP_Restore

The INP_Restore module reads serialized EDRs from the file output by the OUT_Serialize module. It restores EDRs to normal format and pushes them into a pipeline.

Dependencies

Requires the use of the OUT_Serialize module to produce serialized EDRs in the correct format.

When you configure this module, you also configure the EXT_InFileManager module, which manages input, temporary, and done files. See "EXT_InFileManager" in the Infranet documentation for more information about this module.

Registry Entries

Table 40–13 lists the INP_Restore registry entries.

Table 40–13 INP_Restore Registry Entries

Entry	Description	Mandatory
DefaultOutput	Specifies the default output stream.	Yes
InputStream	Configures the EXT_InFileManager module.	Yes

Sample Registry

```
#---------------------------------
# Input section
#---------------------------------
Input
{
  UnitsPerTransaction             = 1

  InputModule
  {
    ModuleName                    = INP_Restore
    Module
    {
      DefaultOutput               = EdrOutput

      InputStream
```

```
        {
          ModuleName                = EXT_InFileManager
          Module
          {
            InputPath               = $DATA/in
            InputPrefix             = testpipeline
            InputSuffix             = .edr

            DonePath                = $DATA/done
            DonePrefix              = testpipeline
            DoneSuffix              = .done

            ErrorPath               = $DATA/err
            ErrorPrefix             = testpipeline
            ErrorSuffix             = .err

            TempPrefix              = tmp

            Replace                 = True

            InputDirEmptyTimeout    = 60
          }
        }
      }
    }
}
```

OUT_DB

The OUT_DB module sends output to the Pipeline Manager database.

See "Sending Output to a Database".

Dependencies

Requires a connection to the Pipeline Manager database.

Registry Entries

Table 40–14 lists the OUT_DB registry entries.

Table 40–14 OUT_DB Registry Entries

Entry	Description	Mandatory
ControlPath	Specifies the name of the control/configuration directory. See "About the OUT_DB Module Configuration Files".	Yes
DataConnection	Specifies the connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes
DeleteWithoutDetails	Specifies if an empty output stream should force a rollback of all actions. Default = **True** See "Handling Empty Output Streams".	No
Destination	Specifies the value for the destination field in the header record. See "Specifying the Destination".	No

Table 40–14 (Cont.) OUT_DB Registry Entries

Entry	Description	Mandatory
DetailTableDefinition	Specifies the name of the file that contains the description of the destination detail table. See "About the OUT_DB Module Configuration Files".	Yes
FieldDelimiter	Specifies the delimiter between each field needed by the tokenizer. See "About the OUT_DB Module Configuration Files".	Yes
HeaderTableDefinition	Specifies the name of the file that contains the description of the destination header table. See "About the OUT_DB Module Configuration Files".	No
NumberOfRows	Specifies the array size for the bulk inserter. A good value is **500**. See "About the OUT_DB Module Configuration Files".	Yes
ParameterFile	Specifies the name of parameter file which contains optional key/value entries See "Parameter File".	Yes
RowNumAlias	Specifies the alias that is replaced by the inserted rows. See "About the OUT_DB Module Configuration Files".	Yes
SaveConfigurationFile	Specifies whether to delete the configuration file of each stream. Default = **False** See "About the OUT_DB Module Configuration Files".	No
Source	Specifies the value for the source field in the header record. See "Specifying the Source".	No
SqlBeginStream	Specifies the name of SQL file that contains an SQL statement. See "SqlBeginStream".	No
SqlEndStream	Specifies the name of SQL file that contains an SQL statement. See "SqlEndStream".	No
StreamNameAlias	Specifies the alias that is replaced by the internal stream name value. See "About the OUT_DB Module Configuration Files".	No
TrailerTableDefinition	Specifies the name of the file that contains the description of the destination trailer table. See "About the OUT_DB Module Configuration Files".	No
WriteDefaultEdr	Specifies if a default EDR is written in empty data streams. Default = **False** See "Handling Empty Output Streams".	No

Sample Registry

```
Streams
{
  EdrOutput
  {
    StreamDestination       = Database
    Destination             = CBC21
    Source                  = D1
    DataConnection          = integrate.DataPool.Login
```

```
    NumberOfRows            = 500
    ControlPath             = control
    FieldDelimiter          = ;
    RowNumAlias             = __RowNum__
    StreamNameAlias         = __StreamName__
    SqlBeginStream          = beginStream.sql
    SqlEndStream            = endStreamNeu.sql
    ParameterFile           = parameter_out.isc
    HeaderTableDefinition   = headerTable.dat
    DetailTableDefinition   = detailTable.dat
    TrailerTableDefinition  = trailerTable.dat
    WriteDefaultEdr         = false
    DeleteWithoutDetails    = true
    SaveConfigurationFile   = true
  }
  Reject
  {
    StreamDestination       = File
    OutputPath              = /data/reject
    OutputSuffix            = .rej
    Replace                 = True
  }
}
```

Important: To ensure output file integrity, specify a unique combination of OutputPath, OutputSuffix, and OutputPrefix values for each output stream defined in the registry.

OUT_DevNull

The OUT_DevNull module removes EDRs that are not needed by Pipeline Manager. See "Configuring Output of Discarded EDRs".

For more information, see "Discarding and Skipping EDRs".

Sample Registry

```
OutputCollection
    {
    DevNull
    {
        ModuleName = OUT_DevNull
        Module
        {
        }
    }
```

OUT_GenericStream

The OUT_GenericStream module handles the output stream for rated EDRs. See "Configuring EDR Output Processing".

For back-out-only rerating, it generates the output file to be loaded into the BRM database by the Rated Event (RE) Loader. See "About Back-Out-Only Rerating" in *BRM Setting Up Pricing and Rating*.

When you configure the OUT_GenericStream module, you configure the EXT_OutFileManager module to specify file management options. See "EXT_OutFileManager".

Registry Entries

Table 40–15 lists the OUT_GenericStream registry entries.

Table 40–15 OUT_GenericStream Registry Entries

Entry	Description	Mandatory
AddInvoiceData	When set to True, the output module adds invoice data to each BRM billing record. Default = **False** See "Adding Pipeline Rating Data to an Invoice".	No
DeleteEmptyStream	Specifies to delete empty output streams. If you run multiple RE Loader processes in parallel, set this option to **True**. Otherwise, RE Loader attempts to load the empty files. Default = **True** See "Configuring Pipeline Manager to Delete Empty Output Streams".	No
EventType	Specifies the BRM event type that the output file contains, such as **/event/delayed/session/gsm**.	Mandatory only for RE Loader-related pipelines
Grammar	Specifies the output grammar description file.	Yes
OutputStream	Contains the configuration for the EXT_OutFileManager.	Yes
ProcessType	Specifies which process created the output file and can be set to one of the following values: ■ **RATING_PIPELINE** ■ **EVENT_EXTRACTION_TOOL** ■ **BACKOUT_PIPELINE** ■ **RERATING_PIPELINE** ■ **PIN_TRANSFORM_CDR**	Mandatory only for RE Loader-related pipelines
Sequencer	Specifies the Sequencer for performing sequence generation. This Sequencer must be defined in the **SequencerPool** section of the registry file. See: ■ "Configuring Sequence Checking" in *BRM System Administrator's Guide* ■ Sequencer	No

Sample Registry

```
#------------------------------------------------------------------
# The /service/telco/gsm/telephony output stream
#------------------------------------------------------------------
TELOutput
{
  ModuleName = OUT_GenericStream
  ProcessType = RATING_PIPELINE
  EventType = /event/delayed/session/gsm
```

```
  Module
  {
    Grammar = ./formatDesc/Formats/Solution42/SOL42_V430_REL_OutGrammar.dsc
    DeleteEmptyStream = True
    Sequencer = SequenceGenerator_1
    OutputStream
      {
      ModuleName = EXT_OutFileManager
      Module
      {
        OutputPath = ./samples/wireless/data/telout
        OutputPrefix = test
        OutputSuffix = .out
        TempPrefix   = tmp.
        TempDataPath    = ./samples/wireless/data/telout
        TempDataPrefix  = tel.tmp.
        TempDataSuffix  = .data
        Replace = TRUE
        }
     }
  }
} # end of TELOutput
```

Important: To ensure output file integrity, specify a unique combination of OutputPath, OutputSuffix, and OutputPrefix values for each output stream defined in the registry.

OUT_Realtime

The OUT_Realtime module converts data in the pipeline EDR output to flist format. It sends the output to the NET_EM module automatically. You don't need to configure a pointer to the NET_EM module.

You can use an iScript to overwrite the mappings and to add custom mappings. You use the registry file to specify the iScript.

For more information, see "Configuring a Real-Time Discounting Pipeline".

Registry Entries

Table 40–16 lists the OUT_Realtime registry entries.

Table 40–16 OUT_Realtime Registry Entries

Entry	Description	Mandatory
MappingScript	iScript file name.	No
DoRating	Specifies the pipeline to be used. Set to **False** for Discounting/Zoning pipeline. Set to **True** for Rerating pipeline.	Yes

Sample Registry Entry

```
OutputCollection
{
  PcmOutput
  {
```

```
      ModuleName = OUT_Realtime
      Module
      {
         DoRating = false
         #MappingScript = <Optional Output IScript>
      }
    }
  }
```

OUT_Reject

The OUT_Reject module writes rejected EDRs to an output stream. The written record is exactly the same as the original input record. See "Configuring Output for Rejected or Duplicate EDRs".

When you configure the OUT_Reject module, you configure the EXT_OutFileManager module to specify file management options. See "EXT_OutFileManager".

Sample Registry

```
Reject
{
  ModuleName = OUT_Reject
  Module
  {
    OutputStream
    {
      ModuleName = EXT_OutFileManager
      Module
      {
        OutputPath = ../output/tel/rej
        OutputPrefix = Sol42
        OutputSuffix = rej
        TempPrefix   = tmp
        Replace = TRUE
        DeleteEmptyFile = FALSE
      }
    }
  }
}
```

Important: To ensure output file integrity, specify a unique combination of OutputPath, OutputSuffix, and OutputPrefix values for each output stream defined in the registry.

OUT_Serialize

The OUT_Serialize module creates serialized EDR records with base64 encoding.

Dependencies

The INP_Restore module is required to read and restore EDRs serialized by this module.

When you configure this module, you also configure the EXT_OutFileManager module, which manages output files. See "EXT_OutFileManager" in the Infranet documentation for more information about this module.

Registry Entries

Table 40–17 lists the OUT_Serialize registry entries.

Table 40–17 OUT_Serialize Registry Entries

Entry	Description	Mandatory
DeleteEmptyStream	Specifies whether empty streams should be deleted. Set to **True** to prevent SEL from attempting to load empty files. Default = **True**	Yes
ProcessType	Specifies which process created the output file. If used, sets the HEADER.CREATION_PROCESS field in the EDR to the value specified.	No
EventType	Specifies the Infranet event type that the output file contains, such as **/event/delayed/session/gprs**. If used, sets the HEADER.EVENT_TYPE field to the values specified.	No
OutputStream	Configures the EXT_OutFileManager.	Yes

Sample Registry

```
#------------------------------------------------------------
# The serialized EDR output stream
#------------------------------------------------------------
SerEdrOutput
{
  ModuleName = OUT_Serialize
  Module
  {
    DeleteEmptyStream = True
    ProcessType = RECYCLING_PIPELINE
    EventType = /event/delayed/session/gprs

    OutputStream
    {
      ModuleName = EXT_OutFileManager
      Module
      {
        OutputPath = $DATA/out
        OutputPrefix = test
        OutputSuffix = .out

        TempPrefix   = .
        TempDataPath    = $DATA/out
        TempDataPrefix  = tel.tmp.
        TempDataSuffix  = .data

        Replace = TRUE
      }
    }
  }
} # end of SerEdrOutput
```

Pipeline Dispatcher

Parses CDR files to multiple identical pipelines. This module routes files with a specified filename prefix and suffix from a single input directory to multiple pipelines in round-robin fashion.

For information, see "Connecting a Module to a Database" in *BRM System Administrator's Guide*.

Registry Entries

Table 40–18 lists the Pipeline Dispatcher registry entries.

Table 40–18 Pipeline Dispatcher Registry Entries

Entry	Value	Description	Mandatory
DispatcherName	N/A	Specifies the name of the dispatcher. You can use any name, for example, **Dispatcher1**. If your system requires multiple dispatchers, create a set of entries for each dispatcher.	N/A
DispatcherName.**InputPath**	String	Specifies the path of the CDR input directory. For example: **InputPath = ./samples/wireless/data/input**	N/A
DispatcherName.**InputPrefix**	String	Specifies the prefix of all CDR files you want routed. The dispatcher only routes files with the specified prefix and suffix to your pipelines. For example: **InputPrefix = test**	N/A
DispatcherName.**InputSuffix**	String	Specifies the suffix of all CDR files you want routed. The dispatcher only routes files with the specified prefix and suffix to your pipelines. For example: **InputSuffix = .edr**	N/A
DispatcherName.**TargetPipelines**	N/A	Subgroup that lists which pipelines to route your CDR files.	N/A
DispatcherName.**TargetPipelines.DestinationPipeline**	String	Specifies to which pipelines to route CDR files. Add an entry for each pipeline. For example: **DestinationPipeline = ifw.Pipelines.W_SAMPLE** **DestinationPipeline = ifw.Pipelines.W_SAMPLE_2** **Important:** The pipelines must use a separate input directory from the CDR input files.	N/A
DispatcherName.**TargetPipelines.Routing**	ROUND_ROBIN	Specifies to use round-robin routing. **Note:** The dispatcher uses round-robin routing only.	Yes

Sample Registry

```
ifw
{
    PipelineDispatcher
    {
        ModuleName = EXT_PipelineDispatcher
        Module
        {
            Dispatcher1
```

```
                    {
                        InputPath = ./samples/wireless/input
                        InputPrefix = TAP3
                        InputSuffix = .edr

                        TargetPipelines
                        {
                            DestinationPipeline = ifw.Pipelines.Pipeline_1
                            DestinationPipeline = ifw.Pipelines.Pipeline_2
                            Routing = ROUND_ROBIN
                        }
                    }
                }
            }
        }
```

41

Pipeline Manager Framework Modules

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager framework modules.

Controller

Use the Pipeline Manager Controller to control and monitor components in the Pipeline Manager framework. The Controller also generates the log message table that is used by the LOG module to generate the Process log file, the Pipeline Manager log file, and the Stream log file. For information, see "LOG".

For information, see "About the Controller" in *BRM Concepts*.

Registry Entries

Table 41–1 lists the Controller registry entries.

Table 41–1 Controller Registry Entries

Entry	Value	Description	Mandatory?
Active	**True** **False**	Activates or deactivates the Pipeline Manager framework. ▪ **True** activates the Pipeline Manager framework. ▪ **False** deactivates the Pipeline Manager framework. See "Starting And Stopping Pipeline Manager Manually" in *BRM System Administrator's Guide*.	Yes
DataPool	N/A	Subgroup that contains entries for all data modules in the Pipeline Manager framework. See "About the data pool" in *BRM Concepts*.	Yes
DiagnosticDataHandler	N/A	Subgroup that configures diagnostic data collection. See "Diagnostic Data Handler" and "Using The Diagnostic Data Handler To Get OMF Diagnostic Data" in *BRM System Administrator's Guide*.	No
EventHandler	N/A	Subgroup that contains the Event Handler entries. See "Using Events to Start External Programs" in *BRM System Administrator's Guide* and "Event Handler".	No

Table 41–1 (Cont.) Controller Registry Entries

Entry	Value	Description	Mandatory?
Instrumentation	N/A	Subgroup that configures Operations Management Framework (OMF) instrumentation data collection. See "About Operations Management Framework" and "Enabling SNMP Instrumentation Data Collection" in *BRM System Administrator's Guide*.	Yes
Instrumentation.ProbeBroker	String	Specifies the path to the OMF insturmentation folder. See "About Operations Management Framework" in *BRM System Administrator's Guide*.	Yes
Instrumentation.SnmpServer	String	Specifies configuration data for the OMF SNMP server. See "About Operations Management Framework" and "Enabling SNMP Instrumentation Data Collection" in *BRM System Administrator's Guide*.	No
Instrumentation.HttpServer	String	Specifies configuration data for the OMF HTTP server. See "About Operations Management Framework" and "Enabling HTTP Display Of Instrumentation Data" in *BRM System Administrator's Guide*.	No
LogMessageTable.MessageFilePath	String	Specifies the path to your error message files. By default, the Pipeline Manager installer installs your error message files in the *Pipeline_Home*/**etc** directory. *Pipeline_Home* is the directory where you installed Pipeline Manager. See "About error message files" in *BRM System Administrator's Guide*.	No
LogMessageTable.MessageFilePrefix	String	Specifies the prefix for your error message files. See "About error message files" in *BRM System Administrator's Guide*.	No
LogMessageTable.MessageFileSuffix	String	Specifies the suffix for your error message files. See "About error message files" in *BRM System Administrator's Guide*.	No
MemoryMonitor	N/A	Subgroup that configures memory monitoring. See "Memory Monitor".	No
Pipelines	N/A	Subgroup that contains the entries for each pipeline. See "About Pipelines" in *BRM Concepts*.	Yes
ProcessLog	N/A	Subgroup that contains entries for the main processing log. See "About Pipeline Manager Log Files" in *BRM System Administrator's Guide* and "LOG".	Yes

Table 41–1 (Cont.) Controller Registry Entries

Entry	Value	Description	Mandatory?
ProcessLoopTimeout	Integer	Specifies the interval, in seconds, between polling for a new semaphore file. This parameter controls the overall event loop, which includes looking for semaphore files. The value must be greater than **0**. See "Using Semaphore Files To Control Pipeline Manager" in *BRM System Administrator's Guide*.	Yes
QueueRequestTimeout	Integer	Specifies the interval for logging buffer sizes, in seconds. A value of **0** turns off logging. See "Configuring Buffer Size Polling" in *BRM System Administrator's Guide*.	Yes
Registry	N/A	Subgroup that contains registry processing entries. See "Using Registry Files To Configure Pipeline Manager" in *BRM System Administrator's Guide*.	Yes
Registry.Buffer	N/A	Subgroup that specifies the registry buffer's size and type. The registry entries in this subgroup depend on the buffer type. See "Configuring Pipeline Buffers" in *BRM System Administrator's Guide*.	Yes
Registry.FileName	String	Specifies the name of a file in dot-separated format that contains current updated registry settings, including semaphore updates. See "Using Registry Files To Configure Pipeline Manager" in *BRM System Administrator's Guide*.	Yes
Registry.FilePath	String	Specifies the path to the registry file. See "Using Registry Files To Configure Pipeline Manager" in *BRM System Administrator's Guide*.	Yes
Registry.NiceFormatFileName	String	Specifies the name of a file that contains current updated registry settings, including semaphore updates. This file format is easier to read than **.FileName** format and can be used to debug a registry file or create a new one. See "Using Registry Files To Configure Pipeline Manager" in *BRM System Administrator's Guide*.	Yes
Semaphore	N/A	Subgroup that contains semaphore processing entries. See "Using Semaphore Files To Control Pipeline Manager" in *BRM System Administrator's Guide*.	Yes
Semaphore.FileName	String	Specifies the name of the semaphore file. See "Using Semaphore Files To Control Pipeline Manager" in *BRM System Administrator's Guide*.	Yes

Table 41–1 (Cont.) Controller Registry Entries

Entry	Value	Description	Mandatory?
Semaphore.FilePath	String	Specifies the path to the semaphore file. See "Using Semaphore Files To Control Pipeline Manager" in *BRM System Administrator's Guide*.	Yes
Semaphore.RetainFiles	**True** **False**	Specifies whether semaphore files are deleted or saved after they are processed. ■ **True** specifies to save semaphore files. The Controller renames the file by appending the current timestamp to the file name in the format *YYYYMMDD_hhmmss* and logs the semaphore file's new name in the **process.log** file. For example, the **semaphore.reg** file is renamed **semaphore_20031022_120803.reg**. ■ **False** specifies to delete semaphore files immediately after they are processed. The default is **False**. See "Using Semaphore Files To Control Pipeline Manager" in *BRM System Administrator's Guide*.	No
TransactionIDController	N/A	Subgroup that contains the Transaction ID Controller entries. See "About Pipeline Manager Transactions" in *BRM System Administrator's Guide* and "Transaction ID Controller".	Yes

Sample Registry File

```
ifw
{
    Active = TRUE
    ProcessLoopTimeout = 10
    QueueRequestTimeout = 0
    Semaphore
    {
        FilePath = ./semaphore
        FileName = semaphore.reg
        RetainFiles = False
    }
    Registry
    {
        FilePath = ./info
        FileName = Sample.reg
        NiceFormatFileName = niceSample.reg
        Buffer
        {
            Size = 1000
        }
    }
    ProcessLog
    {
        ModuleName = LOG
        Module
        {
        }
    }
    LogMessageTable
```

```
    {
        MessageFilePath= ./etc
        MessageFileSuffix= .msg
    }
    EventHandler
    {
        ModuleName = EVT
        Module
        {
        }
    }
    DataPool
    {
        Login
        {
            ModuleName = DBC
            Module
            {
                UserName = TEST
                PassWord = 574B9CD1CBFD1B077760181C111B181D10661B67
                DatabaseName = TE01
                AccessLib = oci11g72
                Connections = 5
            }
        }
    }
    TransactionIdController
    {
        Source = Database
        Generator
        {
            DataConnection = integrate.DataPool.Login
        }
    }
    Pipelines
    {
    }
}
```

Semaphore File Entries

Table 41–2 lists the Controller Semaphore file entries.

Table 41–2 Controller Semaphore File Entries

Entry	Description
Active	Starts and stops the Pipeline Manager framework.
LogTimeStatistic	Specifies whether to write time statistics into the process log file.
QueueRequestTimeout	Specifies the interval for logging buffer sizes, in seconds. A value of **0** turns off logging.

Sample Semaphore Entry

```
ifw.Active = True
```

Events

Table 41–3 lists the Controller Events.

Table 41–3 Controller Events

Event	Trigger
EVT_INTEGRATE_START	Pipeline Manager starts processing.
EVT_INTEGRATE_STOP	Pipeline Manager terminates. No more files are processed.

Database Connect (DBC)

Connects the Pipeline Manager framework to the Pipeline Manager database.

See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.

Registry Entries

Table 41–4 lists the Database Connect registry entries.

Table 41–4 Database Connect Registry Entries

Entry	Value	Description	Mandatory?
AccessLib	**oci10g72** **oci11g72**	Specifies the name of the database access library, without the **lib** prefix and **.so** suffix. ■ Use **oci10g72** for Oracle10g databases. ■ Use **oci11g72** for Oracle11g databases.	Yes
Connections	Integer	Specifies the number of database connections that a database module holds open in a connection pool. The default is **1**.	No
DatabaseName	String	Specifies the database name.	Yes
PassWord	String	Specifies the encrypted database password in hexadecimal format.	Yes
ServerName	String	Specifies the server name. The default is **''**.	No
UserName	String	Specifies the database user name.	Yes
ConvertToUpperCase	True False	Specifies whether to convert the user name used by the password decryption to uppercase. The default is **True**.	No

Sample Registry for Oracle Databases

```
ifw
{
    DataPool
    {
    ...
        #------------------------------------------------------------
        # Database Connection Pipeline
        #------------------------------------------------------------
        Login
        {
            ModuleName = DBC
            Module
            {
                UserName = USER
                PassWord = 574B93D1CBF21D1161611E0A07
                DatabaseName = ORA_DB
                AccessLib = oci11g72
                Connections = 5
```

```
            }
        }
    ...
    }
}
```

Semaphore Entries

Table 41–5 lists the Database Connect Semaphore entry.

Table 41–5 Database Connect Semaphore Entry

Parameter	Description	Mandatory
Reconnect	Closes all open database connections and reconnects to the database. See "Forcing a Database Reconnection" in *BRM System Administrator's Guide*.	No

Sample Semaphore

```
ifw.DataPool.Login.Module.Reconnect {}
```

EDR Factory

Use the EDR Factory to generate and allocate memory to EDR containers. The EDR Factory uses a container description file to generate each container. For information, see "About the EDR Factory" in *BRM Concepts*.

Registry Entries

Table 41–6 lists the EDR Factory registry entries.

Table 41–6 EDR Factory Registry Entries

Entry	Value	Description	Mandatory?
DataConnection	String	Specifies the registry name of the database connection module (DBC). When you use this entry, the EDR Factory connects to the IFW_ALIAS_MAP database table to retrieve alias names. **Note:** Use this field when you use aliases to describe EDR container fields. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	No
Description	String	Specifies the path to the container description file. See "About the Container Description File".	Yes
EdrVersionDataConnection	String	Specifies the registry name of the database connection module (DBC). When you use this entry, the EDR Factory connects to the EDR_FIELD_MAPPING_T database table to retrieve the EDR field mapping.	No
UsageStatistic	String	Specifies the name of the usage statistic file. This file lists all modules that are using EDR container fields together with the fields that are accessed by each module.	No

Sample Registry

```
EdrFactory
{
    DataConnection = integrate.DataPool.Login
    Description = ./FMD/Portal/containerDesc.dsc
    UsageStatistic = edrStatistic.txt
    EdrVersionDataConnection = ifw.DataPool.LoginInfranet
}
```

Event Handler

Use the Event Handler to start external programs when triggered by internal Pipeline Manager events.

For information, see "Using Events to Start External Programs" in *BRM System Administrator's Guide*.

Registry Entries

Table 41–7 lists the Event Handler registry entries.

Table 41–7 Event Handler Registry Entries

Entry	Description	Mandatory?
Buffer	Subgroup that specifies the size and type of the Event Handler's internal queue buffer. The registry entries in this subgroup depend on the buffer type. See "Configuring Pipeline Buffers" in *BRM System Administrator's Guide*.	Yes
Events	Subgroup that contains trigger entries. See "About Mapping Events To Programs" in *BRM System Administrator's Guide*.	Yes
Event.*EventName*	Specifies the event that triggers an external program. Add an entry for each event that triggers an external program. See "About Mapping Events To Programs" in *BRM System Administrator's Guide*.	Yes
Event.*ModuleSendingEvent*	Specifies the registry name of the module that sends the event to the Event Handler. Add an entry for each module that can trigger an external program. See "About Mapping Events To Programs" in *BRM System Administrator's Guide*.	Yes
TimeToWait	Specifies the time in seconds that the Event Handler waits for the external program to terminate. By default, no **TimeToWait** value is assumed. See "Controlling External Programs" in *BRM System Administrator's Guide*.	No

Sample Registry

```
EventHandler
{
    ModuleName = EVT
    Module
    {
        Events
        {
            ifw.DataPool.Customer.Module
            {
```

```
                EVT_ReloadSuccess = ./script/script_1
                EVT_ReloadFailed = ./script/script_2
                TimeToWait = 30
            }
            ifw.Pipelines.*
            {
                EVT_PIPELINE_START = ./script/script_3
                TimeToWait = 10
            }
        }
        Buffer
        {
            Size = 10
        }
    }
}
```

Instances

Use the Instances module to configure multiple instances of sequencers, output streams, or system brands for multiple roaming partners. This module is optional.

> **Note:** This module does not configure multiple instances of pipelines. To do that, use the **ifw.Pipelines.Instances** subsection.

For more information, see "About Configuring Multiple Instances of Sequencers, Output Streams, or System Brands" in *BRM System Administrator's Guide*.

Registry Entries

Table 41–8 lists the Instances registry entries.

Table 41–8 Instances Registry Entries

Entry	Description	Mandatory?
InstantiationName	Specifies the descriptive name of the instantiation, for example, TAPOutputStreamsInstantiation.	Yes, if the Instances module is used.
InstantiationName.**BlockName**	Specifies the template section or entry in the roaming registry file that is used to instantiate multiple registry sections or entries. The template section or entry contains variables for the section name, entry name, or the value of the entry that must be changed in each new instance created.	Yes
InstantiationName.**DataFile**	Specifies the path to the data file generated by the **RoamingConfigGen64** utility. See "RoamingConfigGen64" for more information.	Yes
InstantiationName.**InstanceSpecificEntries**	Subgroup that specifies the entries that must be changed in each new instance created, such as the section name, entry name, the value of an entry, the change mode, and so on.	Yes
InstantiationName.**InstanceSpecificEntries**.*InstanceChangeName*	Specifies the descriptive name of the change required in each instance; for example, ModifyBlockName.	Yes

Table 41–8 (Cont.) Instances Registry Entries

Entry	Description	Mandatory?
InstantiationName.**Instance SpecificEntries**.*InstanceChangeName*.**Instance**	Specifies whether to change the section name, entry name, or the value of the entry in each new instance created. Valid values are: ■ **[BlockName]** specifies that the section name or entry name must be changed in each new instance. For example, to change the section name of the **SequencerPool.SEQ_GEN_TAPOUT_XXX** subsection in each new instance (such as **SEQ_GEN_TAPOUT_OPR01**, **SEQ_GEN_TAPOUT_OPR02**, and so on), set the **Instance** entry to **[BlockName]**. ■ **[BlockValue]** specifies that the value of the entry must be changed in each new instance. **Note:** Use this value only if *InstantiationName*.**BlockName** is a template entry: do not use this value if it is a template section. For example, to change the value of the **SystemBrands.XXX** entry in each new instance (such as TAPOutput_OPR01, TAPOutput_OPR02, and so on), set the **Instance** entry to **[BlockValue]**. ■ *RegistryEntry* specifies the entry in the template registry section for which the value must be changed in each new instance. **Note:** Use this value only if *InstantiationName*.**BlockName** is a template section: do not use this value if it is a template entry. For example, to change the value of the **Module.Recipient** entry in the **TAPOutput_XXX** section, set the **Instance** entry to **Module.Recipient**. For more information, see "About Configuring Multiple Instances of Sequencers, Output Streams, or System Brands" in *BRM System Administrator's Guide*.	Yes
InstantiationName.**Instance SpecificEntries**.*InstanceChangeName*.**UseColumn**	Specifies the column in the data file generated by the **RoamingConfigGen64** utility. This column is used to change the section name, entry name, or the value of the entry in each instance according to the change mode. For example, the TAPOUT_SEQUENCER column is used to change the section name in each instance of the **SequencerPool.SEQ_GEN_TAPOUT_XXX** subsection. See "Sample Registry for Multiple Instances of Sequencers".	Yes

Table 41–8 (Cont.) Instances Registry Entries

Entry	Description	Mandatory?
InstantiationName.**InstanceSpecificEntries**.*InstanceChangeName*.**Mode**	Specifies the mode of changing the section name, entry name, or the value of the entry in each instance using the column values in the data file generated by the **RoamingConfigGen64** utility. Valid values are: ■ **REPLACE** specifies that the section name, entry name, or the value of the entry is replaced with the corresponding column value from the data file. For example, if the entry name is *XXX* and the corresponding column value is **OPR01**, *XXX* is replaced with **OPR01** in the newly created instance. ■ **PREFIX** specifies that the corresponding column value is prefixed with the section name, entry name, or the value of the entry in each instance. For example, if the value of the entry is **.tmp** and the corresponding column value is **OPR01**, **OPR01** is prefixed with **.tmp** and the value is added as **OPR01.tmp** in the newly created instance. ■ **SUFFIX** specifies that the corresponding column value is suffixed with the section name, entry name, or the value of the entry in each instance. For example, if the value of the entry is **NREUR01** and the corresponding column value is **OPR01**, **OPR01** is suffixed with **NREUR01** and the value is added as **NREUR01OPR01** in the newly created instance. The default mode is **REPLACE**.	No

Sample Registry for Multiple Instances of Sequencers

```
Instances
{
   SEQ_GEN_TAPOUT
    {
     BlockName  = SequencerPool.SEQ_GEN_TAPOUT_XXX
     DataFile = ./RoamingPartnerConf.dat
     InstanceSpecificEntries
     {
         ModifyBlockName
         {
           Instance = [BlockName]
           UseColumn = TAPOUT_SEQUENCER
          }
      }
    }
}
SequencerPool
{
SEQ_GEN_TAPOUT_XXX
{
  Source = Database
  Controller
  {
   SequencerType = Generation
   ReuseGap = True
   SequenceLength = 5
   DatabaseConnection = ifw.DataPool.Login
  }
}
```

Sample Registry for Multiple Instances of System Brands

```
Instances
{
 EventSplitting
  {
   BlockName  =
Pipelines.TAPOutCollectPipeline.Functions.Processing.FunctionPool.RoamPartner_
EventSplitting.Module.SystemBrands.XXX
   DataFile = ./RoamingPartnerConf.dat
   InstanceSpecificEntries
   {
     ModifyBlockName
     {
       Instance = [BlockName]
       UseColumn = VPLMN
     }
     ModifyBlockValue
     {
      Instance = [BlockValue]
      UseColumn = TAPOUT_STREAM
     }
   }
}
RoamPartner_EventSplitting
{
   ModuleName = FCT_EnhancedSplitting
   Module
   {
     Active          = True
     DataConnection  = ifw.DataPool.Login
     DefaultOutput   = SuspenseCreateOutput
     SystemBrands
     {
     XXX         = TAPOutput_XXX
      SUSP        = SuspenseCreateOutput
     }
   }
}
```

Sample Registry for Multiple Instances of Output Streams

```
Instances
{
TAPOutputStreaminstantiation
    {
    BlockName = Pipelines.TAPOutCollectPipeline.Output.OutputCollection.TAPOutput_
XXX
    DataFile = ./RoamingPartnerConf.dat
    InstanceSpecificEntries
    {
      ModifyBlockName
      {
        Instance = [BlockName]
        UseColumn = TAPOUT_STREAM
       }
      ModifyOutputStreamSequencer
      {
          Instance = Module.Sequencer
          UseColumn = TAPOUT_SEQUENCER
```

```
      }
       ModifyRecipient
       {
           Instance = Module.Recipient
           UseColumn = VPLMN
        }
       ModifyCountryCode
       {
           Instance = Module.CountryCode
           UseColumn = COUNTRYCODE
        }
       ModifyDecimalPlaces
       {
           Instance = Module.DecimalPlaces
           UseColumn = DECIMALPLACES
        }
        ModifyOutputPath
        {
           Instance = Module.OutputStream.Module.OutputPath
           UseColumn = TAPOUT_PATH
        }
        ModifyOutputPrefix
        {
           Instance = Module.OutputStream.Module.OutputPrefix
           UseColumn = TAPOUT_PREFIX
        }
        ModifyTempPrefix
        {
           Instance = Module.OutputStream.Module.TempPrefix
           UseColumn = TMP_PREFIX
        }
        ModifyTempDataPath
        {
            Instance = Module.OutputStream.Module.TempDataPath
            UseColumn = TAPOUT_PATH
         }
         ModifyTempDataPrefix
         {
             Instance = Module.OutputStream.Module.TempDataPrefix
             UseColumn = TMP_DATA_PREFIX
          }
    }
}
TAPOutput_XXX
{
ModuleName  = OUT_GenericStream
ProcessType = TAPOUTCOLLECT_PIPELINE
EventType   = /event/delayed/session/telco/gsm
Module
{
   Grammar = ./formatDesc/Formats/TAP3-NG/TAP_0312_OutGrammar.dsc
   DeleteEmptyStream = False
   Sequencer = SEQ_GEN_TAPOUT_XXX
   Sender = PORTL
   Recipient = XXX
   CountryCode = XXX
   DecimalPlaces = XXX
   OutputStream
   {
     ModuleName = EXT_OutFileManager
```

```
     Module
     { OutputPath            = ./data/outcollect/tapout/XXX
       OutputPrefix          = CDPORTLXXX
       TempPrefix            = tmptest_XXX_
       TempDataPath          = ./data/outcollect/tapout/XXX
       TempDataPrefix        = test.XXX.tmp.
       TempDataSuffix        = .data
       UseInputStreamName    = [0,0]
       SequencerPrefix       = ""
       AppendSequenceNumber  = True
     }
   }
}

NRTRDEOutput_XXX
          {
            ModuleName  = XXX
            ProcessType = TAPOUTCOLLECT_PIPELINE
            EventType   = /event/delayed/session/telco/gsm

            Module
            {
              Grammar = ./formatDesc/Formats/TAP3/NRTRDE2_v01_OutGrammar.dsc
              DeleteEmptyStream = False
              Sequencer = SEQ_GEN_NRTRDEOUT_XXX

              Sender = EUR01
              Recipient = XXX

              OutputStream
              {
                ModuleName = EXT_OutFileManager
                Module
                {
                  OutputPath            = ./data/outcollect/nrtrdeout/XXX
                  OutputPrefix          = NREUR01XXX
                  TempPrefix            = tmptest_XXX_
                  TempDataPath          = ./data/outcollect/nrtrdeout/XXX
                  TempDataPrefix        = test.XXX.tmp.
                  TempDataSuffix        = .data
                  UseInputStreamName    = [0,0]
                  SequencerPrefix       = ""
                  AppendSequenceNumber  = True
                }
              }
            }
          }

NRTRDEOutputStreams
   {
   BlockName=Pipelines.TAPOutCollectPipeline.Output.OutputCollection.NRTRDEOutput_
XXX
      DataFile = ./conf/RoamingPartnerConf.dat
      InstanceSpecificEntries
      {
        ModifyBlockName
        {
          Instance = [BlockName]
          UseColumn = NRTRDEOUT_STREAM
        }
```

```
            ModifyModuleName
            {
              Instance = ModuleName
              UseColumn = NRTDEOUTPUTSTREAMMODULE
            }
            ModifyOutputStreamSequencer
            {
              Instance = Module.Sequencer
              UseColumn = NRTRDEOUT_SEQUENCER
            }
            ModifyRecipient
            {
              Instance = Module.Recipient
              UseColumn = VPLMN
            }
            ModifyOutputPath
            {
             Instance = Module.OutputStream.Module.OutputPath
             UseColumn = NRTRDEOUT_PATH
            }
            ModifyOutputPrefix
            {
              Instance = Module.OutputStream.Module.OutputPrefix
              UseColumn = NRTRDEOUT_PREFIX
            }
            ModifyTempPrefix
            {
              Instance = Module.OutputStream.Module.TempPrefix
              UseColumn = TMP_PREFIX
            }
            ModifyTempDataPath
            {
              Instance = Module.OutputStream.Module.TempDataPath
              UseColumn = NRTRDEOUT_PATH
            }
            ModifyTempDataPrefix
           {
              Instance = Module.OutputStream.Module.TempDataPrefix
              UseColumn = TMP_DATA_PREFIX
            }
        }
}
```

LOG

Use the LOG module to manage and create your system log files:

See "About Pipeline Manager Log Files" in *BRM System Administrator's Guide*.

Dependencies

The LOG module needs access to the log message table generated by the Controller in order to create the system log files. For information, see "Controller".

Registry Entries

Table 41–9 lists the LOG registry entries.

Table 41–9 LOG Registry Entries

Entry	Description	Mandatory?
FileName	Specifies the name of your system log file. If empty, the name will be built by the date.	No
FilePath	Specifies the path to your system log file.	No
FilePrefix	Specifies the log file prefix.	No
FileSuffix	Specifies the log file suffix.	No
LogLevel	Specifies the minimum severity limit. All messages greater or equal to the limit are logged. For example, enter **major** to log only major and critical error messages. Values are: ■ **critical** ■ **major** ■ **minor** ■ **warning** ■ **normal** ■ **debug** The default is **normal**, which means that all messages are logged. Using the **debug** setting returns additional debugging data.	No
MessageGroup	Specifies the message group name.	No
ProcessName	Specifies the process name.	No
ShowOriginator	Specifies whether to write the name of the module that emitted the message to the log file. **True** specifies to log the module name. This helps Oracle Technical support troubleshoot any problems. **False** specifies to not log the module name. The default is **False**.	No
SuppressErrors	Specifies any error messages to exclude from log files. For example, enter ERR_INSERTING_CLI to prevent those error messages from being logged.	No
WriteMessageKey	Specifies whether the module logs error codes. For example: ERR_FILE_NOT_FOUND. **True** specifies to write both the error code and error message to the log file. This helps technical support troubleshoot any problems. **False** specifies to write only the error message to the log file. The default is **False**.	No

Sample Registry Entry for the Process Log

```
ProcessLog
{
    ModuleName = LOG
    Module
    {
        ITO
        {
            FilePath = /ifw/log/process
            FileName = process
```

```
            FilePrefix = log_
            FileSuffix = .log
            LogLevel = normal
            ProcessName = ifw
            SuppressErrors
            {
                INF_IGNORE_CLI
                ERR_INSERTING_CLI
            }
        }
    }
}
```

Sample Registry Entry for the Pipeline Log

```
PipelineLog
{
    ModuleName = LOG
    Module
    {
        ITO
        {
            FilePath = /ifw/log/pipeline
            FileName = pipe2
            FileSuffix = .log
            LogLevel = minor
            SuppressErrors
            {
                INF_IGNORE_CLI
                ERR_INSERTING_CLI
            }
        }
    }
}
```

Sample Registry Entry for the Stream Log

```
OutputLog
{
    ModuleName = LOG
    Module
    {
        ITO
        {
            FilePath = /ifw/log/stream
            FilePrefix = stream_
            FileSuffix = .log
            LogLevel = normal
            SuppressErrors
            {
                ERR_SPEC_VERSION_INVALID
                ERR_RELEASE_VERSION_INVALID
            }
        }
    }
}
```

Semaphores

Table 41–10 lists the LOG Semaphores.

Table 41–10 LOG Semaphores

Entry	Description
FileName	Specifies the name of the log file. When you change the file name, the current log file is closed and renamed to file name plus timestamp. For example, the **process.log** file would be renamed **process_20030916130000.log**.
LogLevel	Specifies the minimum severity limit. The module logs all messages greater or equal to the limit. For example, enter **minor** to log only minor, major, and critical error messages. Values are: ■ **critical** ■ **major** ■ **minor** ■ **warning** ■ **normal** ■ **debug**
ShowOriginator	Specifies whether to write the name of the module that emitted the message to the log file. **True** specifies to log the module name. This helps Oracle Technical Support troubleshoot any problems. **False** specifies to not log the module name.
SuppressErrors	Specifies any error messages to exclude from the log file. For example, enter ERR_GETTING_DATADESCR to prevent those error messages from being logged.
WriteMessageKey	Specifies whether the module logs error codes. For example: ERR_FILE_NOT_FOUND. **True** specifies to write both the error code and error message to the log file. This helps Oracle Technical Support troubleshoot any problems. **False** specifies to write only the error message to the log file.

Sample Semaphores

```
ifw.ProcessLog.Module.ITO.FileName = process
ifw.ProcessLog.Module.ITO.LogLevel = minor
ifw.ProcessLog.Module.ITO.SupressErrors = ERR_GETTING_DATADESCR
```

Input Controller

Use the Input Controller to manage the input streams for its associated pipeline.

The Input Controller performs the following functions:

- Combines multiple CDR files into one transaction when configured to do so. See "Combining Multiple CDR Files Into One Transaction" in *BRM System Administrator's Guide*.
- Notifies the Transaction Manager (TAM) when a transaction begins.

You configure the Input Controller by editing the **Input** section of the registry file. For more information, see "Configuring the Input Section in the Registry".

Registry Entries

Table 41–11 lists the Input Controller registry entries.

Table 41–11 Input Controller Registry Entries

Entry	Description	Mandatory
InputModule	Subgroup for the Input module. See "INP_GenericStream".	Yes
UnitsPerTransaction	Specifies the number of CDR input files that make up a transaction. By default, each CDR file forms its own transaction. This parameter only affects processing within the pipeline, and the number of output files match the number of CDR input files. The default is **1**. For more information on this parameter, see "Combining Multiple CDR Files Into One Transaction" in *BRM System Administrator's Guide*.	No

Sample Registry

```
Input
{
    UnitsPerTransaction = 2
    InputModule
    {
        ModuleName = INP_GenericStream
        Module
        {
            ...
        }
    }
}
```

NET_EM

The NET_EM module hosts an External Module (EM). This allows the NET_EM module to use the BRM API opcodes to transfer data between real-time rating opcodes and the real-time rerating, discounting, and zoning pipelines.

For more information, see the following "Configuring the NET_EM Module For Real-Time Processing" in *BRM System Administrator's Guide*.

For information about specific types of real-time processing, see:

- "Configuring a Real-Time Rerating Pipeline" in *BRM Setting Up Pricing and Rating*.
- Configuring a Real-Time Discounting Pipeline.
- "About Setting Up Zones" in *BRM Setting Up Pricing and Rating*.

Registry Entries

Table 41–12 lists the NET_EM registry entries.

Table 41–12 NET_EM Registry Entries

Entry	Description	Mandatory
FieldName	Use this entry for real-time rerating. Specifies the field in the event flist to be used to route the event. By using the "**.**" notation, you can specify a field at any level in the flist. The field identified by FieldName must be of type POID or String. See "Configuring NET_EM to Route Rerate Requests Based on the Event Field Value" in *BRM Setting Up Pricing and Rating*.	No
FieldValue	Use this entry for real-time rerating. Specifies the value of the field identified by **FieldName**. See "Configuring NET_EM to Route Rerate Requests Based on the Event Field Value" in *BRM Setting Up Pricing and Rating*.	No
NumberOfRTPipelines	Number of real-time pipelines. **Note:** This number must match the **NumberOfInstances** entry. See "Configuring Multiple Instances of a Pipeline" in *BRM System Administrator's Guide*.	Yes
OpcodeName	Specifies the opcode sending the event to NET_EM. For real-time discounting, use: PCM_OP_RATE_DISCOUNT_EVENT For real-time zoning, use: PCM_OP_RATE_GET_ZONEMAP_INFO For real-time rerating, use: PCM_OP_RATE_PIPELINE_EVENT See "Specifying The Type Of NET_EM Opcode Processing" in *BRM System Administrator's Guide*.	Yes
PipelineName	The pipeline to route the input to. Each real-time pipeline must have a unique name.	Yes
Port	Specifies the port number of the host machine running the NET_EM module.	Yes
Threads	Set this entry to the number of pipelines being managed by the NET_EM module. For example, if you have two pipelines, set this entry to 2.	Yes
UnixSockFile	Specifies the UNIX Sock file when the CM and the Pipeline Manager instance are running on the same machine.	Yes

Sample Registry Entry

```
DataPool
{
     RealtimePipeline
     {
          ModuleName = NET_EM
          Module
          {
               ThreadPool
               {
                    Port = 14579
                    Threads = 3
               }
```

```
                    ReratingOpcode
                    {
                         OpcodeName = PCM_OP_RATE_PIPELINE_EVENT
                         PipelineName = RealtimeReratingPipeline
                         NumberOfRTPipelines = 3
                    }
               }
          }
     }
```

Output Collection

Use the Output Collection module to handle output streams. See "About Configuring the Output Section in the Registry" and "Configuring EDR Output Processing".

Registry Entries

The only registry entries for the Output Collection configuration are the sections for each output stream, for example, OUT_DevNull, OUT_Reject, and OUT_GenericStream.

See the following:

- OUT_Reject
- OUT_DevNull
- OUT_GenericStream
- OUT_DB
- OUT_Realtime

Sample Registry

```
#------------------------------------------------
# Output Section
#------------------------------------------------
Output
{
   WriteDefaultEdr    = False
   DeleteEmptyFile    = True
     MaxErrorRates
     {
     }
     OutputCollection
     {
     #----------------------------------------
     # The DevNull stream
     #----------------------------------------
     DevNull
    ...{
```

Event Messages

Table 41–13 lists the Output Collection event messages.

Table 41–13 Output Collection Event Messages

Message	Description	Send/Receive
CMD_WRITE_LOG	An entry to the pipeline log has to be created.	Receive
REQ_STREAM_NUMBER	Determination of a specific stream number chosen by the first argument Name.	Receive from Output Collection. See "Output Collection".

Output Controller

Use the Output Controller to manage the output streams for its associated pipeline.

For more information, see "About Configuring the Output Section in the Registry".

Registry Entries

Table 41–14 lists the Output Controller registry entries.

Table 41–14 Output Controller Registry Entries

Entry	Description	Mandatory
MaxErrorRates	Subgroup for the maximum error rate entries. This section should list all error codes to monitor and their threshold values. For more information, see "Specifying the Maximum Errors Allowed in an Input File".	Yes
MultiThreading	Subgroup to configure multithreading in output processing. See the discussion on increasing Pipeline Manager throughput when an EDR is associated with multiple output streams in *BRM System Administrator's Guide*.	No
MultiThreading.Active	Specifies whether to enable multithreading in output processing: ■ **True** enables multithreading. ■ **False** disables multithreading. This is the default.	Yes
MultiThreading.NumberOfThreads	Specifies the number of threads the Output Controller creates to manage the output streams for its associated pipeline. For optimum results, Oracle suggests that you set the number of threads to twice the average number of streams associated with an input EDR.	Yes
MultiThreading.BatchSize	Specifies the size of the batch in terms of number of EDRs: ■ **0** indicates that the Output Controller does not operate in a batch mode. ■ A value greater than **0** indicates that the Output Controller operates in the batch mode with the batch size equal to the specified value. Oracle suggests that **BatchSize** be greater than or equal to **BlockSize** if **BlockTransfer** is set to **True**; otherwise, **BatchSize** should be equal to the size of the output buffer. For more information about the **BlockSize**, **BlockTransfer**, and **OutputBuffer** entries, see the discussion on configuring pipeline buffers in *BRM System Administrator's Guide*. The **BatchSize** value should be directly proportional to **NumberOfThreads** and inversely proportional to the EDR enrichment rate.	Yes

Table 41–14 (Cont.) Output Controller Registry Entries

Entry	Description	Mandatory
OutputCollection	Subgroup for the Output Collection module entries. See: ■ About Configuring the Output Section in the Registry ■ Output Collection	Yes
OutputLog	Subgroup for the stream log entries. See: ■ "About Pipeline Manager Log Files" in *BRM System Administrator's Guide* ■ LOG	Yes
SequenceGeneration	Specifies whether the pipeline generates one output file per CDR input file or one output file for an entire transaction. ■ **Unit** generates one output file per CDR input file. ■ **Transaction** generates one output file per transaction. For example, if you combine 5 CDR input files into one transaction, the pipeline creates only 1 output file. The default is **Unit**. See "Combining Multiple CDR Files Into One Transaction" in *BRM System Administrator's Guide*.	No
Sequencer	Specifies the name of the Sequencer for performing sequence checking. This Sequencer must be defined in the **SequencerPool** section of the registry file. See: ■ "Configuring Sequence Checking" in *BRM System Administrator's Guide* ■ Sequencer	No
Statistic	Subgroup to control the statistics related to Pipeline Manager's EDR processing rate. You can view these statistics in the output logs, HTTP browser, and the console output from the SNMP binaries. See "About Configuring Statistics Information in the Output Section".	No

Sample Registry Entry for the Multithreaded Mode

```
Output
{
    MaxErrorRates
    {
        ERR_CUST_NOT_FOUND = 10
    }

   MultiThreading
   {
       Active = True
       NumberOfThreads = 5
       BatchSize = 500
    }
    Statistic
    {
```

```
        EdrCountCriteria = ALL
    }...
    OutputCollection
      ...
    OutputLog
    {
         ...
    }
    SequenceGeneration = Unit
    Sequencer = SequenceCheck1
    ...
}
```

Sample Registry Entry for the Single-Threaded Mode

```
Output
{
    MaxErrorRates
    {
        ERR_CUST_NOT_FOUND = 10
    }

    Statistic
    {
        EdrCountCriteria = ALL
    }...
    OutputCollection
      ...
    OutputLog
    {
         ...
    }
    SequenceGeneration = Unit
    Sequencer = SequenceCheck1
    ...
}
```

ParallelLoadManager

Use the ParallelLoadManager module to load your pipelines, data modules, and function modules in parallel.

For more information, see "Reducing Startup Times with Parallel Loading" in *BRM System Administrator's Guide*.

Registry Entries

Table 41–15 lists the ParallelLoadManager registry entries.

Table 41–15 ParallelLoadManager Registry Entries

Entry	Description	Mandatory
Active	Specifies whether to load the pipelines, data modules, and function modules in parallel. ■ **TRUE** specifies to use parallel loading. ■ **FALSE** specifies to use sequential loading. If the entry is missing, parallel loading is disabled.	Yes
NumberOfThreads	Specifies the number of threads Pipeline Manager uses to load your pipelines, data modules, and function modules.	Yes

Sample Registry

```
ifw
{
   ...
   ParallelLoadManager
   {
      Active = TRUE
      NumberOfThreads = 4
   }
   …
}
```

Pipeline Controller

Use the Pipeline Controller to control its associated pipeline.

For more information, see:

- "About Configuring Pipelines" in *BRM System Administrator's Guide*
- "About the Pipeline Controller" in *BRM Concepts*

Registry Entries

Table 41–16 lists the Pipeline Controller registry entries.

Table 41–16 Pipeline Controller Registry Entries

Entry	Description	Mandatory
Active	Activates or deactivates processing in the pipeline. ■ **True** activates pipeline processing. ■ **False** deactivates pipeline processing.	Yes
CountryCode	Specifies the valid country code for this pipeline. The default is **49** for Germany.	No
DataDescription	Speicifes the stream format description and mapping files See: ■ Configuring the Input DataDescription Registry Section ■ Configuring the Output DataDescription Registry Section	Yes

Table 41–16 (Cont.) Pipeline Controller Registry Entries

Entry	Description	Mandatory
EdrFactory	Subgroup for the EDR Factory. See: ■ "About the EDR Factory" in *BRM Concepts* ■ EDR Factory	Yes
Functions	Subgroup for the function pool entries. See "About configuring function modules" in *BRM System Administrator's Guide*.	Yes
Input	Subgroup for the Input Controller. See: ■ Configuring the Input Section in the Registry ■ Input Controller.	Yes
Instances	Specifies multiple instances of a specific pipeline. See "Configuring Multiple Instances of a Pipeline" in *BRM System Administrator's Guide*.	No
InternationalAccessCode	Specifies the international dial prefix. The default is **00** for Germany. **Note:** You can list multiple access codes by using a comma as a delimiter. For example: **00,001,002**.	No
InternationalAccessCodeSign	Specifies the international access code sign. The default is **+**.	No
InputBuffer	Subgroup that contains the entries for the buffer between the input module and function modules. See "Configuring Pipeline Buffers" in *BRM System Administrator's Guide*.	Yes
MobileCountryCode	Specifies the valid mobile country code for this pipeline. The default is **262** for Germany.	No
MultiThreaded	Specifies whether the pipeline uses multithreaded or single-threaded processing. The default is **True**. ■ **True** specifies multithreaded processing. ■ **False** specifies single-threaded processing. See "Configuring Single-Threaded or Multithreaded Operation" in *BRM System Administrator's Guide*.	No
NationalAccessCode	Specifies the dial prefix for national calls. The default is **0** for Germany.	No
NetworkDestinationCode	Specifies the network destination code, which identifies the home network for roaming calls. The default is **172** for D2.	No
NoOutputUsed	Specifies whether to load the Output module. Set this entry to **True** only when you are using single-threaded processing and the Input and Output modules are combined into one module. The default is **False**. ■ **True** specifies to *not* load the Output module. ■ **False** specifies to load the Output module.	No
Output	Subgroup that contains Output Controller entries. See: ■ About Configuring Output Processing ■ Output Controller.	Yes

Table 41–16 (Cont.) Pipeline Controller Registry Entries

Entry	Description	Mandatory
OutputBuffer	Subgroup that contains the entries for the buffer between the function modules and the output module. See "Configuring Pipeline Buffers" in *BRM System Administrator's Guide*.	Yes
Pipeline_Name	Name of the pipeline.	Yes
PipelineLog	Subgroup that contains pipeline log entries. See: ■ "About Pipeline Manager Log Files" in *BRM System Administrator's Guide* ■ "LOG".	Yes
RejectStream	Specifies the name of the rejection stream. This stream must be defined in the output module. See: ■ Configuring Standard Recycling ■ Recycling EDRs in Pipeline-Only Systems	Yes
TransactionManager	Specifies the subsection for the Transaction Manager. See: ■ "About the Transaction Manager" in *BRM System Administrator's Guide* ■ Transaction Manager	N/A

Sample Registry

```
Pipelines
{
    Pipeline01
    {
        Active = TRUE
        MultiThreaded = TRUE
        CountryCode = 49
        MobileCountryCode = 262
        NationalAccessCode = 0
        InternationalAccessCode = 00
        InternationalAccessCodeSign = +
        NetworkDestinationCode = 171
        RejectStream = XXX
        PipelineLog
        {
            ModuleName = LOG
            Module
            {
            }
        }
        InputBuffer
        {
            Size = 1000
        }
        OutputBuffer
        {
            Size = 1000
        }
        Input
        {
```

```
    InputModule
    {
        ModuleName = XXX
        ModuleStart = XXX
        Module
        {
            ...
        }
    }
} // Input
Functions
{
    FCI
    {
        FunctionPool
        {
            Function01
            {
                ...
            }
            Function02
            {
                ...
            }
        }
    }
} // Functions
Output
{
    ...
    Outputcollection
    {
        Output1
        {
            ModuleName = XXX
            ModuleStart = XXX
            Module
            {
                ...
            }
        }
        RejectOutput
        {
            ModuleName = XXX
            ModuleStart = XXX
            Module
            {
                ...
            }
        }
        Output2
        {
            ModuleName = XXX
            ModuleStart = XXX
            Module
            {
                ...
            }
        }
        DevNull
```

```
                                {
                                    ModuleName = XXX
                                    ModuleStart = XXX
                                    Module
                                    {
                                        ...
                                    }
                                }
                                ...
                            }
                            OutputLog
                            {
                                ...
                            }
                            Sequencer
                            {
                                ...
                            }
                        } #Output
                    } #Pipeline01
                } #Pipelines
```

Sample Registry for Multiple Instances of a Pipeline

This sample shows how to configure multiple instances of a pipeline. For more information, see "Configuring Multiple Instances of a Pipeline" in *BRM System Administrator's Guide*.

```
ifw
{
...
  Pipelines
  {
    Instances
    {
      RealtimeReratingPipelineGPRS
      {
        NumberOfInstances = 3
        InstanceSpecificRegistries
        {
          Entry1 = TransactionManager.BinaryLogFileName
          Entry2 = PipelineLog.Module.ITO.FileName
          Entry3 = OutputLog.FileName
          Entry4 = Functions.Standard.FunctionPool.EdrDump_Initial.Module.FileName
          Entry4 = Functions.Standard.FunctionPool.EdrDump_PostDiscounting.Module.FileName
          Entry5 = Functions.Standard.FunctionPool.EdrDump_PreDiscounting.Module.FileName
          Entry6 = Functions.Standard.FunctionPool.EdrDump_PostRating.Module.FileName
          Entry7 = Functions.Standard.FunctionPool.EdrDump_PreRating.Module.FileName
        }
...
```

Semaphore Entries

Table 41–17 lists the Pipeline Controller Semaphore entries.

Table 41–17 Pipeline Controller Semaphore Entries

Entry	Description
Active	Activates or deactivates processing in the pipeline.

Sample Semaphore Entry

```
ifw.Pipelines.ALL_RATE.Active = True
```

Event Messages

Table 41–18 lists the Pipeline Controller event messages.

Table 41–18 Pipeline Controller Event Messages

Message	Trigger	Parameter
EVT_PIPELINE_START	The pipeline was started.	Pipeline name from the registry.
EVT_PIPELINE_STOP	The pipeline was stopped.	Pipeline name from the registry.

Sequencer

Use the Sequencer to prevent Pipeline Manager from processing the same CDR file twice and to add tracking information to output streams. For information, see "Configuring Sequence Checking" in *BRM System Administrator's Guide*.

Dependencies

When you configure the Sequencer to store state and log data in database tables, this module requires a connection to the Pipeline Manager database.

To assign a Sequencer to a pipeline, you must also configure the **Output** section of the registry file:

- To assign a sequence checker to a pipeline, use the **Sequencer** registry entry in the Output Controller module. For information, see "Output Controller".
- To assign a sequence generator to a pipeline, use the **Sequencer** registry entry in the output module. For information, see "OUT_GenericStream".

Registry Entries

Table 41–19 lists the Sequencer registry entries.

Table 41–19 Sequencer Registry Entries

Entry	Description	Mandatory
SequencerInstance	Specifies the name of the Sequencer instance.	Yes
Source	Specifies whether Sequencer state and log data are stored in files or in database tables. Values are: ■ **File** ■ **Database**	Yes
Controller	Subgroup that contains Controller entries	Yes

Table 41–19 (Cont.) Sequencer Registry Entries

Entry	Description	Mandatory
Controller.SequencerType	Specifies whether the Sequencer performs sequence checking or sequence generation: ■ **Check** configures the Sequencer to perform sequence checking. ■ **Generation** configures the Sequencer to perform sequence generation. See "Configuring Sequence Checking" in *BRM System Administrator's Guide*.	Yes
Controller.DatabaseConnection	Specifies a connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes, only if **Source** = **Database**.
Controller.ReuseGap	Specifies whether the Sequencer assigns skipped sequence numbers to output files. ■ **True** directs the Sequencer to reuse skipped sequence numbers by assigning the skipped sequence numbers to other CDRs. ■ **False** directs the Sequencer to never reuse skipped sequence numbers. The default is **False**.	No
Controller.SequenceLength	Specifies the length of the incoming CDR file's sequence number. The default is **6**.	No
Controller.FileName	Specifies the name of the Sequencer state file. This file stores state information for one Sequencer instance. **Important:** You must create one state file for each Sequencer instance. Otherwise, the Sequencer fails.	Yes, only if **Source** = **File**
Controller.FilePath	Specifies the path to the Sequencer state file.	Yes, only if **Source** = **File**.
Controller.Log	Subgroup that contains Sequencer log file entries.	Yes
Controller.Log.FileName	Specifies the name of the Sequencer log file. **Important:** You must create one Sequencer log file for each Sequencer instance. Otherwise, the Sequencer fails.	Yes, only if **Source** = **File**
Controller.Log.FilePath	Specifies the path to the Sequencer log file.	Yes, only if **Source** = **File**
Controller.UseGapAtStartup	Specifies whether to add a gap for the skipped sequence numbers starting from 0. This entry is required only when the **SequencerType** field is **Check** and the **ReuseGap** field is **True**. You can use this entry even if you have set the **Seq Original Number** field to **0**. ■ **True**. This value directs the Sequencer to add a gap for the skipped sequence numbers starting from 0. ■ **False**. The default value. This value directs the Sequencer to never add a gap for the skipped sequence numbers.	No

Sample Registry for File Storage

```
SequencerPool
{
    SequencerInstance
    {
```

```
            Source = File

            Controller
            {
                SequencerType = Check
                ReuseGap = True
                SequenceLength = 7
                FileName = sequence.dat
                FilePath = /opt/portal/ifw/sequencer

                Log
                {
                    FileName = sequence.log
                    FilePath = /opt/portal/ifw/logs
                }
            }
        }
    }
```

Sample Registry Entry for Database Storage

```
SequencerPool
{
    SequencerInstance
    {
        Source = Database

        Controller
        {
            SequencerType = Generation
            DatabaseConnection = DatabaseModule
            ReuseGap = False
            SequenceLength = 10
        }
    }
}
```

Database Tables

The Sequencer uses the following tables:

- IFW_PIPELINE
- IFW_SEQCHECK
- IFW_SEQLOG_IN
- IFW_SEQLOG_OUT

For information about the fields in database tables, see the documentation in *Pipeline_Home***database**.

Transaction Manager

Use the Transaction Manager to coordinate the state of all transactional modules and components in one pipeline. For information, see "About Pipeline Manager Transactions" in *BRM System Administrator's Guide*.

Dependencies

Requires a reference to the Transaction ID Controller. For information, see "Transaction ID Controller".

Registry Entries

Table 41–20 lists the Transaction Manager registry entries.

Table 41–20 Transaction Manager Registry Entries

Entry	Description	Mandatory?
BinaryLogFileName	Specifies the path and file name of the binary log file, which is used to persist and restore open transactions. **Important:** If you use multiple pipelines, you cannot use the same file for different pipelines. See "About Transaction Log Files" in *BRM System Administrator's Guide*.	Yes
RedoEnabled	Specifies whether the redo mechanism is enabled. **True** enables the redo mechanism. **False** disables the redo mechanism. See "About Cancelling Transactions When a Rollback Occurs" in *BRM System Administrator's Guide*.	Yes
SingleTransaction	Specifies whether only one pipeline transaction is allowed at a time. **True** specifies that only one pipeline transaction can be active at one time. The TAM blocks any new transactions from starting while a transaction is in progress. **False** specifies that multiple pipeline transactions can be active at one time.	Yes
WriteToLogEnabled	Specifies whether the Transaction Manager writes status information to the pipeline log file. **True** enables writing to the pipeline log file. **False** disables writing to the pipeline log file.	No

Sample Registry

```
Pipelines
{
    PipelineName
    {
        TransactionManager
        {
            RedoEnabled = True
            SingleTransaction = True
            BinaryLogFileName = ./
            WriteToLogEnabled = False
        }
    }
}
```

Transaction ID Database Generator

Use the Transaction ID Database Generator to store transaction IDs in database tables. For information, see "Configuring the Transaction ID Controller" in *BRM System Administrator's Guide*.

Dependencies

Requires a connection to the Pipeline Manager database.

Registry Entries

Table 41–21 lists the Transaction ID Database Generator registry entries.

Table 41–21 Transaction ID Database Generator Registry Entry

Entry	Description	Mandatory?
DataConnection	Specifies a connection to the Pipeline Manager database. See "Connecting a Module to a Database" in *BRM System Administrator's Guide*.	Yes

Sample Registry

```
TransactionIdController
{
    Source = Database
    Generator
    {
        DataConnection = ifw.DataPool.Login
    }
}
```

Database Tables

The TAM_TransIdDbGenerator module uses the IFW_TAM database table.

For information about the fields in database tables, see the documentation in *Pipeline_Home***database**.

Transaction ID File Generator

Use the Transaction ID File Generator to store transaction IDs in a file. For information, see "Configuring the Transaction ID Controller" in *BRM System Administrator's Guide*.

Registry Entries

Table 41–22 lists the Transaction ID File Generator registry entries.

Table 41–22 Transaction ID File Generator Registry Entries

Entry	Description	Mandatory?
FileName	Specifies the path and file name of the Transaction ID Controller state file. See "About The Transaction ID State File And Table" in *BRM System Administrator's Guide*.	Yes
Increment	Specifies the number of transaction IDs that are cached. See "About Storing IDs in Cache" in *BRM System Administrator's Guide*.	Yes

Sample Registry

```
TransactionIdController
{
    Source = File
    Generator
    {
        FileName = /data/system/info/transIdInfo.dat
        Increment = 10
    }
}
```

Transaction ID Controller

Use the Transaction ID Controller to generate transaction IDs for all pipelines. For information, see "Configuring the Transaction ID Controller" in *BRM System Administrator's Guide*.

Registry Entries

Table 41–23 lists the Transaction ID Controller registry entries.

Table 41–23 Transaction ID Controller Registry Entries

Entry	Description	Mandatory?
Generator	Subgroup for the generator entries. See: ■ Transaction ID File Generator ■ Transaction ID Database Generator	Yes
Source	Specifies whether the Transaction ID Controller stores transaction IDs in files or database tables. Values are: ■ **File** ■ **Database** See "About The Transaction ID State File And Table" in *BRM System Administrator's Guide*.	Yes

Sample Registry for File Storage

```
TransactionIdController
{
    Source = File
    Generator
```

```
    {
        FileName = /data/system/info/transIdInfo.dat
        Increment = 10
    }
}
```

Sample Registry for Database Storage

```
TransactionIdController
{
    Source = Database
    Generator
    {
        DataConnection = ifw.DataPool.Login
    }
}
```

42

Pipeline Manager Utilities

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager utilities.

Database Loader

The **Database Loader** utility loads and unloads aggregation data into and from a database.

For information about aggregation, see "Setting Up Pipeline Aggregation".

Dependencies

This utility needs a connection to the DBC database module, and the DBL library (**libDBLXXX.so**). See "Database Connect (DBC)".

Location

Pipeline_Home/**tools**

where *Pipeline_Home* is the directory in which you installed Pipeline Manager.

Syntax

```
dbLoader -r registry [-f files] [-u]
```

Parameters

-r
Defines the registry file.

-f
Defines the file pattern (regular expression).

-u
Undo mode.

Registry Entries

Table 42–1 lists the Database Loader registry entries.

Table 42–1 Database Loader Registry Entries

Entry	Description	Mandatory
BULKSIZE	Specifies the Oracle array size for bulk inserts (loadmode 2 and 3).	Yes
DIRECTIONMODE	Defines the selection order of the control files (1 file name, 2 sequence).	Yes
FILES.ARCHIVE.PATH	Specifies the path where the successfully loaded files are stored.	Yes
FILES.ARCHIVE.SUFFIX	Specifies the suffix of the successfully loaded data files.	Yes
FILES.BAD.PATH	Specifies the path where the bad files are stored.	Yes
FILES.BAD.SUFFIX	Specifies the suffix of the bad data files.	Yes
FILES.CONTROL.PATH	Specifies the path for the input aggregate control files.	Yes
FILES.CONTROL.SUFFIX	Specifies the suffix of the input aggregate control files.	Yes
FILES.DATA.PATH	Specifies the path for the input aggregate data files.	Yes

Table 42–1 (Cont.) Database Loader Registry Entries

Entry	Description	Mandatory
FILES.DATA.SUFFIX	Specifies the suffix of the input aggregate data files.	Yes
FILES.MERGE.PATH	Specifies the path where the source merge data files are stored.	Yes
FILES.MERGE.SUFFIX	Specifies the suffix of the source data files before merging/sorting.	Yes
FILES.REJECT.PATH	Specifies the path where the rejected files are stored.	Yes
FILES.REJECT.SUFFIX	Specifies the suffix of the rejected data files.	Yes
LOADMODE	Specifies how to load data: ■ **1**: Single row updates and inserts. ■ **2**: Single row updates and bulk inserts. ■ **3**: Single row updates and bulk inserts. Before loading, the files can be merged or sorted and split into smaller pieces. Undo mode is always 1.	Yes
MAXSPLITLINES	Specifies the maximum number of lines per data file after splitting (loadmode 3).	Yes
ROLLBACKSEGMENT	Specifies which Oracle rollback segment to use when loading the database. How to set this entry depends on your database software setup. If your Oracle9i database uses automatic undo management, comment out or remove this registry entry. If your database does not use undo management, specify a rollback segment. The Oracle9i software provides an automatic undo management feature, which creates undo tablespaces rather than rollback segments for undo information. If you use this undo management feature *and* specify a rollback segment for the Pipeline Manager **Database Loader** utility, the utility fails when it attempts to load the database. To prevent this problem, don't specify a rollback segment.	No
SORTCMD	Specifies the external sort command (loadmode 3).	Yes
SORTING	Specifies a flag if files of identical structure should me merged and sorted (loadmode 3).	Yes
SORTMAXFILESIZE	Specifies the maximum destination size of the merged and sorted files (loadmode 3).	Yes
SORTTMPDIR	Specifies the path where sort stores temporary files (loadmode 3).	Yes
SPLITTING	Specifies whether to split data files before loading (reduce transaction size) (loadmode 3).	Yes

Sample Registry

```
DBLOADER
  {
    Active              = TRUE
    ProcessLoopTimeout  = 10
    QueueRequestTimeout = 0
    Instrumentation
    {
      #------------------------------------------------------------
```

```
  # ProbeBroker registry entries.
  # ProbeInfoFilePath - The path that contains all probe
  # info files used by instrumented objects.
  #------------------------------------------------------------
  ProbeBroker
  {
    ProbeInfoFilePath = ./instrumentation
  }
}
LogMessageTable
{
  MessageFilePath   = ./etc
  MessageFileSuffix = .msg
}
DiagnosticDataHandler
{
  DiagnosticFilePath = ./log
  DiagnosticFileName = diagnostic.dat
}
#
# main parameter
#
DIRECTIONMODE   = 2
LOADMODE        = 2
BULKSIZE        = 100
ROLLBACKSEGMENT = R04
SORTING         = true
SORTCMD         = sort
SORTTMPDIR      = .
SORTMAXFILESIZE = 2000000000
SPLITTING       = true
MAXSPLITLINES   = 40000
#
# database section
#
DataPool
{
  Database
  {
    ModuleName = DBC
    Module
    {
      DatabaseName = $ORACLE_SID
      UserName     = AGGREGATOR
      PassWord     = 595EA7DFC8C6C3D8A1AFDADC0600180F12771D73
      AccessLib    = oci11g72
      Connections  = 1
    }
  }
}
#
# File Section
#
FILES
{
  CONTROL
  {
    PATH    = ./data/aggregate/cntl
    SUFFIX  = .ctl
  }
```

```
    DATA
    {
      PATH    = ./data/aggregate/done
      SUFFIX  = .dat
    }
    REJECT
    {
      PATH      = ./data/aggregate/reject
      SUFFIX    = .rej
      THRESHOLD = 85
    }
    REJECT_HANDLE
    {
      PATH      = ./data/aggregate/reject
      SUFFIX    = .rej
      THRESHOLD = 85
    }
    ARCHIVE
    {
      PATH    = ./data/aggregate/archive
      SUFFIX  = .arc
    }
    BAD
    {
      PATH    = ./data/aggregate/bad
      SUFFIX  = .bad
    }
    MERGE
    {
      PATH    = ./data/aggregate/merge
      SUFFIX  = .mrg
    }
  }
  #
  # log section
  #
  ProcessLog
  {
    ModuleName = LOG
    Module
    {
      ITO
      {
        MessageFilePath = etc
        MessageFilePrefix = error
        MessageFileSuffix = error.msg
        FilePath = ./data/aggregate/log
        FileName = process
        FilePrefix = DBL_
        FileSuffix = .log
        ProcessName = dbLoader
        MessageGroup = DBLOADER
      }
      Buffer
      {
        Size = 1000
      }
    }
  }
}
```

db2irules.pl

Use the **db2irules.pl** script to extract rule sets from the Pipeline Manager database to the Rule Set XML file.

See "Importing and Exporting Validation Rules" in *BRM Developer's Guide*.

> **Important:** This utility uses DBI and DBD drivers which are not part of the Pipeline Manager installation. You download these drivers from `http://www.cpan.org` and compile and install them separately.

Location

Pipeline_Home/**tools/IRules2Db/db2irules.pl**

> **Important:** Since there are dependencies between the **db2irules.pl** script and the **PerlParser.pm** XML library located in the same directory as the script. Always run the script from this location.

Syntax

```
db2irules.pl [-d] [-u] dbi:dcs password user_name file_path rule_set_id
```

Parameters

If you start the **db2irules.pl** script without any parameters, a usage description and an example for each parameter are displayed.

dcs

The database connection string. This required parameter enables the script to access the database. The string is different for each database type. Example dcs for Oracle:

```
Oracle:orcl
```

> **Note:** The database connection string is the standard database access module for Perl scripts. It defines a set of methods, variables, and conventions that provide a consistent database interface, independent of the actual database being used.

password

This parameter is required to connect to the database. It is your standard Pipeline Manager database password.

user_name

This parameter is required to connect to the database. It is your standard user name for the Pipeline Manager database.

file_path

Use this parameter to specify where you want to export the rule set. If you want to use the same directory in which the rule set is stored, use **./** as file path. If you don't set this parameter, the rule set is exported automatically to the current directory.

rule_set_id

Use this parameter to extract only one specific rule set, which is identified by its unique ID. If you don't set this parameter, the **db2irules.pl** script will extract all rule sets from the database. If you use this parameter, you must use the *file_path* parameter. This *rule_set_id* refers to the IFW_RULESET.RULESET database field.

-u

This parameter creates a unique file name for the rule set, based on date and time. It uses the following format: *RULESET_yyyy-mm-dd_hh-mm-ss.xml*. Use this parameter to ensure that you do not override an existing XML file when extracting rule sets. If the file name for a rule set contains spaces, replace them with the underscore character (_).

Example:

```
db2irules.pl -u dbi:Oracle:orcl scott tiger TAP3_VAL
```

-d

This parameter deletes the specified rule set(s) from the database after you extracted them. If you use this parameter, a transaction is opened with the database. If any of the rule set deletes fail, the entire delete sequence is rolled back to preserve database integrity. If all rule set tables are deleted successfully, the transaction is committed to the database.

Example:

```
db2irules.pl -d -u dbi:Oracle:orcl scott tiger
```

Diagnostic Data Handler

Use Diagnostic Data Handler to get data about Pipeline Manager after a crash, exception, critical error, or while it is running.

For more information, see "Using The Diagnostic Data Handler To Get OMF Diagnostic Data" in *BRM System Administrator's Guide*.

Registry Entries

Table 42–2 lists the Diagnostic Data Handler registry entries.

Table 42–2 Diagnostic Data Handler Registry Entries

Entry	Description	Mandatory
DiagnosticFilePath	Path to the log file that is created by Diagnostic Data Handler.	Yes
DiagnosticFileName	File name of the log file that is created by Diagnostic Data Handler.	Yes

Sample Registry

```
DiagnosticDataHandler
{
   DiagnosticFilePath = ./log
   DiagnosticFileName = diagnostic.dat
}
```

irules2db.pl

Use the **irules2db.pl** script to insert a rule set from Validation Rules XML file into the Pipeline Manager database.

See "Importing and Exporting Validation Rules" in *BRM Developer's Guide*.

> **Important:** This utility uses DBI and DBD drivers which are not part of the Pipeline Manager installation. You download these drivers from `http://www.cpan.org` and compile and install them separately.

Location

Pipeline_Home/**tools/IRules2Db/irules2db.pl**

> **Important:** Since there are dependencies between the **irules2db.pl** script and the **PerlParser.pm** XML library which is located in the same directory as the script. Always run the script from this location.

Syntax

```
irules2db.pl [-f] dbi:dcs password user_name rule_set_name backup_file_path
```

Parameters

If you start the **irules2db.pl** script without any parameters, a usage description and an example for each parameter are displayed.

dcs

The database connection string. This required parameter enables the script to access the database. The string is different for each database type. Example dcs for Oracle:

```
Oracle:orcl
```

> **Note:** The database connection string is the standard database access module for Perl scripts. It defines a set of methods, variables, and conventions that provide a consistent database interface, independent of the actual database being used.

password

This parameter is required to connect to the database. It is your standard Pipeline Manager database password.

user_name

This parameter is required to connect to the database. It is your standard user name for the Pipeline Manager database.

rule_set_name

Use this parameter to specify the name of the Rule Set XML file that you want to import to the database. This parameter supports fully qualified and relative path names.

Examples:

- **./tap3_val.xml**
- **/home/data/tap3_val.xml**
- **/../files/tap3_val.xml**
- **tap3_val.xml**

backup_file_path

Use this parameter to specify the path for storing the extracted rule set before it is deleted from the database and then after modification inserted from the Rule Set XML file into the database. Use this parameter with the **-f** parameter.

-f

This parameter forces the rule set into the database. The **irules2db.pl** script connects to the database and starts parsing the Rule Set XML file. When it finds the name of the rule set, it calls the export script that contains the **-u** and **-d** parameters. If the **db2irules.pl** script finished successfully, the **irules2db.pl** script continues parsing the XML file and imports the rule set to the database. If any of the rule set columns fail to be inserted, the **irules2db.pl** script rolls back the transaction and exits. If all columns are inserted into the database successfully, the rule set for the transaction is committed.

LoadIfwConfig

Use this utility to extract data from or load data into the Pipeline Manager database. This enables you to:

- Migrate data from a legacy database to the Pipeline Manager database. See "Migrating Price List Data From Legacy Databases" in *BRM Setting Up Pricing and Rating*.
- Transfer data between Pipeline Manager databases; for example, from a test database to a production database. See "Transferring Data Between Pipeline Manager Databases".

> **Caution:** The 7.4 version of the **LoadIfwConfig** utility is not backwards-compatible with previous versions of the utility. Any data exported by a previous version of the utility must also be loaded with that same version. In addition, any custom scripts or procedures that are dependent on the utility's functionality might need to be modified to work with the 7.4 version.

The **LoadIfwConfig** utility can run in these modes:

- *Non-interactive mode*: You use commands that batch several related parts of the extracting or loading process. You must enter a full command, including the utility name for each set of actions.
- *Interactive mode*: You issue a command for each step in the process of extracting or loading. After you enter interactive mode, the prompt changes to an angle bracket and commands are single words for performing particular actions. You can view a list of the change sets that will be extracted or loaded.

Location

Pipeline_Home/**bin**

Syntax: Non-Interactive Mode

```
LoadIfwConfig   {-rall [-t Modifidate] | -r [-t Modifidate] | -p [f] | -u | -I}
                [-c] [-nodep] -i InputFile [-o OutputFile] [-h] [-v]
```

Parameters: Non-Interactive Mode

-rall [-t *Modifidate*
Extracts all objects from the Pipeline Manager database. This parameter does not require an input XML file.

Using **-t** *Modifidate* retrieves only pricing objects that were modified after the specified timestamp. Enter the time in the ISO-8601 format: *YYYY-MM-DD***T***hh:mm:ss* or *YYYY-MM-DD* with the server time zone as the default.

-r [-t *Modifidate*]
Extracts from the database the objects listed in *InputFile*.

Using **-t** *Modifidate* retrieves only pricing objects that were modified after the specified time. Enter the time in the ISO-8601 format: *YYYY-MM-DD***T***hh:mm:ss* or *YYYY-MM-DD* with the server time zone as the default.

-p [f]
Deletes objects from the database.

Using the **f** parameter turns off the delete confirmation.

-u
Updates the Pipeline Manager database. Data is not actually updated in the database until it is committed with the **-c** parameter.

-I
Inserts data into the Pipeline Manager database. Data is not actually inserted into the database until it is committed with the **-c** parameter.

-c
Commits the data to the database. You use this command in conjunction with the **-u** and **-I** parameters.

-nodep
Suppresses any object dependency relationships that you configured in the *Pipeline_Home*/**tools/XmlLoader/CustomConfig.xml** file. This allows the utility to extract from the database only those objects that meet your criteria and to ignore any dependent objects. For more information about object dependencies, see "About Specifying to Extract Child and Dependent Objects".

-i *InputFile*
When extracting pipeline data by using the **-r** or **-rall** parameter, this is the name of the XML file that specifies the list of objects to extract from the source Pipeline Manager database.

When loading pipeline data by using the **-u** or **-I** parameter, this is the name of the XML file that contains the data you are loading into the destination Pipeline Manager database.

When deleting pipeline data by using the **-p** parameter, this is the name of the XML file that specifies the list of objects to delete from the Pipeline Manager database.

-o *OutputFile*
Specifies the output file to which the Pipeline Manager data is extracted. By default, the utility writes the output to a file named **default.out** in the current directory.

-h
Displays help about using the utility.

-v
Displays information about successful or failed processing as the utility runs.

Syntax: Interactive Mode

```
LoadIfwConfig  [read InputFile] [write OutputFile] [retrieve_all [-t Modifidate]]
               [fetch [-t Modifidate]] [list] [delete] [commit] [update] [insert]
               [help] [nodep] [verbose on|off] [quit]
```

Parameters: Interactive Mode

read *InputFile*
Specifies to read the specified input file into internal memory.

write *OutputFile*
Specifies the output file to which the Pipeline Manager data is extracted. By default, the utility writes the output to a file named **default.out** in the current directory

retrieve_all [-t *Modifidate*]
Extracts all objects from the Pipeline Manager database.

Using **-t** *Modifidate* retrieves only pricing objects that were modified after the specified time. Enter the time in the ISO-8601 format: *YYYY*-*MM*-*DD***T***hh*:*mm*:*ss* or *YYYY*-*MM*-*DD* with the server time zone as the default.

fetch [-t *Modifidate*]
Extracts from the database the objects listed in internal memory. You use this parameter after you use the **read** parameter.

Using **-t** *Modifidate* retrieves only pricing objects that were modified after the specified time. Enter the time in the ISO-8601 format: *YYYY*-*MM*-*DD***T***hh*:*mm*:*ss* or *YYYY*-*MM*-*DD* with the server time zone as the default.

list
Lists the current pipeline data stored in internal memory.

delete
Deletes from the database the objects listed in *InputFile*.

commit
Commits the data to the database. You use this command in conjunction with the **update** and **Insert** parameters.

update
Updates the Pipeline Manager database. Data is not actually updated in the database until it is committed with the **commit** parameter.

insert
Inserts data into the Pipeline Manager database. Data is not actually inserted into the database until it is committed with the **commit** parameter.

help
Displays help about using the utility.

-nodep
Suppresses any object dependency relationships that you configured in the *Pipeline_Home***/tools/XmlLoader/CustomConfig.xml** file. This allows the utility to extract only those objects that meet your criteria and to ignore any dependent objects. For more information about object dependencies, see "About Specifying to Extract Child and Dependent Objects".

verbose [on | off]
Sets verbose information:

- **verbose on** displays the status of the command most recently executed.

  Use the **ProcessLog** section of the registry file to specify the name and location of the file where debug messages are written.

- **verbose off** displays the status only if there is an error.

quit
Quits from the utility.

Results

If the **LoadIfwConfig** utility is successful, it displays a confirmation message. If unsuccessful, it displays errors.

Memory Monitor

Use the Memory Monitor module to warn you when available system memory is low and to shut down Pipeline Manager when memory reaches a specified threshold.

For more information, see "Monitoring Pipeline Manager Memory Usage" in *BRM System Administrator's Guide*.

Registry Entries

Table 42–3 lists the Memory Monitor registry entries.

Table 42–3 Memory Monitor Registry Entries

Entry	Description	Mandatory
ScaleUnit	Specifies the unit for monitoring memory. ■ **P** specifies percentage. ■ **K** specifies Kilobytes. ■ **M** specifies MegaBytes.	Yes
ShutdownFreeMemLimit	Specifies the amount or percentage of remaining system memory that triggers Pipeline Manager to gracefully shut down. **Note:** For percentage, you must enter a value from 1 to 99 inclusive.	Yes
WarningFreeMemLimit	Specifies the amount or percentage of remaining system memory that triggers Pipeline Manager to issue a warning to the user. **Note:** For percentage, you must enter a value from 1 to 99 inclusive.	Yes

Sample Registry

```
ifw
{
    MemoryMonitor
    {
        ScaleUnit = P
        WarningFreeMemLimit = 10
        ShutdownFreeMemLimit = 5
    }
}
```

pin_container_to_stream_format

Use this utility to create EDR stream, input and output mapping, and input and output grammar files from an EDR container description file. FCT_CallAssembling then uses these files in the process of converting partially assembled call records to a new container description.

For more information on the process of converting EDRs to a new EDR container description, see "Upgrading Incomplete Calls to the New Container Description" in *BRM System Administrator's Guide*.

Location

BRM_Home/**bin**

where *BRM_Home* is the directory in which you installed BRM components.

Syntax

```
pin_container_to_stream_format -c container_description_filename -g grammar_file_
prefix -m mapping_file_prefix -s stream_file_prefix | -h
```

Parameters

-c *container_description_filename*
Specifies the container description file to use to generate a stream file and the mapping and grammar files. Replace *container_description_filename* with the container description file to use.

-g *grammar_file_prefix*
Creates the input and output grammar description files based on the container description file. Replace *grammar_file_prefix* with a prefix to add to the grammar filenames.

-m *mapping_file_prefix*
Creates the input and output mapping description files based on the container description file. Replace *mapping_file_prefix* with a prefix to add to the mapping filenames.

-s *stream_file_prefix*
Creates the stream description file based on the container description file. Replace *stream_file_prefix* with a prefix to add to the stream filename.

> **Important:** If you do not specify one or more of the **-g**, **-m**, or **-s** parameters, this utility generates the files using the container description filename as a prefix. However, if you specify these options, you must also specify their arguments. Otherwise this utility returns an error.

-h
Displays help for this utility.

Example

This example:

```
pin_container_to_stream_format -c containerDesc.dsc -g OLD_ -m OLD_ -s OLD_
```

Creates these files using the information in **containerDesc.dsc**:

- **OLD_Stream.dsc**
- **OLD_InGrammar.dsc**
- **OLD_OutGrammar.dsc**
- **OLD_InMap.dsc**
- **OLD_OutMap.dsc**

Results

The **pin_containter_to_stream_format** utility notifies you only if it encounters errors.

pin_recycle

Use this utility to search for failed EDRs in the BRM database and queue the EDRs for recycling or test recycling, or delete them. This utility can:

- Recycle calls from the same CDR file as part of the BRM standard recycling feature. For details, see "About the Standard Recycling Mechanism".
- Recycle all EDRs that contain the same recycle key as part of either Suspense Manager or standard recycling. For details, see "About Recycling Suspended EDRs after Rating Interruptions".
- Recycle all EDRs that have the same suspense reason code.

Important: To connect to the BRM database, the **pin_recycle** utility requires a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

This utility calls the suspense manger opcodes to actually perform the recycling. For more information, see "Suspense Manager FM standard opcodes" in *BRM Developer's Reference*.

Location

BRM_Home/**bin**

Syntax

```
pin_recycle [ -f CDR_file] [ -k recycle_key ] [ -d | -D| -r reason_code| -t ]
```

Parameters

-f *CDR_file*
Queues all the failed EDRs that arrived in a single CDR file. Pipeline Manager rates these calls as soon as it can.

-k *recycle_key*
Searches for and queues EDRs for rating that contain:

- The *recycle_key*, an application-specific string that is added to each EDR as it is suspended by Pipeline Manager. See "About Standard Recycling" for details.
- A status of **suspended**.

These EDRs are queued for rating by Pipeline Manager as soon as possible.

-d
Searches for and deletes all EDRs with a status of **succeeded** or **written off**.

-D
Searches for and deletes all EDRs with a status of **succeeded**, **written off**, or **suspended**.

-r *reason_code*
Searches for and recycles all EDRs that have the specified reason code.

-t
Specifies a test recycle. In test mode, **pin_recycle** creates a report about the processing, but does not make any changes to the database. Test results written to the directory and file you specified using the FCT_Suspense module **RecycleLog** registry entries. You must also set the FCT_Suspense **LogTestResults** registry entry for standard recycling implementations.

Results

This utility logs messages to **stdout**.

The following message is returned after you use **pin_recycle** to recycle EDRs:

```
pin_recycle tool, number_of_EDRs EDRs Submitted for Recycling
```

The following message is returned after you use **pin_recycle** to test recycle EDRs:

```
pin_recycle tool, number_of_EDRs EDRs submitted for test recycling
```

The following message is returned after you use **pin_recycle** to delete EDRs:

```
pin_recycle tool, number_of_EDRs suspended EDRs deleted
```

purge_np_data.p

Use this utility to purge existing records from the number portability data file that are older than a specified date and time. See "Purging and Reloading the Memory Records".

Location

Pipeline_Home/**bin**

Syntax

```
purge_np_data.pl NP_FileName TimeStamp [-b backup_filename] [-n][-help]
```

Parameters

NP_FileName
Specifies the name of the number portability data file that will be purged.

TimeStamp
Specifies the date prior to which all the number portability records are purged. After the data is purged, the number portability data file is updated with the purged data.

Format: *YYYYMMDDhhmmss*.

-b *backup_filename*
Specifies the name of the backup file that will contain the unpurged number portability records.

-n
Sorts in the ascending order of the CLI. Default sorting is in the ascending order of the time stamp.

-help
Displays the syntax and parameters for this utility.

Results

The **purge_np_data.pl** utility notifies you when it successfully purges the number portability data file. Otherwise, it displays an error message.

RoamingConfigGen64

Use this utility to retrieve the roaming partner data from the Pipeline Manager database and create the roaming configuration data file. The data file is used by the Instances module to configure multiple instances of sequencers, output streams, or system brands based on the template sections or entries in the roaming registry file.

For more information, see "About Configuring Multiple Instances of Sequencers, Output Streams, or System Brands" in *BRM System Administrator's Guide*.

Location

Pipeline_Home/**bin**

Syntax

```
RoamingConfigGen64 -l database_access_library -s server_name [-d database_name] -c operator_code
[-o output_path] [-b base_path] [-h]
```

Parameters

-l database_access_library
The database access library. For example, **liboci10g6312d.a** for Oracle on AIX.

-s *server_name*
Specifies the name of the host machine running the Pipeline Manager database.

-d *database_name*
Specifies the database name of the Pipeline Manager database. The default is an empty string (' ').

-c *operator_code*
Specifies the home network operator code. The default is **PORTL**.

-o *output_path*
Specifies the output path for the data file generated by the **RoamingConfigGen64** utility. By default, the data file is saved in the *Pipeline_Home*/**conf/** directory.

-b *base_path*
Specifies the base path to the directory for Transferred Account Procedure (TAP) and Near Real Time Roaming Data Exchange (NRTRDE) output files. The default path is *Pipeline_Home*/**data/outcollect/**.

For example, if the base path is *Pipeline_Home*/**data/outcollect/**, the following new subdirectories are created in the *Pipeline_Home*/**data/outcollect/** directory:

- **tapout/** for TAP output files
- **nrtrdeout/** for NRTRDE output files

-h
Displays the syntax and parameters for this utility.

> **Note:** When prompted, enter the database user name and password.

Example

```
RoamingConfigGen64 -l liboci10g6312d.so -s $ORACLE_SID -c EUR01
```

where:

- **liboci10g6312d.so** is the database access library.
- **$ORACLE_SID** is the database alias.
- **EUR01** is the home network operator code.

Results

The **RoamingConfigGen64** utility creates the roaming configuration data file. Otherwise, it displays an error message.

settlement_extract

Use this utility to retrieve roaming settlement information from the IC-Daily tables in the Pipeline Manager database. When Pipeline Manager rates roaming usage, it stores the amounts owed each roaming partner in the IC-Daily tables.

Important: To ensure only unbilled events are extracted, before running this utility, you must close the bill run for each roaming partner account. You close the bill run by using the Pricing Center. See "Closing a Billrun" in *BRM Configuring Roaming in Pipeline Manager*.

For more information about roaming and settlement, see "About Rating Roaming Events" in *BRM Configuring Roaming in Pipeline Manager*.

This utility creates one file containing all settlement information stored in the Pipeline Manager database that has not already been extracted. The settlement information includes the amounts owed to each network that was used for roaming calls.

Note: To connect to the BRM database, the **settlement_extract** utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

Important:

- This utility requires Perl version 5.004_00.
- This utility uses DBI and DBD drivers which are not part of the Pipeline Manager installation. You download these drivers from www.cpan.org and compile and install them separately.
- (HP-UX only) Before running this utility, you must load the **libjava.so** library. One way of doing this is to set the LD_PRELOAD environment variable to point to the library file:

For example:

```
# setenv LD_PRELOAD /u01/app/oracle/product/817/JRE/lib/PA_
RISC/native_threads/libjava.so
```

Location

BRM_Home/**apps/uel**

Syntax

settlement_extract.pl [**-u**] **dbi:***dcs username password* [*filepath*]

Parameters

-u
Creates a unique file name for the new file using the current time. The format of the file name is:

"settlement_YYYY-MM-DD_hh-mm-ss.txt"

dcs
The database connection string. This required parameter enables the script to access the database. The string is different for each database type. Example dcs for Oracle:

```
Oracle:orcl
```

> **Note:** The database connection string is the standard database access module for Perl scripts. It defines a set of methods, variables, and conventions that provide a consistent database interface, independent of the actual database being used.

username
The database username.

password
The database password.

filepath
The location where the file should be written to. If you don't include this parameter, the file is written to the current directory.

Results

Creates a roaming settlement data file and reports success or displays an error.

stateconfigtool

Use this utility to load state configuration (**state.config**) files for use with the Pricing Center Pipeline Manager data migration feature.

Important: Before you run **stateconfigtool**, make sure that the following files are listed in your system CLASSPATH environment variable:

- **msbase.jar**
- **msutil.jar**

For more information, see Migrating pipeline pricing data in *BRM Pricing Center Online Help*.

Location

*Pipeline_Home***/tools/StateConfigTool**

where, *Pipeline_Home* is the directory where Pipeline Manager is installed.

Syntax

```
stateconfigtool -f file_name -d database_type -h host -n port -u user_name -p password -i database_id
```

Parameters

-f

The path and file name of the of the **state.config** file to be loaded. This file contains descriptions about changeset state transitions, such as currentState, nextState, and Action.

The default directory is *Pipeline_Home***/tools/StateConfigTool**.

-d

The database type. The supported database is **oracle**.

-h

The host name of the computer running the Pipeline Manager database.

-n

The port number used by the Pipeline Manager database.

-u

The login name for connecting to the database.

-p

The password for the specified user name.

-i

The database ID of the Pipeline Manager database.

Results

The utility loads the contents of the state.config into the Pipeline Manager database. The states defined in the file become available in the Change Set Manager when it is restarted.

Related Topics

See "Understanding the Change Set Life Cycle" in *BRM Configuring Pipeline Rating and Discounting*.

StopRapGen

The **StopRapGen** utility searches the database to collect information required by the Stop RAP Generator pipeline to create Stop Return Returned Account Procedure (RAP) files.

It retrieves information on the following:

- Transferred Account Procedure (TAP) files that were received by BRM and stored in the database more than seven days ago
- Stop Return RAP files that were generated by BRM and sent more than seven days ago to the Visited Public Mobile Network (VPMN) operator.

> **Note:** The output from the **StopRapGen** utility is used by the Stop RAP Generator pipeline to generate the Stop Return RAP file.
>
> Use the **StopRapGen** utility along with the Stop RAP Generator pipeline.

Location

Pipeline_Home/**bin**

where *Pipeline_Home* is the directory in which you installed Pipeline Manager.

Syntax

```
StopRapGen64 database_access_library server_name database_name path [prefix]
[days]
```

Parameters

database_access_library
The database access library. For example, **liboci10g6312d.a** for Oracle on AIX.

server_name
Specifies the name of the host machine running the Pipeline Manager database.

database_name
Specifies the database ID of the Pipeline Manager database.

path
Specifies the output directory of the flat file generated by the **StopRapGen** utility. This file is used by the Stop RAP Generator pipeline.

> **Tip:** The output directory for the **StopRapGen** utility should be the same as the input directory for the Stop RAP Generator pipeline.

prefix
Specifies the prefix to be added to the output flat file. The default prefix is **RC**.

days
Specifies the number of days to consider for generating a Stop Return RAP file. The default is 7, in accordance with the RAP standard.

Example

```
StopRapGen64 liboci10g6312d.so $ORACLE_SID '' ./data/stoprap/in
```

where:

- **liboci10g6312d.so** is the database access library.
- **$ORACLE_SID** is the database alias.
- ' ' is the empty string passed in as the database name.
- **.data/stoprap/in** is the output directory of the sample usage data for the **StopRapGen** utility (the flat file it generates). This is also the input directory of the Stop RAP Generator pipeline.

Results

The **StopRapGen** utility generates the input required by the Stop RAP Generator pipeline.

ZoneDBImport

The **ZoneDBImport** utility loads data in the IFW_STANDARD_ZONE table of the Pipeline Manager database.

This utility uses the following files:

- **Control File (zoneLoader.ctl**)

  The **zoneLoader.ctl** file controls how the data is loaded. It contains information about the table name, column datatypes, field delimiters, and so on.

  Initialize the infile variable with the path and file name of the file that contains the data to be imported.

- **Execution File (zoneLoader.pl)**

  Update the entries for the **DatabaseName** and **UserName** with the database name and user name of the current database.

Location

Pipeline_Home/**tools**

Syntax

```
./zoneLoader.pl
```