

amdocsrating

# Rating 6.0

## Specification (Internal)

**For Internal Use Only**



© 2006 Amdocs

This document contains proprietary and confidential information of Amdocs and shall not be reproduced or transferred to other documents, disclosed to others, or used for any purpose other than that for which it is furnished, without the prior written consent of Amdocs. It shall be returned to the respective Amdocs companies upon request.

The trademark and service marks of Amdocs, including the Amdocs mark and logo, are the exclusive property of Amdocs, and may not be used without permission. All other marks mentioned in this material are the property of their respective owners.

## Document Information

Software Version:	<b>6.0</b>
Publication Date:	<b>January 2005; updated November 2006 for SP8</b>
Catalog Number:	<b>212370</b>

# Contents

<b>Contents .....</b>	<b>iii</b>
<b>Rating 6.0 Usage Rating Specification (Internal) .....</b>	<b>1</b>
<b>1. Introduction .....</b>	<b>3</b>
Purpose and Scope.....	3
Main Functions.....	3
Main Modules .....	4
Interactions with Other Components .....	4
Benefits of Amdocs Rating .....	4
Terminology .....	5
<b>2. Event Processing Overview .....</b>	<b>7</b>
Event Processing Flow.....	7
<b>3. Rating an Event: Functionality Overview .....</b>	<b>11</b>
Events .....	11
Guiding to Service: Qualification and Priorities .....	11
Rating an Event.....	14
Performance Indicators .....	18
Dispatcher .....	20
<b>4. Pricing Engine .....</b>	<b>21</b>
Inputs .....	21
Pricing Engine Flow.....	22
<b>5. Rating Envelopes .....</b>	<b>29</b>
Postpaid Rater (Offline Charging) .....	29
Rerating.....	30
Online Charging .....	31
<b>6. Rating Outputs and Usage Data Storage .....</b>	<b>33</b>
Postpaid Rating Output .....	33
Data Storage Types .....	33
Usage Flow from Rating to Oracle .....	34
Replicator Mechanism.....	35
Dispatcher Mechanism.....	35
Performance Indicator Maintenance .....	35
<b>7. Rating Inputs .....</b>	<b>37</b>
Rating Data Storage.....	37

## Rating 6.0 Specification (Internal)

---

Customer Information Input.....	37
Product Catalog Input .....	39
<b>8. Rating Output Interfaces .....</b>	<b>41</b>
Rating Extract.....	41
Online Usage Queries.....	42
Dispatcher .....	42

## Rating 6.0 Rerating Specification (Internal) ..... 43

<b>1. Introduction.....</b>	<b>45</b>
Purpose and Scope.....	45
Rerating Overview.....	45
<b>2. Rerating Scenarios .....</b>	<b>49</b>
Customer Management – Retroactive Activities .....	49
Customer Management – Allowance Activities .....	49
Rating – Chronologically Inaccurate Events .....	49
Rating – Inaccurate Product Catalog Entries .....	50
Billing – Undo Bill Preparation.....	50
PI Reconstruct – Inaccurate Accumulation .....	50
<b>3. Identifying Subscribers for Rerating.....</b>	<b>51</b>
Overview .....	51
Customer Management – Marking Subscribers.....	53
Rating – Marking Out-of-Sequence Events.....	55
Manual Marking of Subscribers.....	58
Agreement-level Handling and Marking of Subscribers .....	58
<b>4. Determining Event Rerating Requirements.....</b>	<b>59</b>
Receiving Subscriber Activity Events.....	59
Handling Subscriber Activity Events .....	59
Implementation Repository Definitions.....	60
Handling Subscriber Events.....	62
<b>5. Processing Subscribers for Rerating .....</b>	<b>63</b>
Rerating Events After Bill Cycle is Locked .....	63
Rerating Events While Cycle Is Open .....	64
Recommended Solution .....	64
<b>6. Extracting Data for Reguiding and Rerating .....</b>	<b>65</b>
Subscriber Pre-rating Job .....	65
Subscriber Extract Job .....	65
Background Information .....	66
Previous Bill Cycle Rerate Extract .....	66
Cycle Rerate Extract .....	67

<b>7. Reguiding and Rerating Processes .....</b>	<b>69</b>
Reguiding .....	69
Preparing Files for Rerating .....	69
Rerating Flow .....	69
Rerating Completion.....	70
<b>8. Rerating Due to Billing Request.....</b>	<b>71</b>
Billing – Preparation of Rerate Population File.....	71
Rerating in Customer Mode .....	71
<b>Rating 6.0 Maintenance Specification (Internal).....</b>	<b>73</b>
<b>1. Introduction .....</b>	<b>75</b>
Purpose and Scope.....	75
Main Functions .....	75
<b>2. Cycle Control .....</b>	<b>77</b>
Cycle Tables .....	77
Cycle Life Cycle.....	79
Cycle Operations.....	81
<b>3. Performance Indicator Maintenance Process .....</b>	<b>83</b>
General .....	83
Reconstructing Performance Indicators .....	84
PI Maintenance Process Description .....	84
<b>4. Truncate Cycle Process.....</b>	<b>87</b>
Usage Partitioning Background.....	87
Number of History Cycles To Keep .....	87
Truncate Cycle Process Flow.....	87
<b>Rating 6.0 Recovery Specification (Internal) .....</b>	<b>89</b>
<b>1. Introduction .....</b>	<b>91</b>
Recovery Overview .....	91
Purpose and Scope.....	91
Interaction with Other Components.....	91
Process Flow.....	92
<b>2. Rating Flow .....</b>	<b>93</b>
<b>3. Preparation for Recovery.....</b>	<b>95</b>
Recovery Methodology.....	95
Rating Flow – Preparing the Data for Recovery.....	95
<b>4. Recovery Process .....</b>	<b>97</b>
Recovery Flow.....	97

<b>5. Postpaid Recovery Table .....</b>	<b>99</b>
---	-----------

<b>Rating 6.0 Usage Queries Specification (Internal).....</b>	<b>101</b>
---	------------

<b>1. Introduction.....</b>	<b>103</b>
Overview .....	103
Architectural Design Standards.....	104
Purpose and Scope.....	104
<b>2. Usage Query Description.....</b>	<b>105</b>
Query Flow.....	105
Query Definition .....	106
Technical Overview.....	109
Implementation Repository Definition .....	109
API Definition .....	113
Billing Cycle API.....	114
Amdocs Front End Client .....	115
<b>3. Usage Query Architecture.....</b>	<b>121</b>
EJB and Rating API Connectivity.....	121
Routing Requests to Responsible Rating Data Store .....	122
Database Connectivity .....	123
<b>4. Implementation Examples.....</b>	<b>125</b>
Event List .....	126
PI Details for Rate-Based PIs .....	128

<b>Appendixes .....</b>	<b>131</b>
-------------------------	------------

<b>Appendix A. Rerating Data Storage Table .....</b>	<b>133</b>
Subscriber Rerate Data Storage .....	133
Mark Subscriber for Rerate .....	133

<b>Appendix B. Rolling Allowance Handling .....</b>	<b>135</b>
Rolling Allowances and Rerating .....	135
Rolling Allowances – Scenario 1 .....	135
Rolling Allowances – Scenario 2.....	135
Recommendation .....	136

<b>Appendix C. Mark Next Cycle for Rerate.....</b>	<b>137</b>
Allocation of Rolling Allowances .....	137
Recommendation .....	137

<b>Appendix D. Rerating Map.....</b>	<b>139</b>
--------------------------------------	------------

<b>Index .....</b>	<b>141</b>
--------------------	------------

amdocsrating

# **Rating 6.0 Usage Rating Specification (Internal)**





# 1. INTRODUCTION

---

Communications service providers offer various services to their customers, and numerous pricing structures to support a wide range of marketing strategies.

The services offered might include a variety of products in different lines of business, including:

- Mobile networks
- Long-distance calls
- Private network (physical and virtual) services
- Data traffic, such as GPRS or UMTS sessions
- Internet-based services, such as:
  - Information
  - Email
  - Interactive games
  - Other entertainment services

Each usage of such services generates an “event.” The Amdocs Rating component processes and assigns prices to such events.

## Purpose and Scope

This part of this book provides a functional and technical description of Amdocs Rating. It includes a general overview of Rating and explains the principles involved in its design.

## Main Functions

Rating’s main functions are:

- Receive various types of events
- Identify rating plans, allowances, discounts, and additional charges, and apply these to events
- Maintain accumulations of event attributes
- Dispatch rated events to various destinations
- Record the results of the transactions

In addition, Rating may perform the following:

- Rerate previously rated events
- Rate recurring and one-time charges
- Expose rating information (rated events and performance indicators) to downstream Amdocs components and external systems

## Main Modules

Amdocs Rating comprises these main modules:

### Pricing Engine

The Pricing Engine is implemented as a set of APIs which combines all functions required for rating an event. Reading relevant customer information and performance indicators (PIs) and referring to the Product Catalog (containing all dynamic definitions and business logic), it rates the event, updates PIs, and outputs the rated event. The Pricing Engine is described in detail in chapter 4.

### Envelopes

The Pricing Engine is used in various processes, and these vary in the context in which they receive events and in which they handle the output after rating the events.

The following envelope processes perform various Rating functions:

- Postpaid Rater (Offline Rater) – Processes events submitted by files from various network mediation devices or other file inputs.
- Online Charging – Processes events submitted in real time for prepaid authorization and debiting, advice of charge requests, and similar online interactions between the subscriber –and Rating.
- Rerating – Reprocesses events that were previously rated, to resolve various business scenarios requiring rerating before bill production.

Envelopes are described in detail in chapter 5.

## Interactions with Other Components

Rating interfaces with the following components in the event processing flow:

- Acquisition & Formatting
- Customer Management
- Product Catalog
- Billing

## Benefits of Amdocs Rating

Amdocs Rating offers these benefits to communications service providers (CSPs):

- Rapid introduction of new event types, without changing or reinstalling the core system
- Dynamic definition and extension of data accumulators (performance indicators) and their behavior
- Rapid introduction of new rating logic to react to fast-changing business requirements and customer needs
- Dynamic rate configuration, allowing rates to depend on any number of event, customer, or performance indicator attributes

- Flexible guiding to service, using any of the event or customer attributes
- Enhanced performance for real-time throughput
- Stand-alone components and clear interfaces with other applications decrease dependencies on other applications
- Deployable in both Amdocs and third-party environments

## Terminology

The following special terms and acronyms are used in this book.

Term	Definition
BE	Business entity
BOH	Business organization hierarchy
Cross-cycle PI	See <i>Rolling Allowance</i> .
CSP	Communications service provider
DWH	Data warehouse
EDR	Event detail record
EJB	Enterprise Java Bean
Event	Usage of telecommunications network services
IMDB	In-memory database
OC	One-time charge
PI	Performance Indicator. Contains information on all accumulated rated events for a subscriber or group of subscribers. Any attribute and amount of the event can be accumulated. Usually, the chargeable attributes, as well as the event rates, are accumulated. PIs are used for retaining data associated with allowances, discounts, and budget control schemes.
PIT	Pricing item type
QA	Quality Assurance
RC	Recurring charge
Replicator	A daemon process that reads rated events from the TimesTen log and writes them into files. Then records in the file are inserted into the database.  TimesTen maintains logs and a database image on disk, to help recover from a loss of data due to hardware failure. The Replicator mechanism uses this log to record events inserted into database.
Rolling allowance	A special type of <i>Performance Indicator</i> , which can maintain an accumulation over multiple cycle months. At the end of a given cycle month, the rolling allowance is transferred to the next cycle with its value intact. Also known as a cross-cycle allowance.
SIT	Service item type

## Rating 6.0 Usage Rating Specification (Internal)

---

Term	Definition
TimesTen	A high-performance relational IMDB that supports the ODBC (Open Database Connectivity) and JDBC (Java Database Connectivity) interfaces. This database is supported on multiple platforms. It allows transaction management, full or partial replication, and also makes its logged data accessible through an open API.
TRB	Transaction Broker. The customer care and billing system of a CSP contains several functional components. Each component is responsible for a defined set of functions. Each component is autonomous and self-contained. The Amdocs Transaction Broker is responsible for exchanging information between these separate components.
Usage	Rated events and performance indicators

## 2. EVENT PROCESSING OVERVIEW

---

“Event processing” refers to all the functions related to receiving events from various sources (such as GSM switches, UMTS and GPRS networks, fixed-network switches, IP application servers, and roaming clearinghouses), formatting them, and rating them.

Event processing comprises two processing chains:

- Online Charging – For real-time processing
- Offline Charging – For non-real-time processing.

This part of this book deals primarily with offline charging. (For a brief description of Online Charging, see the “Rating Envelopes” chapter.)

Offline event processing comprises two components:

- Acquisition & Formatting – Integrates all the functions involved in receiving events from mediation devices and preparing them for processing by Amdocs Rating.
- Rating – Associates events with their services, assigns prices to them, and applies allowances, discounts, and other price adjustments.

### Event Processing Flow

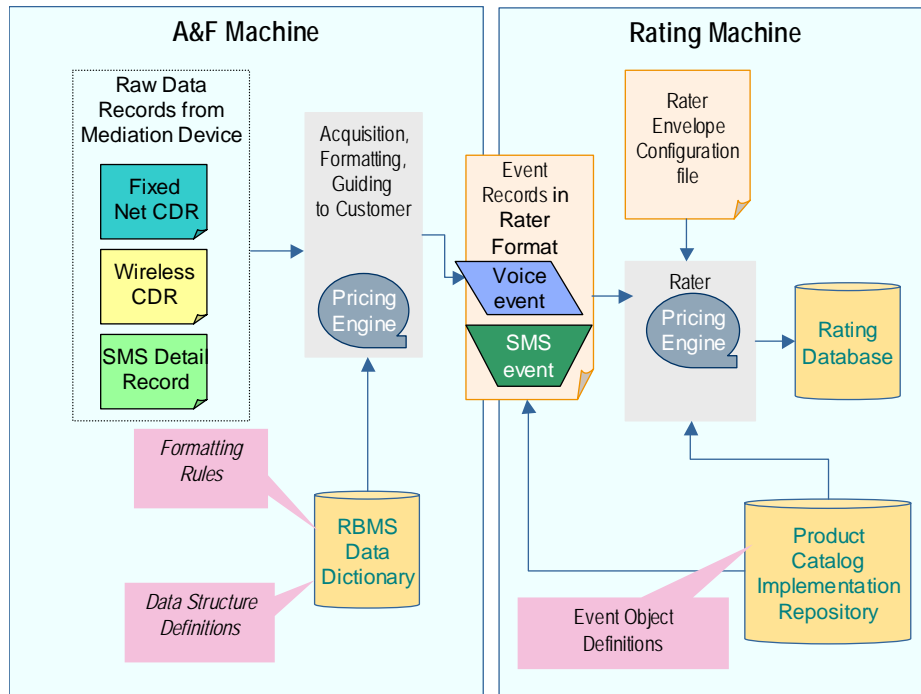
Events are received, processed, and rated according to the sequence of tasks described below.

#### Acquisition & Formatting Flow

The Acquisition & Formatting component performs these actions:

1. Receives raw input events from various sources and identifies their event types.
2. Identifies the customers who own the events.
3. Distributes the events to the correct pay channels.
4. Maps the event records into standard formats for their event types, as input for the Rating process.
5. Forwards the events to Rating.

The diagram below illustrates the inputs to Acquisition & Formatting (“A&F machine”), and the flow to Rating (“Rater machine”) in detail.



**Figure 2-1: Acquisition & Formatting to Rating Flow**

Acquisition & Formatting edits and validates each event and assigns its proper type. The event type is used by Rating as one of the main filters in determining how to rate the event.

Rating is integrated with Acquisition & Formatting by sharing Product Catalog data layout definitions. After checking and classifying events, Acquisition & Formatting converts them into the format that Rating uses, using the Pricing Engine API.

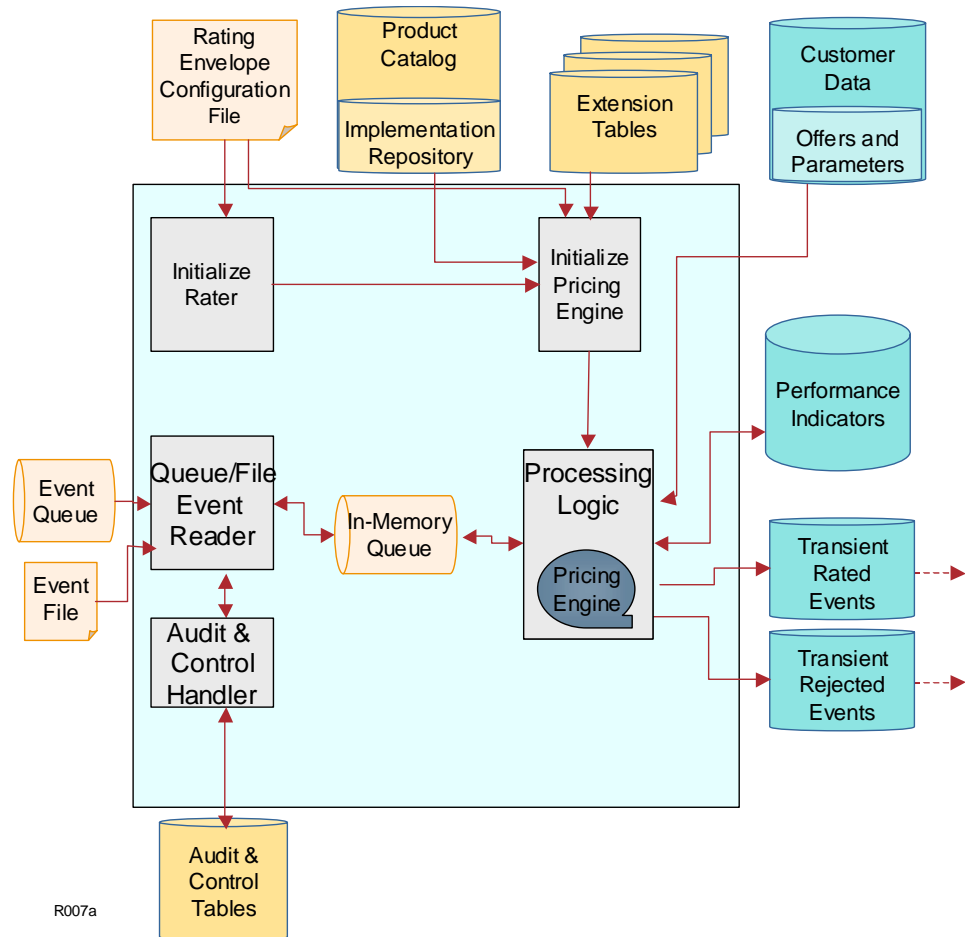
The Acquisition & Formatting component includes the Guiding to Customer function. Thus, events include both event and customer identification attributes when they reach Rating.

## Rating Flow

Rating performs these actions on events received from A&F:

1. Applies the correct packages to be used for rating the events.
2. Identifies and allocates allowance packages (free units) to the events.
3. Applies the correct rates to the events.
4. Applies discounts to the events.
5. Applies additional charges to the events.
6. Applies the events' contributions to the performance indicators (event accumulators).
7. Evaluates the effects of the events on budget control limits.

8. Dispatches the rated events to the destinations that were specified for them.
9. Saves the results of the process into a database.



The following diagram illustrates the major elements of the Rating transaction flow:

**Figure 2-2: Rating Transaction Flow**

After the Rater and Pricing Engine are initialized, the Data Event Reader (part of the Postpaid Rater envelope) retrieves the event from the input medium. The event is transferred through an internal in-memory queue to the Pricing Engine. The Pricing Engine selects the pricing items for rating, rates the event, updates the performance indicators affected by the event, and updates the output media with the results.





### 3. **RATING AN EVENT: FUNCTIONALITY OVERVIEW**

---

This chapter describes the Pricing Engine functionality for rating an event. (The “Pricing Engine” chapter describes the Pricing Engine technical flow.)

Rating is required to:

- Receive various types of events
- Guide the events to customers’ or market services
- Apply rating plans, allowances, discounts, and additional charges to the events
- Maintain counters (performance indicators) of event attributes
- Dispatch events to various destination systems

These functions are described in detail below.

#### **Events**

An event can be defined as any use of a service on the CSP’s network by a subscriber. An event can result from such activities as a voice call, data transfer across a wireless network, a short text message (SMS), an online charging authorization request, etc.

Each such event may have different properties and may require different processing and rating logic. Events are therefore categorized into different event types. Events of the same type share the same data structure; that is, all events of the same event type have the same attributes. The event type plays a significant role in mapping the event to its rating model and to a specific rate.

Among the basic functions provided by Rating are:

- Support for a vast number of different event types
- A mechanism for rapidly introducing new event types in the Product Catalog without changing the core system

#### **Guiding to Service: Qualification and Priorities**

A subscriber acquires one or more offers, either directly or through its business entity (BE) policy or customer hierarchy. An offer can contain multiple pricing packages; pricing packages can contain multiple pricing items. For example, a pricing package could contain price definitions for regular voice calls, “Friends and Family” voice calls, and commerce transactions, together with a free-minutes allowance for voice calls that depends on the number of commerce transactions performed.

All this means that Rating must perform a qualification check to determine which rating scheme should be used for each event. Rating must identify the packages and specific pricing items that contribute to the processing and

rating of the event from all the offers the subscriber acquired. This is done by the Pricing Engine's Guiding to Service function.

The qualification check is done using the event type, event service filter, and a qualification criterion (built from expressions evaluating functions and attributes, combined through logical operators) which is attached to the item type or to the package.

For example, the qualification criteria for the mobile-to-mobile call pricing item might be a simple check that the event type is a "Voice call," and that the service filter is "mobile to mobile." The qualification criteria for the Friends and Family pricing item might add to these expressions a check that the called number (recorded in the event record) is on the subscriber's friends and family list (which resides in the customer data).

## **Qualification by Event Type and Service Filter**

Acquisition & Formatting, which includes the Guiding to Customer function, maps events to subscribers as well as to event types. This means that the event record that Rating receives from Acquisition & Formatting already contains customer identification attributes and a specific event type relevant for the transaction (voice, SMS, data, etc.).

Acquisition & Formatting also computes the service filter, which is the basic qualifying attribute used to match event types to rate items. The computation of the service filter is based on fields received in the event record. For example, a Voice event may have such service filters as mobile to mobile, mobile to fixed, and international.

Rating's first qualification checks are to match the event's type and service filter to the pricing item event type and service filter.

Event types that are parents of the events' type are also qualified. For example, for qualifying voice events, if the Duration event type is a parent of the Voice event type (that is, Voice inherits from Duration), then pricing items that are related to the Duration event type are qualified as well.

A pricing item with an empty service filter is qualified for all events.

The following table illustrates the qualification results for the case of a customer who has a Voice event with the MTM (Mobile to Mobile) service filter.

<b>Pricing Item</b>	<b>Event Type</b>	<b>Service Filter</b>	<b>Qualified</b>	<b>Explanation</b>
Standard rate for mobile	Voice	MTM	Yes	Qualified because the event and service filter match
Standard rate for fix	Voice	FIX	No	Not qualified because service filter does not match
Standard rate for data	Data traffic		No	Not qualified because event type does not match

Pricing Item	Event Type	Service Filter	Qualified	Explanation
Allowance	Duration event (voice is inherited from the duration event)		Yes	Qualified because event type is a parent of the Voice event type and service filter is empty

## Qualification Criterion

In addition to the event type and service filter qualification checks, a qualification criterion can be added to the pricing item type or pricing package. This qualification criterion supports a flexible Guiding to Service mechanism, which can use any attribute known to Rating to guide the event to a service.

A qualification criterion is a Boolean expression used to associate an event with an item (for rating, discounting, etc.). The qualification criterion is evaluated at run time to determine whether the item qualifies the event for any purpose.

There are two types of qualification criteria – global and private. A global qualification criterion can use any attribute from the customer, event, or a reference entity. A private qualification criterion can use the item parameters entity as well. The criterion is built by forming expressions evaluating functions and attributes, and combining these expressions with logical operators.

For example, a pricing item is available only for calls to two designated friends' numbers while in a specified geographical zone. The qualification criterion checks whether:

10. The called number (the “B number”) is on the customer’s list of two “friends,” and
11. The subscriber was in the specified geographical zone while making the call (the origination cell in the event’s item parameters is one of the cells specified for the item).

## Priority in Selecting and Executing Services

Pricing items include a priority mechanism to assist Rating in selecting the preferred item or defining the order of execution for other types of items. The qualification priority of an item is defined in the Product Catalog in the following ways:

- A priority number on the package
- The order of the items within the package
- The offer expiration date (offers expiring sooner get higher priority)

The execution priority is defined by the order of qualification cases within a pricing item type.

For more details regarding the way the Pricing Engine defines the order of execution of pricing items, see Chapter 4, “Pricing Engine.”

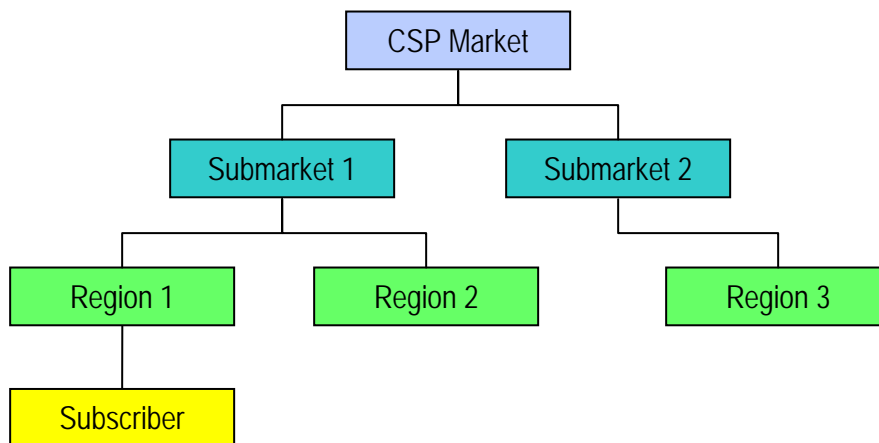
## Business Organization Hierarchy and Package Priority

The CSP's business organization hierarchy (BOH) can have various levels of complexity.

A business-entity-level (market-level) offer can be attached to each node in the hierarchy. For each event, the Pricing Engine retrieves all market-level offers that are attached to the subscriber's business entity (BE) and to BEs at higher levels in the BOH tree.

The Pricing Engine decides on the selection and order of the items according to the priorities assigned to the pricing packages included in the market and submarket offers.

The diagram below illustrates an example of a BOH:



**Figure 3-1: Business Organization Hierarchy**

In this example, because the subscriber belongs to Region 1, the Pricing Engine retrieves all market-level offers that are attached to the Region 1, Submarket 1, and CSP Market nodes.

In the example, all the offers include the International Voice pricing item, which is qualified for the event.

The pricing packages are assigned these priorities:

- Region 1: priority 10
- Submarket 1: priority 5
- CSP market: priority 20

The Submarket 1 pricing package has higher priority than the other two offers, so the Submarket 1 pricing item is selected for rating.

## Rating an Event

Rating an event can be a simple computation where the rate is straightforward – for example, short text messages that are rated at a flat charge of 10 cents each. Generally, however, event rating is more complex. Event rating is based on a rate table associated with the pricing item. The rate tables offer sufficient

flexibility to perform multiple computations in calculating the rate for an event.

## Voice Event Rating Examples

The following examples of voice event rating illustrate the flexibility of Amdocs Rating, where the attributes that affect rating may be any combination of reference data, event data, customer data, and accumulators.



*These examples are only a few of the endless possible implementations for pricing Item types.*

### Rate by Duration Only

A voice event with a flat price per duration unit requires a very simple rate table. In this example, a flat rate of 30 cents is applied to each time unit (for example, a minute). This rate table has the following format:

Quantity	Rate
1	0.30

### Rate by Period and Duration

A voice event can be priced differently in different periods of the day. In the example rate table below, a lower rate applies to each duration unit during off-peak hours.

Period	Quantity	Rate
Peak	1	0.30
Off Peak	1	0.20

The event must be divided into peak and off-peak segments to compute the rate. Additional reference data are required to define the peak period of the day.

### Rate by Accumulated Usage, Period, and Duration

A voice event also can be priced according to aggregated accumulated voice usage – for example, with reduced rates for higher calling usage volumes during the billing cycle. In this case, the rate table has this format:

Period	Duration Step (Minutes)	Rate
Peak	0 – 100	0.35
Peak	101 – 200	0.30
Peak	201 +	0.28
Off-Peak	0 – 100	0.25
Off-Peak	101 – 200	0.20
Off Peak	201 +	0.17

In this case, the event must be divided into peak and off-peak segments, and accumulators of voice call usage throughout the billing cycle must be accessed and updated, to compute the rate.

## Flexible Rating Schemes

For maximum flexibility to quickly introduce new services to the market, the Pricing Engine does not include all the rating logic. Instead, this logic is defined in the Product Catalog's Implementation Repository.

The pricing item type (PIT) defines a template for the functionality of a single, independent pricing element. It is a technical abstraction allowing the reuse of a definition in many independently defined pricing packages by specifying the item parameters. The item type includes definitions for qualification and handling of events. Each item type can handle one or multiple related events.

Roles are used for classifying item types. The qualification policy is defined according to the role. The qualification policy includes:

- The minimum and maximum values of items qualifying the event for execution can be defined per role policy. In general, only one item of a rate role is selected, while an unlimited number of items from other roles can be selected for execution (allowance, discounts, additional charges, etc.).
- Order of execution.

The following roles are relevant for rating of usage events by their order of execution (the current core implementation also includes billing roles, and more roles may be added):

- Rate – Apply the correct rate to the event
- Allowance – Allocate allowance packages to the event
- Discount – Apply discounts to the event
- Additional charge – Apply additional charges to the event
- Budget control – Evaluate the effect of the event on consumption control limits
- Benefit – Evaluate the effect of the event on benefits granted to the customer

The following subsections describe each role in detail.

## Rate Role

Several steps may be implemented to prepare all the data required for accessing the rate table. These steps may include:

- Rounding of chargeable attributes (duration, volume, etc.).
- Determining step ranges and splitting the event into several segments.
- Applying allowances prior to determining the final rate from the rate PIT, in order to exploit available free units.
- Accessing performance indicators accumulated over time, which influence the key to the rate table.
- Processing any functions that are required to execute the logic of the rating model.
- Applying discounts (the discount role is called from the rate role).

Once the rate table has been accessed, the rate is determined. Different rate tables may be accessed for the same event, and the final rate may be the sum of several rates, the minimum of each rate, the maximum of each rate, or any other formula required by the CSP.

## Allowance Role

Allowances are packages that provide free units, either specific or general units:

- Specific units include such allowances as free minutes for voice calls, free short text messages, or free data transfer volume units.
- General units are converted to specific units by means of a translation table. This allows use of the same allowance package for different event types with different chargeable units. For example, an allowance package might provide 100 units, where each unit is equivalent to one voice minute, 0.5 KB of data traffic, or three short text messages. The consumption and replenishment strategy is defined in the item parameters, and accumulators are required to record the consumption of the allowance units.

The consumption and replenishment strategy specifies the way units are consumed and replenished. The following elements can be specified:

- Unit accumulation – either:
  - Use it or lose it – The free units are granted for the specified period. Any units remaining at the end of the period are lost.
  - Roll over – Any units not consumed during the period are rolled over to the next period.
- Unit replenishment – either:
  - One time – The package specifies a fixed amount of units with no replenishment.
  - Recurring – the specified number of units is replenished every period.
- Any other element required by the CSP's business scenarios.

## Discount Role

Once the base rate has been calculated, discounts may be applied to the rate if the event qualifies for it. Discounts may be expressed as a percentage reduction of the rate, or as an absolute value. Discounts are generally applied when a certain usage threshold has been exceeded – for example, a discount of 10% on all voice calls if the total commerce expenditure exceeds 200 euros. To provide this functionality, an accumulator must be provided to record the total commerce expenditure. Qualification criteria to be checked prior to applying the discount are that the event type is a mobile call and that the commerce expenditure has exceeded the threshold amount.

## Additional Charges Role

Various business cases may require additional charges beyond the main rate charge. Examples include an additional charge for special numbers, and tolls that some CSPs charge in addition to air time charges.

The rating schemes of PITs for additional charges are similar to the rate PIT rating schemes.

### Budget Control Role

The Budget Control function monitors the amount that the customer spends during a billing cycle. These control functions may be initiated by:

- The CSP, through credit limits to control fraud and unwarranted usage, or
- The customer (this is often referred to as a credit watch)

The credit limit is checked after the event is rated and discounted. If the credit limit has been reached, Rating triggers a customer notification by writing an output notification event (the specific response depends on the implementation).

### Benefit Role

Once the base rate has been calculated, benefits may be granted to the customer based on performance thresholds. For example, Rating can notify the subscriber, when a predefined threshold is reached, that the subscriber is entitled to a handset upgrade.

## Performance Indicators

A performance indicator (PI) contains information on all accumulated rated events for a subscriber or group of subscribers. Any attribute and amount of the event can be accumulated; usually the chargeable attributes as well as the event rates are accumulated.

In addition, PIs are used for maintaining allowances, discounts, and budget control schemes.

The performance indicators are specified by the administrator when defining the pricing item type (PIT) in the Product Catalog. The PI structure depends on the computation model the PIT and PI are required to support.

### Objectives of Performance Indicators

Performance indicators fulfill the following roles:

- Provide totals for each event type, ready for Billing. On the bill day, the PI extract creates charges from the PIs.
- Enable up-front rating of every event as it arrives, without accessing previously rated events stored in the Rated Event table; also provide an accumulator for any rating model using steps.
- Apply allowances to the event, based on monitoring a PI. The allowance PI holds the replenishment quota and remaining quota per pricing item.
- Store reservations for Online Charging reserve requests (such as allowance reservations).
- Provide input to discount schemes based on the PIs. The discounting model might depend on PIs and thus implement rating discounts. An example is the application of a 10% discount for every voice call above a



defined threshold, if the total duration reaches the threshold. These discounts are applied during the rating process itself.

- Grant benefits based on monitoring of a PI. Benefits can be granted to a customer as a result of meeting a PI threshold. External systems may be used to execute the benefit, such as sending a letter or an SMS notifying the customer of eligibility for a handset upgrade.
- Provide budget control functions based on monitoring of a PI. In the simplest case, budget control is associated with the total performance of the subscriber. In this case, a PI is defined which stores a sum of all the charges, independent of the event types. When the threshold is reached, an alert or barring transaction is triggered.

## Performance Indicator Structure

Performance indicators are associated with subscribers or agreements. The structure of PIs can be divided into two sections, fixed and dynamic, which comprise different sets of data.

### Fixed Structure

The fixed structure contains base fields common to all PIs. These fields allow for easy database access to perform the rating in real time. The fields are:

- Pricing Item ID – The unique identifier of the pricing item.
- Subscriber or Agreement ID – The record can be created either per subscriber or per agreement owner, depending on the role of its pricing item and the way it is assigned to a customer.

Rate, discount, and additional charge role PIs are always at the subscriber level (although the offer may be at the group level). All other roles (allowance, budget control, etc.) are at the agreement level for group-level offers, and at the subscriber level for subscriber-level offers. When the PI is at the agreement level, it accumulates attributes at the level of an agreement associated with a node in the customer hierarchy.

- Cycle attributes – Cycle code, cycle month, and cycle year.
- Customer ID – The unique customer identifier.
- Offer instance – Unique identifier of an offer instance, allowing the subscriber to buy the same offer more than once. The offer instance distinguishes between the PIs of the same pricing item type.

### Dynamic Structure

The PI record includes dynamic data, with content that depends on the role of the accumulator and the specific functionality of the pricing item type with which it is associated. The PI can include counters maintained separately per period, per step, or per charge code.

Performance indicators are defined in the Product Catalog and associated with a particular pricing item type. When defining the PI, the administrator stipulates the attributes required to be stored for the PI, and their breakdown. These additional attributes include:

- Breakdown dimension attributes, which allow the slicing of the PI into different accumulators when any of these attribute values change. These breakdowns usually are made for billing purposes. Examples include:
  - Break down the PI by account and pay channel so that the charge created from the PI dimension can be distributed directly to the payer's pay channel.
  - Break the PI by tax change date so that each portion of the PI is taxed using relevant rates.
- Standard attributes, which include information required by a downstream system or for the pricing item type rating scheme. Examples include last event date, number of events, or complex attributes like number of events per period and charge amount per period.

### Global and Partitioned PIs

Rating supports the Product Catalog's global flag for PI attributes, so the Pricing Engine can distinguish between "global" and "partitioned" PI attributes. "Global" attributes are attributes that exist once in a partitioned PI, whereas "partitioned" attributes have a different instance in every partition. ("Partitioned" is the default value.)

## Performance Indicator Classification

There are two categories of performance indicators:

- Self-contained – These PIs and their accumulators have no effect on future cycle periods. For example, total volume and charges for a data session are measured in the context of a cycle period, and the information is not needed or rolled over to the next cycle.

A new self-contained PI is created by the Pricing Engine for each cycle.

- Cross-cycle – These PIs and their accumulators affect future cycle periods. For example, an allowance PI whose balance is rolled over to the next cycle period is defined as "cross-cycle."

A new cross-cycle PI is created for each new cycle with the information from the previous cycle.

## Dispatcher

The Dispatcher mechanism distributes an event to different destinations immediately after it has been rated.

Following are examples for the use of the Dispatcher mechanism:

- Send the rated event to an external system, which may be the customer's service provider, a data warehouse, or an international carrier.
- Rate an event provided by a service or content provider according to the rate plan of the subscriber using the service, and then using this rate, send the event to the service or content provider's account for revenue sharing based on a percentage of the rate applied to the subscriber.
- Send all rated events to the Rated Events table.

## 4. PRICING ENGINE

---

The Pricing Engine is the brain of Rating. It performs all the business logic processing for Rating, as well as other price-related processing such as the Online Charging server's authorization function and more. The Pricing Engine supports very flexible price definitions and many different options, because most of the business logic is defined for each CSP's specific implementation, rather than embedded in the Rating core code.

### Inputs

The Pricing Engine uses the following three types of input:

- Configuration definitions
- Product Catalog data
- Operational data

Each of these inputs is described in the subsections that follow.

### Configuration Definitions

Configuration definitions are found in two sources:

- Configuration file – This XML file contains configuration settings for the Pricing Engine, such as a list of extension libraries to load, reference database connection information, versioning parameters, and so on. The engine loads the settings once at startup.
- Extension libraries – These are sharable libraries implementing pricing functions used in the pricing computation flow. The engine does not link with these libraries, but rather loads them dynamically. Such extension functions are introduced to the Product Catalog's Implementation Repository by defining their names and parameters. Once extension functions are introduced to the Implementation Repository, they can be used in the pricing flow specified through the Implementation Repository GUI.

### Product Catalog Data

The Pricing Engine receives inputs from these parts of the Product Catalog:

- Implementation Repository
- Product Catalog information (offers, packages, pricing items)
- Auxiliary Repository

## Operational Data

The Pricing Engine receives input from these Operational sources:

- Event object – The event includes information from the network and customer information that is added by the Guiding to Customer function of Acquisition & Formatting. The Pricing Engine receives this object as an input parameter through its API. The API provides two options:
  - The first expects the data of this object to be passed as a continuous memory buffer (read either from a file or from a message queue).
  - The second gets the data as an entity object, instantiated and populated (regularly field by field) by the caller.
- Customer object – The Pricing Engine loads this object from the Rating customer tables in the first stages of processing an event. In addition, the Pricing Engine uses customer and subscriber identification attributes on the event that are populated by the Guiding to Customer function.
- Performance indicators – The Pricing Engine loads the PIs after guiding the event to the pricing item and before executing the pricing item's computation logic (which usually refers to the PI). Since a PI is maintained per cycle, the Pricing Engine creates a new record in each cycle. The Pricing Engine retrieves the relevant entry from the table according to the static primary key fields and according to the dynamic dimensions (defined in the Product Catalog).

Before loading a performance indicator, the Pricing Engine checks whether the PI is accessed by any handler. If there is no reference to it in any handler, the Pricing Engine does not retrieve, create, or update it. This preserves resources and improves performance.

When a PI is not found, a new record or internal dimension is created and the initialization handlers of the PIT are invoked to initialize the record. Two options exist:

- If the PI is defined as cross-cycle, the Pricing Engine retrieves the record from the previous cycle, and this record is passed to the initialization handlers of the PIT.
- If the PI is not cross-cycle, a new empty record is passed to the initialization handlers of the PIT.

## Pricing Engine Flow

The Pricing Engine can be in one of two states:

- Initialization
- Event processing

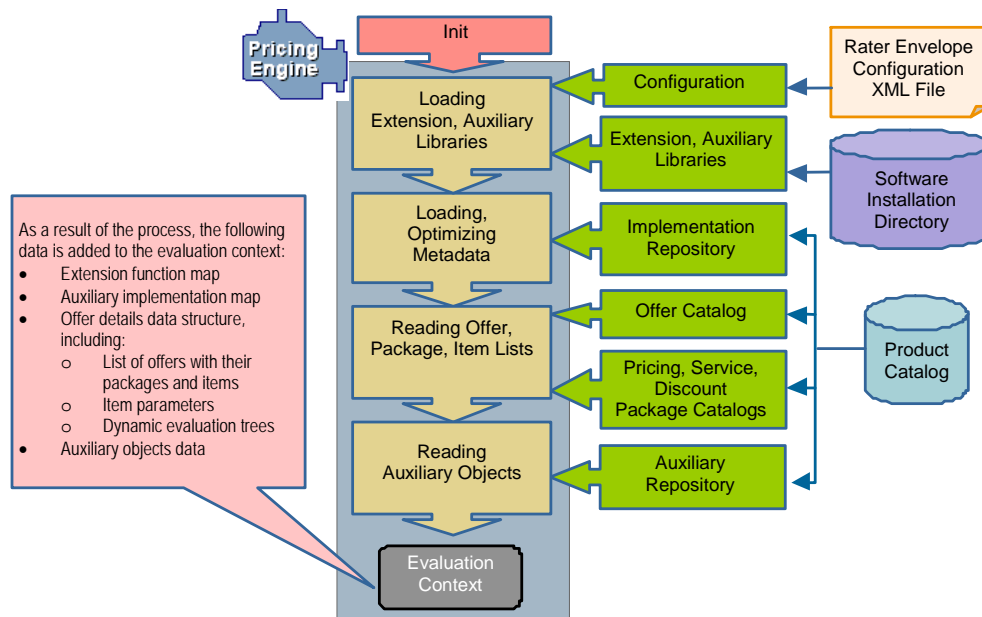
The following subsections describe these states in detail.

### Initialization

The Initialization process loads all reference data to internal in-memory objects and creates efficient references among them. The Pricing Engine

generally tries to do as much preparation as possible during initialization, to minimize the time needed for event processing.

The Initialization process flow is illustrated below:



**Figure 4-1: Pricing Engine Initialization**

The Initialization process is performed when the Pricing Engine starts, and when re-initialization is requested as a result of changes in reference data (such as auxiliary objects, Product Catalog XML files, and the like) and the Pricing Engine has to reload the reference data into memory.

The following steps are performed in the Initialization process:

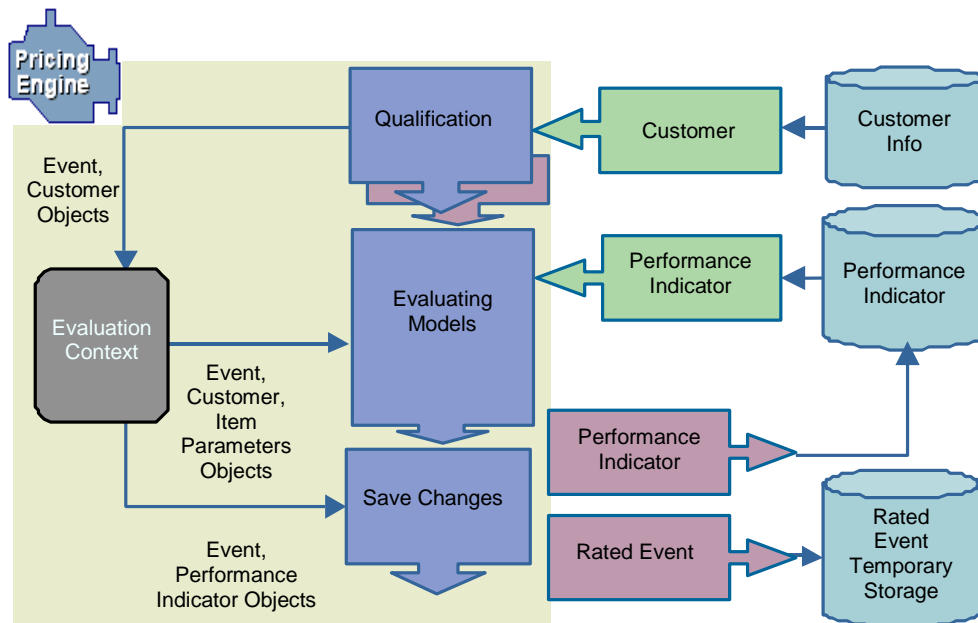
12. Load configuration.
13. Load extension libraries.
14. Load Implementation Repository.
 

The Pricing Engine run mode (on-line or off-line) is specified in the configuration file. In addition, the run mode can be specified for each handler. The PE loads only those handlers that match the run mode specified in the configuration and those that should always be activated.
15. Load Product Catalog items (offers, packages, pricing items).
16. Invoke the Init method for all extension functions defined in the Implementation Repository. These methods are responsible for loading and optimizing any data they require for event processing.
17. Initialize auxiliary objects, which may be loaded to memory at this stage.

## Event Handling

This process is performed once for each event. As a result, either the total charge for the event is determined, or another type of logic is executed (such as an authorization-related logic for a prepaid authorization request event).

The diagram below illustrates the event handling flow:



**Figure 4-2: Pricing Engine Event Handling Flow**

All stages of the processing of a single event are based on evaluating statements of dynamic logic. These statements are implemented as execution trees instantiated from the Implementation Repository. These trees are designed to be recursively executed.

When executing the tree's nodes, it is possible to directly access only data of entities related to the event and current pricing item (event, customer parameters, customer offer parameters, event variables, PIT variables, item parameters, and performance indicators). This data, together with some control logic, are called the evaluation context.

Event handling consists of the following steps:

1. Prepare the evaluation context for qualification:
  - a. Put the event entity into context.
  - b. Put the event variables into the context.
  - c. Load the customer data from the database and put into context.
2. Qualify the packages and items.
3. Enhance the context (add information) for all qualified items:
  - a. Load item parameter entities.
  - b. Load performance indicator entities and put them into context.
  - c. Put the PIT variables into the context.
4. Apply processing logic by invoking event handlers for each of the qualified qualification cases.
5. Dispatch:
  - a. Check whether the event should be dispatched, according to qualification criteria defined in the event dispatching case.

- b. Map the event to an external record when required; otherwise send the whole event.
6. Save the results if required:
  - a. Save all modified entities to the database.
  - b. Save a temporary copy of the rated event and dispatched records to the database or file for further processing and updating of the backend Rating database.
  - c. Save the external records that were created in the dispatching phase.
  - d. Commit the database transaction.

The following sections describe these steps in greater detail.

### Qualification

The Qualification process (also called Guiding to Service) qualifies a collection of pricing items for involvement in price determination.

The following factors affect the qualification process logic:

- Grouping of pricing items together into pricing packages
- Priority of the packages to be checked during the process
- Pricing item roles
- Order of pricing items within the pricing packages

The input for the qualification step is all pricing packages that are associated with the customer or subscriber. These include both packages assigned to the customer or a higher node in its organization hierarchy, and, business entity-level (market-level) automatically qualified packages.

The following steps are performed during the qualification process:

1. Package qualification – The Pricing Engine collects the packages and:
  - a. The Pricing Engine sorts the packages according to their priorities.
  - b. Then it evaluates each package having a qualification criterion. If the package fails the criterion evaluation, the package is filtered out of the list.
2. Item qualification – Each item in each remaining package is qualified. First, the events are filtered. The definition of an item specifies the type of service to which it applies. Events are first classified by Acquisition & Formatting, which associates the event type and service filter with each event for built-in filtering. For each qualification case (refer to handled event) in the item type:
  - a. Check whether the event is suitable for the qualification case; that is, the event type is exactly the same or a descendent of the event type specified in the qualification case.
  - b. Check whether the service filter of the event is suitable for the qualification case service filter. When the qualification case service filter is null, the pricing item suits all service filters.
  - c. If the qualification case contains a qualification criterion, invoke it.

- When packages reference other packages, only high-level packages are subject to qualification. For example, package A references package B, and is associated with customer C1. Package B is associated with customer C2. When an event for customer C1 arrives, only A's criterion is evaluated, but with an event for customer C2, B's criterion is evaluated.

The result of the qualification process is a list of pricing items to be applied to the given event.

### Execution Policy

The role policy specifies the minimum and maximum number of items that can be selected for each role. The default policy for each qualification case of a certain role is defined in the role PITs. It can be overridden in the derived PITs. The qualification cases are selected until the maximum policy is reached. If the minimum number of items for the event type is not reached, the event is rejected.

For example, the table below shows the role policy defined for each role:

Role	Minimum	Maximum
Rate	1	1
Allowance	0	99
Additional Charge	0	99

The following subscriber items are qualified to the event:

Item	Role	Priority
Standard Voice	Rate	2
F&F Voice	Rate	0
Closed User Group Voice	Rate	1
Allowance (duration 100)	Allowance	8
Allowance (duration 200)	Allowance	5
Allowance (duration 300)	Allowance	2

The table below shows the order of the items after sorting by priority, and which will be executed according to the role policy:

Item	Role	Priority	Executed?
F&F Voice	Rate	0	Yes
Closed User Group Voice	Rate	1	No
Standard Voice	Rate	2	No
Allowance (duration 300)	Allowance	2	Yes
Allowance (duration 200)	Allowance	5	Yes
Allowance (duration 100)	Allowance	8	Yes

Only the first item will be executed for the Rate role, because the policy is for both a minimum and maximum of 1. All allowance items are executed, because the maximum of the allowance role is 99.



### **Event Handler Evaluation**

This step applies computation logic associated with the qualified pricing items to the event. Before the actual execution of the PIT handlers, the PE loads all the necessary PIs from the database, or creates new PIs when none exist, and invokes the PIT's initialization handlers. In some cases, no qualified qualification case of a PIT requires the PI; in such cases, PIs are not loaded, created, or stored at all.

As explained above, some items require a certain processing order and others are not invoked directly by the Pricing Engine at all.

To do this, the Pricing Engine separates the items to be invoked manually into lists, one per item role. The remaining items are then resorted, taking into account the following factors (listed in order of importance):

- The item's invocation priority for its role
- The package's priority. If two packages have the same priority, the offer with the earlier expiration date comes first.
- Within a package, the item's order determines its priority.

### **Event Handler Invocation**

Once the list is sorted, the Pricing Engine loops over the list and invokes event handlers for each item.

Event handlers are responsible for updating the PI associated with the pricing item (according to PIT definitions), for performing any pricing logic (rating, calculating allowances, discounting, calculating additional charges, calculating chargeable attributes from monetary amounts, etc.), and for inserting the updated event object into the data storage area.

The Pricing Engine also updates PI reservation values when processing reserve, debit, and release requests for Online Charging.

### **Segmentation**

Most event charge calculations are not as straightforward as a flat rate per unit. More complex calculations require segmenting the chargeable attribute's quantity into smaller parts and rating each segment differently. For example, total air time might be divided into peak and off-peak segments, with a higher rate applied to peak units.

Segmentation can be repeated as many times as required, according to business requirements. The sum of quantities of the segments always must equal the chargeable attribute's quantity.

Each new segment, generated by the division of other segments, is assigned to:

- A quantity that is equal to or smaller than the quantity of its original segment.
- A key value in the rate table, which will be used later to locate the rate.

The segments are ordered by generation. This order is crucial for some segmentation operators.

The result of the segmentation is a sequence of segments. Each segment contains a quantity and key values that were assigned to it during partitioning. The key values define how the segment should be rated, according to the rate table.

### Dispatch

This activity is performed after the event calculations.

Each record to be dispatched is transformed to an external record, which is collected by the Rated Event Dispatcher mechanism. A separate record is written to the Rated Event output for each dispatching destination selected for a given event.

To determine the dispatching destinations defined for the current event, the Pricing Engine checks if the qualification criterion is empty or is evaluated as True.

If the condition is True, the rated event is mapped to the logical format required by the destination. When a mapping case does not exist, the Event entity is the logical format.

### Write Results

At the end of event processing, the Pricing Engine writes the rated event, PIs, and dispatched records to the relevant media.

The output media can differ from one environment to another. For example, Online Charging receives results in buffers, while Rating outputs are written into a database and files.

## 5. RATING ENVELOPES

---

The Pricing Engine is the only module that calculates charges. It can rate any chargeable quantities with any rating logic, using the CSP's definitions in the Product Catalog.

Rating uses different envelopes for different operations or rating environments. An envelope is an executable that transfers events from input media (files, messages, etc.) to the Pricing Engine, and back out from the Pricing Engine to the output media. Each envelope is built to match the technical requirements of the required business flow. All envelopes interface with the same Pricing Engine.

The envelopes that call the Pricing Engine include:

- **Postpaid Rater** – Used by the main postpaid rating process; receives events from Acquisition & Formatting and rates them.
- **Rerating** – Retrieves rated events from the Rating database and rates them again.
- **Online Charging** – Processes events submitted in real time for prepaid authorization and debiting, advice of charge requests, and similar online interactions between subscribers and Rating.

This chapter describes the functionality and flow of the envelopes, and also provides some overview information about additional processes that relate to the envelopes.

### Postpaid Rater (Offline Charging)

The Postpaid Rater (also referred to as the Usage Rater) receives events from Acquisition & Formatting after the formatting and guiding to customer processes, and calls the Pricing Engine to rate the event. The Pricing Engine writes the outputs to the output media.

The process is a daemon which reads files ready for processing from Audit & Control. One or more rater daemons can serve one rating partition and multiple cycles.

To avoid bottlenecks when processing a large volume of data, the rater may run multiple instances in parallel. Each rater rates its own population. The population is identified by Acquisition & Formatting, and can include one or more cycles that are stored in a particular partition. Acquisition & Formatting writes the different populations to different files and tags each file in the Audit & Control tables according to its population using the data group field. Each rater selects its own files according to the data group, which is an input for the rater. The rater supports multi-threaded processing.

A set of utilities is provided to support administration actions on the servers, such as stopping rating for specific cycle, stopping the server, and reinitializing.

## Postpaid Rater Flow

The Postpaid Rater flow is:

1. Connect to the relevant data stores.
2. Initialize the Pricing Engine.
3. Read the next file from Audit & Control.
4. Process each event in the file:
  - a. Call the Pricing Engine to process the event.
  - b. Accumulate statistics.
  - c. Commit the transaction for each group of events, where the number of events per group is configurable. Before committing the data, update the Last Event Processed table with the last event ID (for use by the Recovery mechanism).
5. Close the file and update the Audit & Control tables (number of records read, written, and rejected).
6. Loop to step 3 (read the next file).

## Recovery Mechanism

Rating normally terminates gracefully. When it receives a termination signal from Application Monitoring & Control (AMC), which controls all processes in the system, it continues processing the current file and performs a durable commit on all processed transactions before shutting down.

However, the process might be terminated while a file is being processed, by mistake or due to an application or system error, before it finishes processing all the events.

The Rating Recovery utility restores the system to a valid state. While processing events (before they are committed to the database), Rating maintains a record of the last event processed in the current file. The Rating Recovery process:

1. Retrieves the key of the last event processed in the recovered file from the Last Event Processed table.
2. Starts processing again from the point of failure, skipping previously processed events.

## Rerating

Events are rated as they enter the Rating system. However, some events may need to be guided and/or rated again later, based on new information (for example, if the subscriber purchased a new offer with retroactive effect). These events must be rerated before billing.

The Rerating flow follows these steps:

1. Identify the subscribers that require reguiding and/or rerating, and analyze this population to minimize the number of subscribers processed.

2. Extract the subscriber events to be reguided and rerated from the Rated Event database into files that are destined either for Acquisition & Formatting (for reguiding) or Rating (for rerating).
3. Feed the extracted events that need reguiding to Acquisition & Formatting for reguiding.
4. Rerate the extracted events that are directly marked for Rerating and those that have been reguided. (The events are first sorted by Customer ID and event date to ensure correct chronological processing.)

The diagram below shows the flow for reguiding and rerating selected events:

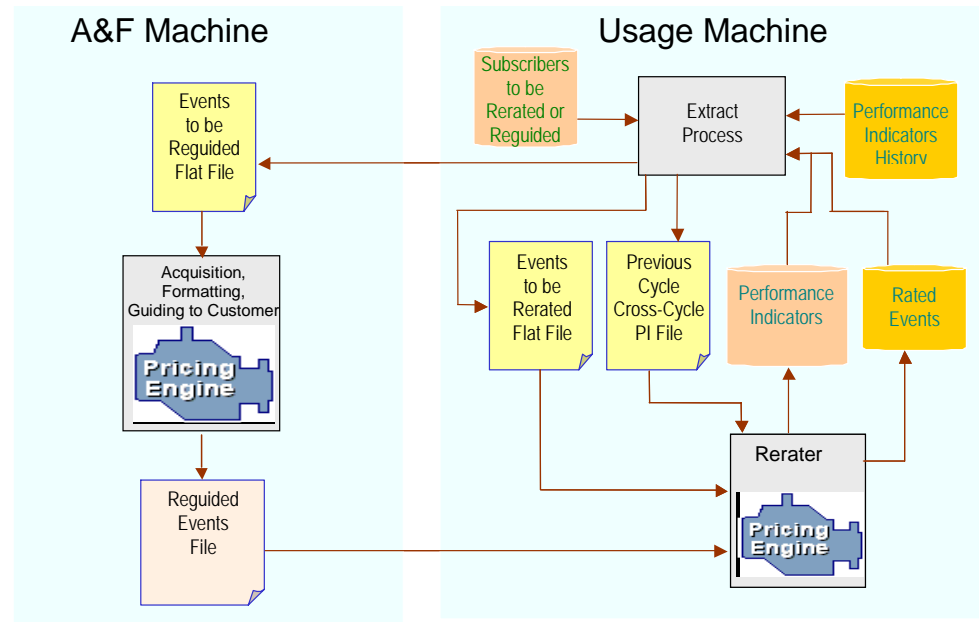


Figure 5-1: Rerating Flow

## Online Charging

Online Charging is responsible for the online and real-time transaction-based processing of charging and balance recharge requests that originate from mediation devices and Replenishment Management.

Online Charging is not in the scope of the Rating component, but a component by itself which interfaces with the Pricing Engine.

### Online Charging Flow

Request and charging events are transmitted by the mediation device to the Formatting and Routing Engine either directly, via a TCP/IP socket-based messaging interface, or via adapters. The Formatting and Routing Engine (FR) accepts, formats, and guides the subscriber to its customer. The event record is routed via a socket-based messaging interface to the Charging Engine.

The Charging Engine calls the Pricing Engine, which rates the event, writes the result when required, and returns the result back to the Charging Engine.

The Charging Engine updates the account's balance with the charge amount of the event or, in case of a charge request, updates the reserved amount.

The Online Charging flow is shown in the following figure:

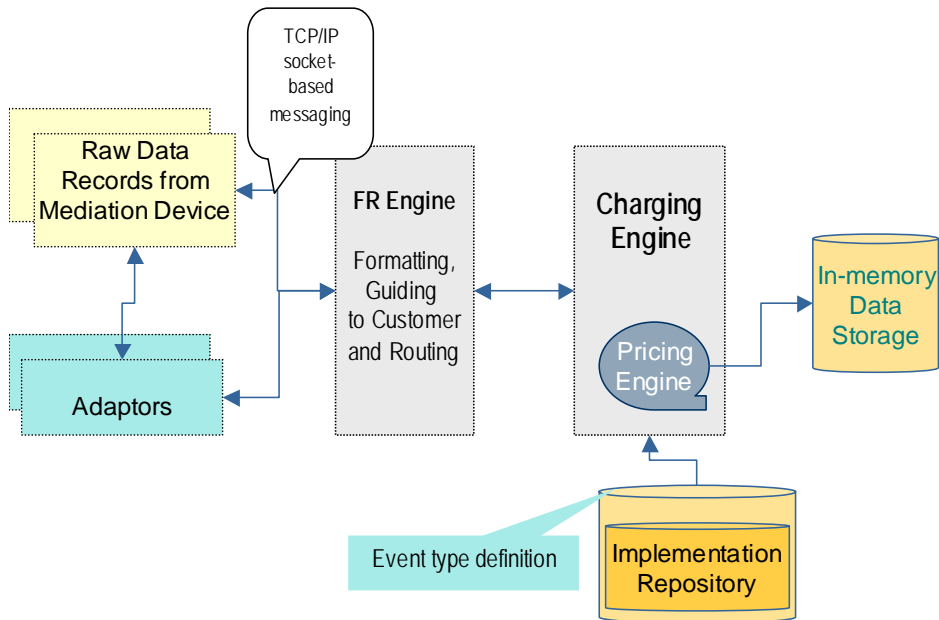


Figure 5-2: Online Charging Flow

## Interface to Pricing Engine

The Charging Management function of Online Charging is responsible for the transaction-based processing of formatted and guided charge requests (events). It processes the following charge requests:

- Session-based reservation and charge (debit and credit) requests
- Authorization requests, which involve a combination of balance credit, spending limit, and eligibility authorization checks
- Advice of charge (AOC) requests, which provide price and tariff parameters

Charging Management interfaces with the Pricing Engine, which rates the event and returns the results: charge amount, reservation charge amount, authorization answer, or AOC parameters. The Pricing Engine functions the same in Online Charging and offline charging (Postpaid Rater). All rating schemes are defined in the Product Catalog.

The Pricing Engine maintains a flag in its interface which indicates whether to save the results into the database. Online Charging does not always require saving the results; AOC and authorization requests are done without saving; while charge requests are written into the database (Online Charging sets the flag to True).

The Pricing Engine also can update performance indicator (PI) quantity values according to reserve, debit, and release requests to Online Charging, if the CSP desires.

## 6. RATING OUTPUTS AND USAGE DATA STORAGE

---

The Pricing Engine writes its results (PIs, rated events, etc.) to various output media. This chapter describes the data storage types supported by Rating and the flow of usage records from Rating temporary storage to the Oracle database.

### Postpaid Rating Output

Rating's main outputs are:

- Performance indicators
- Rated events
- Rejected events
- Dispatched records

The Pricing Engine writes the performance indicators directly into either an in-memory database (IMDB) or Oracle Rating data storage.

The rated events, rejected events, and dispatched records (referred to as “usage” here) are written to files or the Rated Event in-memory database. The decision whether to write to the IMDB or a file depends on the Rating configuration.

The rated events and rejected events are inserted into Oracle tables by the Replicator and Dispatcher mechanisms, which are described below. The dispatched records are written by the Dispatcher mechanism to their destinations.

### Data Storage Types

Rating can work with different data storage types, depending on the configuration of the environment. The data storage type used are Oracle databases and TimesTen in-memory databases.

The following table describes which of the Rating tables must be in Oracle only and which can be in either Oracle or TimesTen.

Table	TimesTen	Oracle
Performance Indicator	✓	✓
Customer Tables	✓	✓
Rated Events		✓
Rejected Events		✓

There are three possible options for the configuration of the database:

- TimesTen mode – All output is written to TimesTen

- Oracle mode – All tables are in Oracle
  - Combined mode – Some tables are in TimesTen and others are in Oracle
- Each of these modes is described in the subsections that follow.

### TimesTen Mode

The TimesTen mode is recommended for environments that require real-time rating, such as Online Charging for prepaid services and AoC (advice of charge, which informs a subscriber online of call or service rates). The IMDB provides very high performance but consumes memory resources.

In this mode, usage is written to TimesTen or files, and then inserted into Oracle tables by other processes as described below.

### Oracle Mode

For postpaid rating, where very high performance is not crucial, Rating can operate in Oracle mode, where all the tables are in the Oracle environment.

In this mode, usage is written to files and then inserted into Oracle tables by other processes as described below.

### Combined Mode

In combined mode, some tables may be in Oracle and others in TimesTen, to optimize the balance between performance and resources. For example, it may be decided to store customer data in the TimesTen database while the PIs are in Oracle.

In this mode, usage is written to TimesTen or files, and then inserted into Oracle tables by other processes as described below.

## Usage Flow from Rating to Oracle

Rating writes usage records (rated events, rejected events, and dispatched records) all together to a temporary usage storage area, which can be either files or the Rated Events table in TimesTen. In either case, the data are then transferred to Oracle.

### TimesTen to Oracle

If usage records have been temporarily stored in TimesTen:

1. The XLA Replicator reads rated events, rejected events, and dispatched records from TimesTen logs and writes them into files.
2. The Dispatcher reads the files and:
  - a. Inserts rated events into the Oracle Rated Events table.
  - b. Inserts rejected events into the Oracle Rejected Events table.
  - c. Sends dispatched events to their destinations.



## **Files to Oracle**

If usage records have been temporarily stored in files, the Dispatcher reads the files and:

1. Inserts rated events into the Oracle Rated Events table.
2. Inserts rejected events into the Oracle Rejected Events table.
3. Sends dispatched events to their destinations.

## **BLOB Compression**

The Rated Event and Rejected Event database tables include a column for storing dynamic data in BLOB format. This column is populated in Full Binary and Hybrid persistence modes.

To optimize resource usage, Rating can call third-party dynamic libraries that implement an API for compressing and expanding binary data, to compress the BLOB column when storing data in the Oracle database.

Rated Event and Rejected Event table BLOB data are compressed by the Dispatcher process, which populates Oracle tables. BLOB data are expanded again by the Pricing Engine when queried for events or rejected events.

## **Replicator Mechanism**

The Replicator mechanism is a daemon process that reads rated events from TimesTen logs and writes them into files. These records then are inserted into the database.

TimesTen maintains logs and a database image on the disk in order to recover from a hardware failure without loss of data. The Replicator uses this log to record events inserted into the database.

After an event is rated, it is inserted into TimesTen by updating a single event record in the TimesTen Rated Events table (each update overrides the previous record). The transaction is updated in the TimesTen log and captured by the Replicator process. This technique saves the overhead of deleting rated events from TimesTen.

## **Dispatcher Mechanism**

The Dispatcher mechanism dispatches rated events to various destinations.

Core targets, which always should be defined for the Dispatcher mechanism, are the Oracle Rated Events and Rejected Events tables.

For more details about Dispatcher mechanism, see “Dispatcher” in chapter 8.

## **Performance Indicator Maintenance**

Rating creates, updates, and maintains performance indicators as described below.

Rating also supports the Product Catalog’s global flag for PI attributes, so the Pricing Engine can distinguish between “global” and “partitioned” PI attributes. “Global” attributes are attributes that exist once in a partitioned PI,

whereas “partitioned” attributes have a different instance in every partition. (“Partitioned” is the default value.)

### Performance Indicator Creation

Rating creates and updates PI records when it processes an event. Rating identifies all the PIs that may influence, or be affected by, the event, based on the event type and the Product Catalog definition.

If a record does not exist for the PI in a specific cycle period, Rating creates one.

If the relevant offer includes allowances that can be rolled over to the new cycle period, the counter of remaining units is rolled over to the new period when the new record is created, and the previous cycle period is locked for that subscriber.

If no event is received that creates a PI record for a cross-cycle PI, the maintenance process creates new cross-cycle PIs, ensuring that the roll-over information is printed on the bills of subscribers who have no events during the cycle.

### Performance Indicator Archiving

PIs can be kept in memory during the Rating process. However, the amount of data that can be stored in memory is limited, so only essential data should reside in memory. These data include PI data that applies to open cycles.

Once a cycle is locked, no new events can affect the records, and the Rating process no longer requires these records.

In the archiving process, all data associated with the locked cycle are extracted to a file using a general extract mechanism. The extract file is then loaded into the Conventional PI History database in the Oracle environment, where the PI records are available for queries, rerating, and potential extracts.

## 7. RATING INPUTS

---

Rating requires information from external systems to rate an event. This information includes:

- Customer information, which includes the offers acquired by the customer and other customer-specific parameters.
- Product Catalog information, which includes information from the Implementation Repository (where the event types, simple and complex data types, and pricing logic are defined), the Pricing Package Catalog, Service Package Catalog, Discount Package Catalog, and the Auxiliary Repository.

### Rating Data Storage

All external data required by Rating are loaded into the Rating data storage area by processes that are responsible for:

- Extracting the information from another system (if the system is an Amdocs component) and creating formatted records that are recognized by the load process.
- Loading the formatted records to the Rating data storage area. This data storage may be defined either in the conventional database, or in the IMDB.

The importing of external data into local data storage complies with Rating's basic objectives, ensuring that:

- Rating is decoupled from other components, permitting integration with both Amdocs components and third-party systems (the "best of breed" approach).
- All data are accessed locally, eliminating redundant network data transfers and complying with Rating's performance requirements.
- Only required data are stored, minimizing the Rating server's data storage requirements.

Pricing Engine, Rating, and Online Charging reference data can be updated at run time. Daemon processes that need ongoing updates of reference data can initiate updates without affecting batch processes, which continue to use the previous reference data until a new version is available.

### Customer Information Input

The Customer Extract and Customer Loader processes (together also referred to as the Update Handler) are responsible for extracting, loading, and updating the Rating customer information data storage area with data retrieved from the Customer Management system.

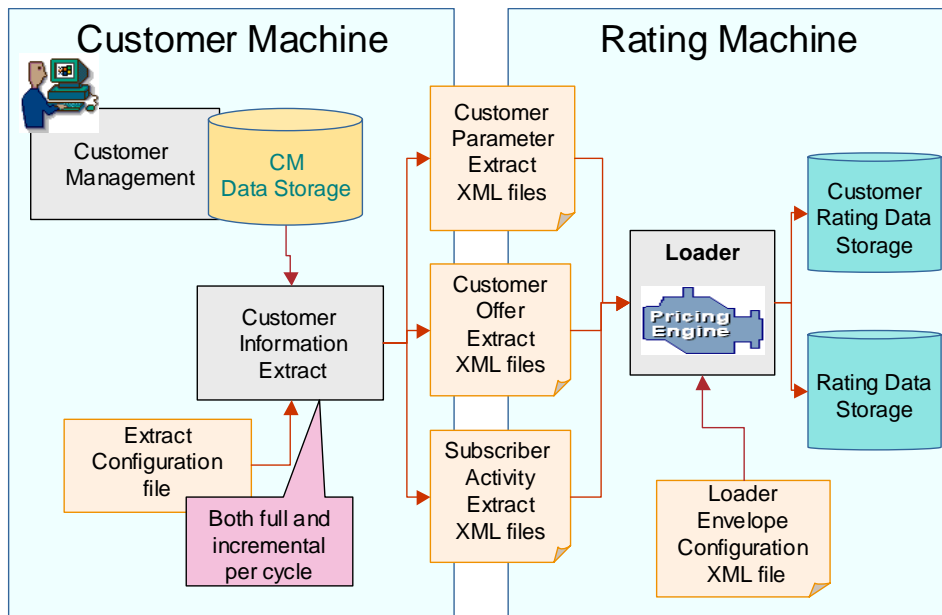
## Customer Data Extracted for Rating

The Update Handler extracts all information that must be retrieved from the Customer Management database for Rating purposes. The data can be divided into two types, which represent two database tables:

- Offer information, which contains a list of offers applied to the subscriber. This information is required for Rating to qualify the event (that is, guide it to a service).
- Subscriber-specific parameters, used by Rating both in the qualification criteria and the calculation models. Subscriber parameters may be stored in the Customer Management database at two different levels:
  - Dynamic Parameters – These offer-associated parameters are linked to a specific subscriber and populated on offer acquisition. For example, a “friends and family” reduced rate requires a list of selected numbers, which are stored in the Customer Offer Parameters table.
  - Static parameters – These additional parameters are linked to the subscriber and stored in various Customer Management tables. For example, the subscriber’s birth date, which may be required to qualify for a senior citizen discount, is stored at the subscriber level, and not necessarily associated with a specific offer applied to the subscriber.

## Customer Information Flow

The diagram below describes the data flow from the Customer Management area to the Rating customer data area:



R019

Figure 7-1: Customer Information Flow to Rating

## Product Catalog Input

Rating requires reference data from the Product Catalog in order to:

- Recognize the hierarchical structure and elements of offers, packages, and pricing, service, and discount items
- Recognize the various defined data types and structures
- Recognize the qualification criteria, pricing and service item types, event handlers, and extension functions required to process and rate the event
- Recognize any secondary reference data (auxiliary tables) used by the pricing logic

Product Catalog distributes its data to a generic distribution table in XML format. The Reference Table Synchronizer (RTS) copies the data to the application distribution tables and activates the relevant callback functions that extract the data from the XML files and populate the relevant Pricing Engine reference data tables.

Product Catalog data loads for offers, packages, and pricing items are incremental (that is, only new data are loaded), in contrast to the Implementation Repository. The Rating API recognizes Product Catalog baseline versions, updating the Rating tables accordingly (merging prior versions into the baseline version and deleting obsolete versions).

The data generated by the Product Catalog can be categorized in three different groups:

- Implementation Repository
- Pricing, Service, and Discount Package Catalogs and Offer Catalog
- Auxiliary Repository

## Implementation Repository

The Implementation Repository is the part of the Product Catalog that defines:

- Elementary data types, for example, the Volume data type and its unit of measure
- Complex data structures, such as definitions of events, external records, customer models, and mapping and dispatching cases for external records and events
- Pricing item types, which contain rating logic, performance indicators and their extract definitions, and item parameter definitions
- Qualification criteria and their definitions

## Pricing Package, Service Package, Discount Package, and Offer Catalogs

A pricing package groups together multiple pricing items to be sold together. A service package groups together multiple service items to be sold together. A discount package groups together multiple discount items to be provided together. An offer groups together multiple pricing, service, and discount packages.

Rating retrieves the following information from the Pricing Package Catalog, Service Package Catalog, Discount Package Catalog, and Offer Catalog on loading the Pricing Engine:

- Hierarchy of offers, pricing packages and pricing items, service packages and service items, and discount packages and discount items
- Pricing item-specific, service item-specific, and discount item-specific parameters
- References from the pricing items to pricing item types, and from service items to service item types, and from discount items to discount item types, which are part of the Implementation Repository and define the actual rating logic.

### Auxiliary Repository

The Auxiliary Repository contains secondary reference data used by the pricing logic. This includes such objects as period sets, special day sets, and tables based on the generic reference table mechanism.

These data typically are loaded from Oracle database tables when the Pricing Engine is loaded. However, some of these data are not known to the core Pricing Engine, and are loaded by extension libraries from external data sources.

## 8. RATING OUTPUT INTERFACES

---

Rating processes events, the results of which are rated events and performance indicators that are used by downstream components and external systems.

Components which may require the results of the Rating process include:

- Billing – The Billing process requires the accumulated usage (stored in performance indicators) of the subscriber to prepare the bill.
- Bill Formatter – Customers may request detailed billing. The subscriber's rated events are input to the bill formatting process.
- Customer Management – Queries may be performed on billed and unbilled usage.
- External systems such as data warehouses, third-party providers such as international carriers, and value-added service providers may require Rating information.

All data required by external components are supplied by Rating extracts and APIs. This approach complies with Rating's basic objectives; ensuring that:

- Rating is decoupled from other components, permitting integration with both Amdocs components and third-party systems (the "best of breed" approach).
- External components do not access the Rating area directly, so they cannot degrade Rating's high performance.

These Rating modules are responsible for exposing Rating results to external components:

- Rating extract
- Online usage queries
- Dispatcher mechanism

The interfaces of each of these modules to other systems are described in the sections that follow.

### Rating Extract

Rating extracts PI and rated event information from the Rating database. The Rating extract is used mainly by the Billing application.

The population for the Rating extract may be:

- The entire billing cycle population
- A list of customers supplied by Billing (or another external component)

The Rating extract flow is:

1. Extract the required information from the Rating data storage area (rated events or performance indicators, depending on the run mode).

2. Create and populate a file containing records in an external record layout defined in the Product Catalog.

Customers requiring rerating are flagged and sent to the Rerating envelope, while the rest can proceed to Billing.

## Online Usage Queries

Customer Management may require data synchronously from Rating. These data may include:

- Subscriber usage – A list of all a subscriber's rated events over a period of time
- Subscriber accumulators – A list of all a subscriber's performance indicators over a period of time

Rating supplies an API which uses the same modules that are used in the Rating Extract process.

The Customer Management Online API searches the data storage according to the input criteria (subscriber, cycle information, and date range), and returns a list of either rated events or performance indicators belonging to the subscriber over the requested period.

The format returned is the format requested by the calling system, which is defined in the Product Catalog Implementation Repository as an external record layout type.

## Dispatcher

The Dispatcher mechanism dispatches rated and rejected events to their intended destinations. The Dispatcher mechanism gets its instructions from a dispatching case, which the administrator defines for each event type in the Product Catalog.

Because events are dispatched immediately after rating, rated events can be sent continuously to internal and external destinations, such as service providers, data warehouses, or partner relationship management systems.

Each dispatching case contains three elements:

- Qualification criterion – Qualifies the event type for a dispatch destination.
- External record reference – The external record type to which the rated event is mapped.
- Dispatcher target identifier – The identifier of the dispatch destination to which the case applies.



amdocsrating

# **Rating 6.0 Rerating Specification (Internal)**



# 1. INTRODUCTION

---

Rerating is part of Amdocs Rating. Rater, the principal rating application of Amdocs Rating, rates events as they enter the system. For various reasons, this might produce incorrect rates. The incorrectly rated events must be rerated before the subscriber is billed. Rerating performs this operation.

## Purpose and Scope

This part of this book provides functional and technical descriptions of Rerating. It includes a general overview of Rerating, and lists some of its design principles.

It also presents details of the processes involved in identifying subscribers that may qualify for rerating, as well as various scenarios illustrating the rerating use.

## Rerating Overview

This section briefly describes the main functions performed by Rerating, its interaction with other Amdocs Rating components, and the process flow.

### Main Functions

The Rerating process comprises two major functions:

- A set of activities that identify those subscribers that qualify for rerating, the events to be rerated, and the subscribers they belong to
- The Rerating activity that recalculates the event rates

### Interaction with Other Components

Rerating interacts with the following Amdocs components:

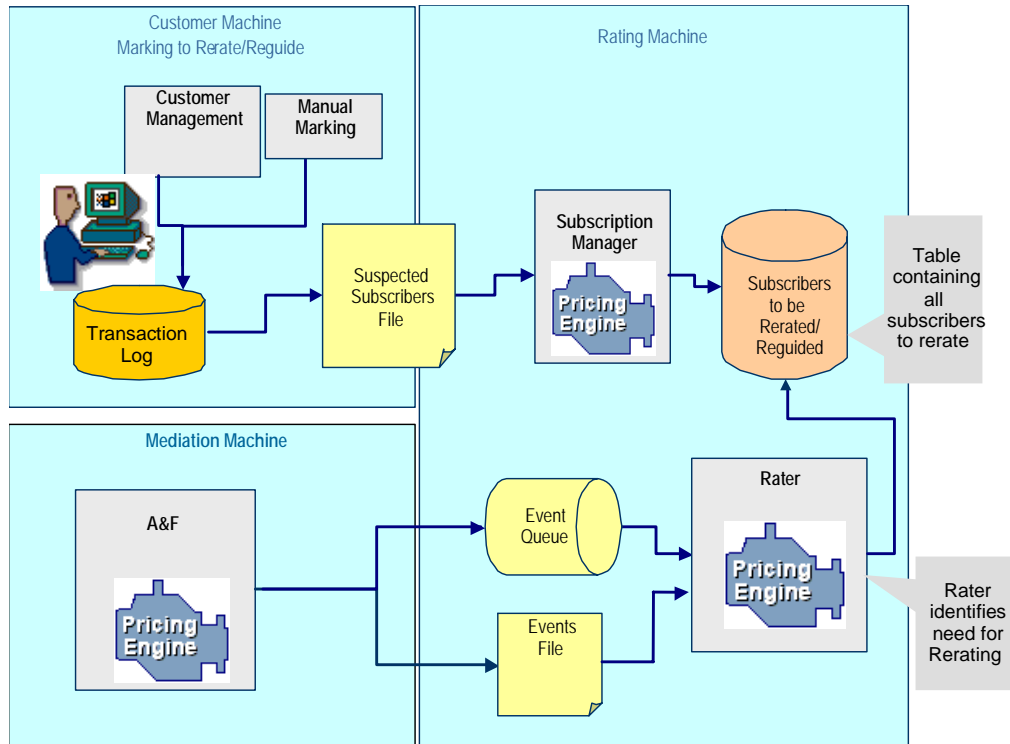
- Acquisition & Formatting
- Product Catalog (indirectly, via the Pricing Engine)
- Billing (the Billing process may identify a set of customers requiring rerating)
- Customer Management (the Customer Management process sends activities that trigger rerating)

### Process Flow

The flow of the Rerating process is divided into several steps:

1. Identifying and marking customers and subscribers that require rerating.
2. Further analyzing the marked population to reduce the number of subscribers marked for rerating.

This process is illustrated by Figure 1-1.



**Figure 1-1: Marking subscribers for rerating**

3. Reguiding the subscriber events destined for reguiding. The extracted events are reguided by Acquisition & Formatting.
4. Extracting the subscriber events to be rerated. The Usage data is extracted from the Rated Event database into files that are destined for Rerating.
5. Rerating the subscriber events marked for rerating. All events to be rerated are sorted by their Customer ID and event date, to ensure correct chronological processing.

The Rerating flow is illustrated by Figure 1-2.

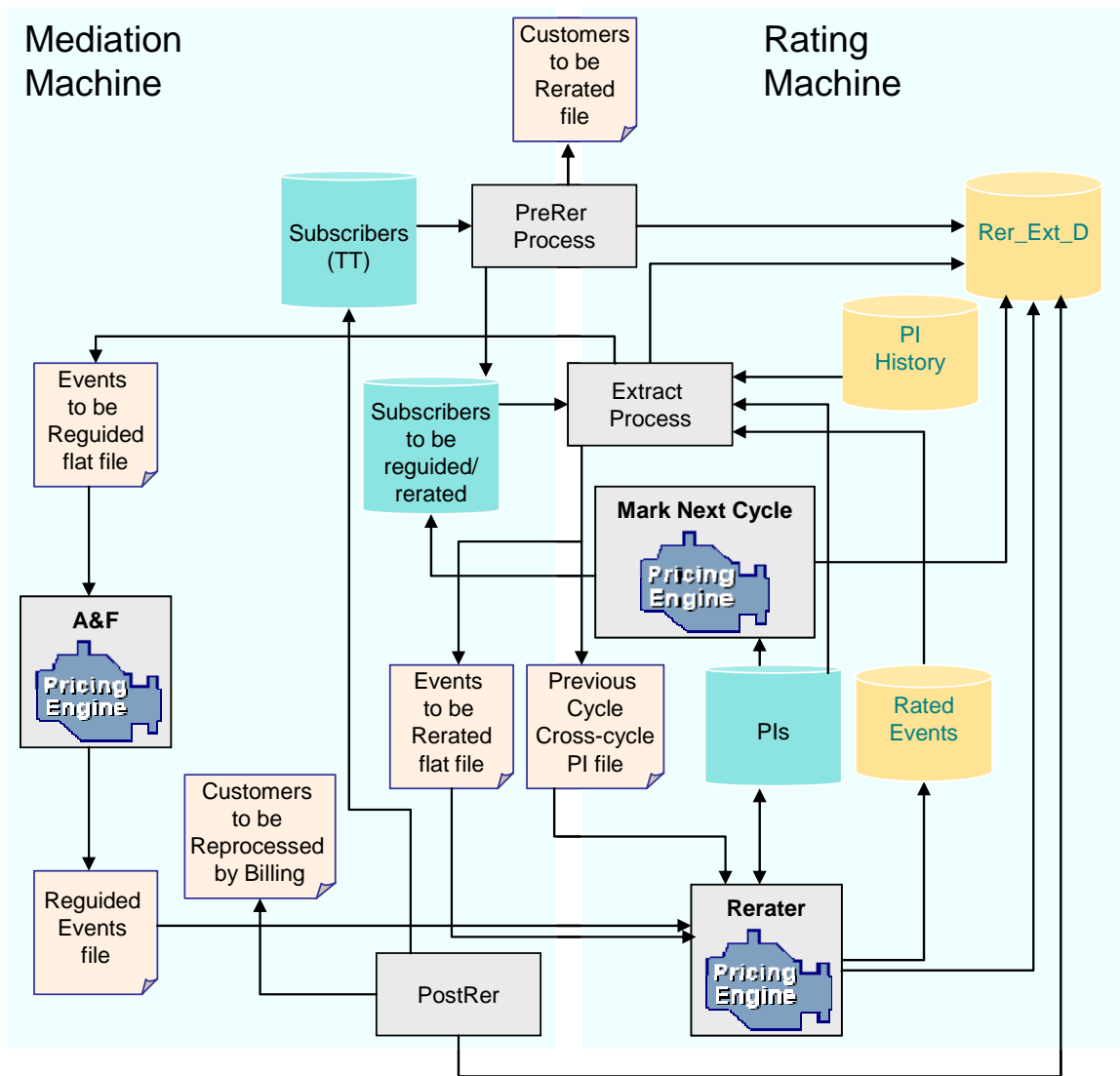


Figure 1-2: Extracting information and Rerating flow

## Billing-requested Rerating

After rating has been completed and the cycle has been locked, control of the cycle is passed to Billing. During Bill Preparation, the Billing QA process may detect that the Rating Undo is required for a certain population due to erroneous rating, charges, etc.

In this case, Billing prepares a list of customers and subscribers requiring rerating, and invokes the Rerating process.



## 2. RERATING SCENARIOS

---

This chapter describes Billing scenarios that trigger Rerating.

### Customer Management – Retroactive Activities

Customer Management retroactive activities (backdating) may affect the rates of events that have already been rated.

Following are examples of backdated transactions that can trigger rerating:

- Cancellation of a subscriber whose offers include prorated allowances (allowances that should be allocated proportionally to the period of time).
- Service change that includes:
  - Backdated price plan change
  - Backdated assignment of a new offer
  - Backdated removal of an offer
  - Backdated change of an effective date for an existing offer
  - Backdated re-assignment of services to a different pay channel

Some of the above activities require both reguiding and rerating, and some – only rerating. For example, re-assignment of a service to a different pay channel requires both reguiding and rerating include the.

### Customer Management – Allowance Activities

Customer Management activities that affect the allowance (free units) allocated to a subscriber may affect the rating of events.

The following Customer Management transactions may require rerating:

- Cancellation of a subscriber whose offers include prorated allowances
- Expiration of an offer that includes prorated allowances

Cancellation of a subscriber or offer during a cycle month implies that the originally allocated allowance was for the full cycle month, and should be allocated proportionally to the effective period.

### Rating – Chronologically Inaccurate Events

Events may enter the Billing system in the wrong chronological order. If the subscriber's offer includes allowances (free units), the allowances may be incorrectly applied to events, because certain events were received in the system before events that were generated earlier. This may also occur in stepped offers where the step used to calculate the rate is incorrect due to chronologically inaccurate events.

The effect of a chronologically inaccurate event must be analyzed in order to determine whether rerating is actually required.

## Rating – Inaccurate Product Catalog Entries

Reference data used by Rating may be found to be incorrect after rating of events has been performed. This may happen when introducing new programs, or new versions for existing programs, with insufficient verification. Examples may include:

- Incorrect rates in the Rate tables – For example, a rate may be defined as \$2.00 per minute instead of \$0.20 per minute. All subscribers who partake in this rating plan require rerating. Such mistakes may be detected only by the bill QA process.
- Incorrect qualification criteria – Implementation Repository, which contains definitions of the qualification criteria, may be incorrect. In this case, events may be qualified to an incorrect pricing item.
- Incorrect prioritization – In this case, the incorrect price item may be qualified to events.
- Incorrect handlers – Implementation Repository may contain inaccurate handlers, which in some cases cause incorrect rating results. In this case, the affected subscribers need to be rerated after the handler is fixed.

The above list may include any item/parameter/handler that is incorrectly defined in Product Catalog, and is discovered after the events have already been rated.

## Billing – Undo Bill Preparation

Prior to confirming a bill, a sample customer population undergoes quality assurance in order to assure accuracy of their bills. If errors are detected, the Billing Undo is prepared to revert the billing process for either the entire cycle or a specific population of customers and subscribers.

If the billing administrator requested Undo with Rerate, this process moves all the customers in the selected group to the Rerate table, extracts a list of customers that need to be rerated, and sends the list to Rerating.

## PI Reconstruct – Inaccurate Accumulation

The PI reconstruct process inserts cross-cycle PIs into the next bill cycle month and initializes them. For each of the previous month's cross-cycle PIs, a new cross-cycle PI is defined, and its value is set to the accumulation remaining at the end of the previous month.

When rerating is performed, the previous cross-cycle PI value may change. This can create inconsistency between the old and new cross-cycle PIs.

Although the PI of the next bill cycle month is marked for rerating by the current bill cycle's Rerate, there are some scenarios where the problem requires special handling. This is further explained in *Appendix C*.



## 3. IDENTIFYING SUBSCRIBERS FOR RERATING

---

This chapter describes the processes used for flagging subscribers for rerating.

### Overview

The first process in the Rerating flow is the identification and marking of subscribers that require rerating.

As described above, one of the two components may trigger a check as to whether rerating is required:

- Customer Management – As a result of backdated and regular activities
- Rating – As a result of change in some rating programs, or of events arriving in incorrect chronological order

It is also possible to force rerating for an arbitrary population using maintenance procedures that need to be performed periodically due to erroneous Product Catalog entries, and under certain unpredictable operational circumstances. Rerating can also be forced by Billing as part of the Undo operation.

The process flow for determining the need for rerating on an ongoing basis is an integral part of the general Billing system architecture. Subsequent sections describe this flow.

### Customer Management Activities

Customer Management activities are published on the TRB for use by any component that subscribes to this information. This mechanism is used by Event Processing to update its Customer database for both guiding and rating. The same mechanism is also used to trigger the process determining whether rerating is required.

The general approach is to extract all “suspect” activities (described in the *Rerating Scenarios* chapter), using the same Update Handler mechanism for synchronization as for all the activities required by the Rater Customer database, into an XML file.

The Subscription Manager process receives these activities and transforms them into standard “events,” called “Subscriber Activity” events. These events are passed to the Pricing Engine. Each event is defined in Product Catalog as any other network event processed by the system, and the Product Catalog architecture is used to implement the rules for determining whether rerating is required. The event includes all the attributes required for this determination, which can be retrieved from the transaction published by Customer Management. This usually includes the activity code, subscriber keys, date information, and so on.

The Product Catalog implementation uses qualification cases and handlers to “trap” such events and determine if they should trigger rerating. The determination is based on specific subscriber offers or on a general policy. If the transaction should trigger rerating only for a specific offer (as in the case of subscriber cancellation, which requires rerating only if the subscriber has pro-ratable allowances), the handler is only implemented on the specific allowance *pricing item type* (PIT). If, on the other hand, the transaction should require rerating in any case (as in the case of a backdated price plan change), the handler is implemented on the market-level default offer, which is always qualified for such events.

The above flexible strategy enables the system to decide when to mark subscribers for rerating at the implementation level, which may differ from operator to operator. See Figure 1-1 for the diagram depicting the flow of this process.

The Subscription Manager process may work in two different modes:

- Pessimistic/lazy – All “suspect” activities mark the subscriber for rerating
- Optimistic – Implementation Repository is investigated in order to determine whether the subscriber requires rerating

### Rating Detection

Rating is responsible for detecting at least two situations that might require rerating:

- A specific family of rating programs that need constant “monitoring”
- Out-of-sequence events, which may invalidate previous rating results (wrong allowance applied to events, wrong steps applied to events, and so on)

The general approach is to exploit the flexibility in designing the PI structures, implementation handlers for each PIT, and use extension functions for implementing this “monitoring” or “rerate determination logic” in Product Catalog. The responsibility for assessing the situation correctly and for marking the subscribers for rerating lies with the Product Catalog implementation. As the next section shows, a number of cases may have more than one solution. The logic used for detecting the necessity for rerating depends on both the frequency of occurrence of the case and on the complexity of the required rerating.

For example, an incorrect sequence of events does not have any significance for a “flat rate” program, whereas it may be significant for a “stepped rate” program. Consequently, the required logic should reside in the PITs of the stepped rate programs. Furthermore, for a stepped rate program, only certain situations must cause rerating.

In order to differentiate between these cases, information stored in the PI must enable the addition of special to the PIT handlers. It is up to the implementer to use the above approach or a simpler one, which marks the subscriber for rerating for every occurrence of out-of-sequence events. If there are few customers with stepped programs, and the total overhead introduced by rerating is insignificant, the simpler approach is recommended. If, however,

rerating introduces significant overhead, the extra effort to accurately determine the rerating need is well justified.

## **Customer Management – Marking Subscribers**

This section provides the analyses and solutions for specific cases. It should be emphasized that further tuning, as well as implementation of additional approaches, is possible. The system enables introduction of new programs that do not fall into the categories described here, and development of the logic required for rerate need detection associated with these new programs.

### **Guidelines**

Customer Management publishes transactions (such as subscriber cancellation) that may result in the need for rerating. As Customer Management does not have all the information necessary to determine if a transaction should result in rerating the subscriber's data, additional information, known only to Rating, is required to supplement the Customer Management marking. The following section describes the process of identifying and marking subscribers that require rerating.

### **Customer Management to Subscriber File Flow**

Subscriber change, service change, backdated transactions, and any other transactions that indicate the need for rerating, are published on the Transaction Broker. An extract batch process, listening to the transactions published by Customer Management, performs the following:

1. Retrieves all activities that may influence rerating. For example:
  - a. Identifies all subscriber cancellation transactions
  - b. Identifies all service change transactions
  - c. Identifies all backdated transactions, which include:
    - Change of price plan
    - Service change, which includes:
      - Assignment of a new offer
      - Change of an effective date for an existing offer
      - Re-assignment of service(s) to a different pay channel
2. Inserts the subscriber into an XML file that contains the following data:
  - Subscriber ID
  - Customer ID
  - Payment Channel ID
  - Cycle Code
  - Activity Code
  - Issue Date
  - Effective Date
3. Closes the file when it contains X records, or once Y time has elapsed.

4. Sorts the file by bill cycle code.
5. Divides the file into separate files based on the cycle code.
6. Notifies Audit & Control of the existence of the files targeted for Subscription Manager processing.

This job may run daily or as a daemon process.



*This process is based on the Customer Incremental Extract process, which extracts changes in the Customer area that affect the Rater Customer database. The full or incremental processes may be run as a single process or as separate processes.*

## Subscriber File to Subscription Manager Flow

Once the file has been created, the Subscription Manager processes it. It performs the following operations:

1. Retrieves the file from Audit & Control.
2. Works in one of the following two modes (defined in the configuration file):
  - Pessimistic – All transactions published by Customer Management result in the subscriber being marked for rerating.
  - Optimistic – Subscription Manager examines Implementation Repository in order to verify that rerating is required. In this mode, Subscription Manager takes the following steps:
    - i. Prepares the special “Subscriber Activity” event type record.
    - ii. Populates the event’s Service Filter field with the Activity Code received.
    - iii. Invokes the Pricing Engine API that determines whether rerating is required and marks the subscriber or customer for rerating accordingly.
3. Notifies Audit & Control of completion of the file processing.



*This process is based on the Customer Incremental Load process that updates the Rater Customer database with changes identified in the Extract process. The full or incremental processes may be run as a single process or as separate processes.*

These files contain the subscribers who may require rerating/reguiding.

## Pricing Engine Processing of Subscriber Activity Events

The Pricing Engine receives the subscriber activity events from Subscription Manager and, using the definitions provided in Product Catalog, determines whether the subscriber requires rerating/reguiding due to the activity that has occurred. For details, see the “Rating – Marking Out-of-Sequence Events” section.

## Rating – Marking Out-of-Sequence Events

Upon receiving an event, Rater compares the event date with the last recorded event date in the PI. If the current event has an earlier date, the subscriber may require rerating.

Depending on the implemented rating scenario, the Product Catalog definitions determine if rerating is required, and the subscriber is marked accordingly.

### Option 1 – Mark Subscriber without Further Investigation

Rating, via a core extension function, adds the subscriber to the Subscriber Rerate data storage table.

This table resides in an in-memory database (IMDB). For further details, see Appendix A.

### Option 2 – Investigate if Rerating is Required

Inaccurate chronological events require rerating only in the cases where:

- Allowances have been incorrectly granted on previous events
- Incorrect rates were incurred due to stepped rating plans

### Allowance Evaluation Mechanism

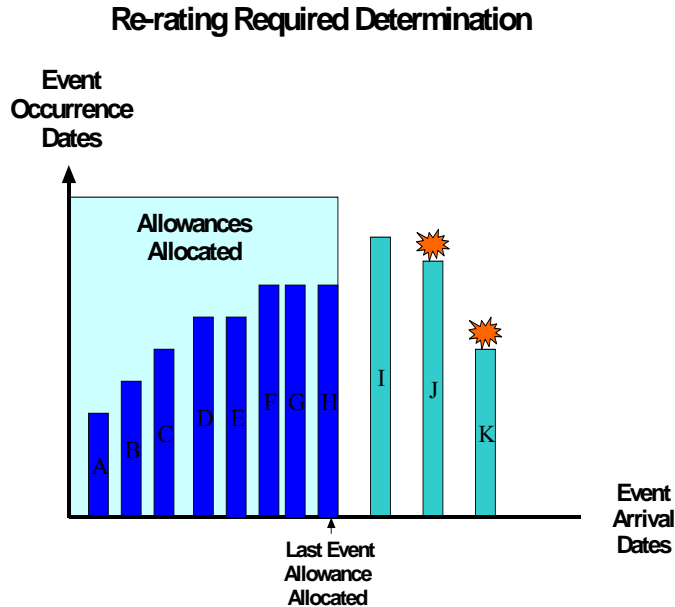
Rating is required to ascertain that:

- The event that entered the system late is entitled to an allowance, and
- If the allowance granted on a previously received event is to be annulled or modified.

Chronologically out-of-sequence events do not necessarily imply that rerating is required. Rerating is only required when:

- The subscriber has allowances (free units), and
- The event that was received late should have received the allowances, whereas an event received later received these allowances.

The following diagram describes the logic in the Implementation Repository:



**Figure 3-1: Chronologically inaccurate rerating due to allowance determination**

In the above diagram, events “J” and “K” are received in the incorrect order. Therefore, these events imply that the subscriber might be marked for rerating. Event J has no effect on the subscriber’s rating. Even though Event J occurred before Event I, and was received later than Event I, it would not have been entitled to receiving allowance, because all allowances were applied prior to Event J’s occurrence.

Event K does have an effect on the subscriber’s rating. If Event K had been received at the time of its occurrence (between B and C), then Event H would not have been given the allowance.

To support the above conditions, the Last Event Date Allowance Allocated data should be maintained in the Allowance PI.

The Allowance PIT should be extended to:

- Check if the event date is earlier than the last event date allocated allowance, that is, if the events arrived in the wrong chronological order.
- Insert the record into the Subscriber Rerate data storage table if required.

## Stepped Evaluation Mechanism

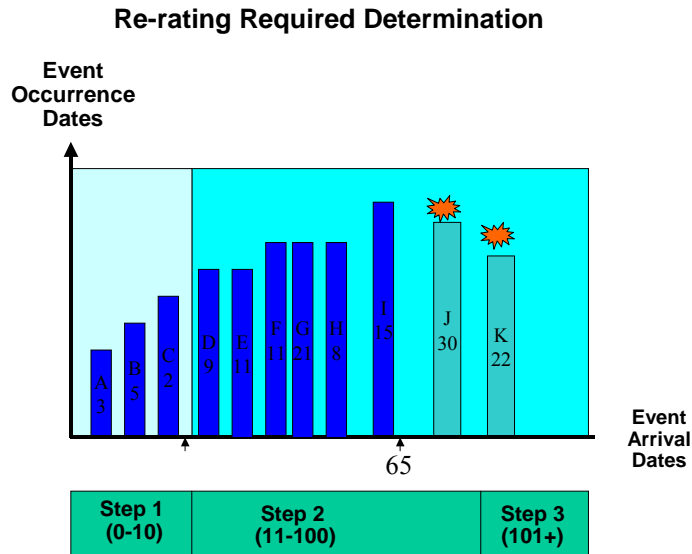
Rating is required to verify if the event that entered late would affect the step level of any events that occurred later than the newly arrived event.

Chronologically out-of-sequence events do not necessarily imply that rerating is required. Rerating is only required when:

- The subscriber has a stepped rating plan, and

- The event received affects the step of the next events.

The following diagram describes the logic in the Implementation Repository:



**Figure 3-2: Chronologically inaccurate rerating due to step determination**

Events “J” and “K” are received in an incorrect order. Therefore, these events are candidates for marking the subscriber for rerating.

Event J may have no effect on the subscriber’s rating. The event occurred between H and I. Even though it was received later than Event I, it would not affect the step determination of Event I. Event K does have an effect on the subscriber’s rating. If Event K had been received at the time of its occurrence (between E and F), then Event I would have been rated with a different step level.

In order to meet these conditions, an array of all steps and the date of the last event that participated in the step would be required to be maintained in the PI.

The cost of data storage and complex implementation does not justify this sophistication. As a result, the following is recommended:

- If a stepped rating plan exists, and an out-of-sequence event is detected, the subscriber is marked for rerating unless the subscriber event is currently being rated at the last step level, and the event occurred after the last step level has begun.
- The Last Event Date on the previous step is stored in the rated PIs.
- A new handler called Check Steps for Rerate, invoked on a stepped pricing item, is introduced in order to:
  - Check that the event date is earlier than that of the last event received (general rate), and
  - Check that, if the current step is the last step of the Rate table, the event occurred prior to the changing of the step, or

- Check if the step is not the last step, and
- Insert the record into the Subscriber Rerate data storage table if required.

## Manual Marking of Subscribers

If rerating is required due to incorrect reference table data, the affected population needs to be located, and the resulting list of subscribers added to the Subscriber Rerate data storage tables. To distinguish between the possible cases, a subscriber selection process has to be developed, usually implemented as an SQL script, according to the problem at hand. Once a list of subscribers is assembled, it is loaded to the Subscriber Rerate data store table.

As a standard, the system provides a job that locates subscribers that have a specific offer. This job is adequate for any Product Catalog data problem that can be directly translated to a list of subscribers having this offer and requiring rerating.

This job receives as input:

- Bill Cycle Code
- Offer IDs
- Effective Date Range

All subscribers that have the specific offer(s) are extracted from the Customer Management (or Rater) data storage, and inserted into the Subscriber Rerate data storage table.

## Agreement-level Handling and Marking of Subscribers

If the Pricing Engine recognizes that the subscriber partakes in an agreement-level pricing item, the entire customer record is marked for rerating. In this case, all subscribers related to the customer are rerated, ensuring that the agreement-level PIs (in particular, Allowances PIs) are calculated correctly.



## 4. DETERMINING EVENT RERATING REQUIREMENTS

---

This chapter describes the procedures the Pricing Engine performs to determine whether the subscriber should be marked for rerating and reguiding and to perform the actual marking.

### Receiving Subscriber Activity Events

The Pricing Engine receives Subscriber Activity events (as opposed to network events). The base Subscriber Activity event includes the following attributes:

- Event Type
- Event ID
- Bill Cycle Code
- Bill Cycle Month
- Bill Cycle Year
- Subscriber ID
- Customer ID
- Network Start Time
- Start Time
- Service Filter
- Activity Source
- Issue Date
- Target Cycle Code

Based on the above data, the Pricing Engine determines whether rerating is required.

### Handling Subscriber Activity Events

Subscription Manager invokes the Pricing Engine, which determines whether rerating or reguiding is required, according to:

- Definitions in Implementation Repository
- Contents of the event (in particular, the event's activity)
- Offers acquired by the customer
- Contents of the customer PIs

## Implementation Repository Definitions

Implementation Repository contains the definitions of subscriber activities and the conditions that determine whether rerating is required.

Subscriber activities that require rerating can be divided into the following two categories:

- Activities that require rerating or reguiding regardless of the customer's offers. Examples include:
  - Rerate activity – A Customer Management activity requesting that a subscriber be rerated. For details, see the “Customer Management Activities” section.
  - Backdated re-assignment of services to a different pay channel.
- Activities that require rerating/reguiding depending on the customer's offers. Examples include:
  - Subscriber Cancellation – If the subscriber was granted allowances that should be prorated
  - Offer Cancellation – If the offer includes allowances that should be prorated

### Offer-independent Activities Requiring Rerating

In cases where subscriber events with specific activity codes have been created by Customer Management, and have been communicated to Rater, the subscriber must be marked for rerating or reguiding.

For example, an market-level offer contains two pricing packages, each containing one pricing item. The two types of pricing packages include:

- Activities that require subscriber marking
- All subscriber activities

#### All Activities Requiring Subscriber Marking

This pricing package should have the highest priority. It contains a pricing item that captures all the subscriber activities that require subscriber marking. This pricing item contains:

- Qualification criteria
- Event handlers

The qualification criteria define all the subscriber activities that require marking for rerate or reguiding. The criteria include expressions such as:

```
Event Type = Subscriber Activity
((Event. Service Filter = "Service to PayChannel")
and
(Event.Issue_Date != Event.Effective_Date))
or
(Event.Service Filter = "Rerate")
or
(Event.Activity = .....)
```

The event handler that supports this functionality:

- Determines the target system (according to the Activity Code)
- Invokes the Mark Subscriber for Rerate extension function to update the data storage.

### **All Subscriber Activities**

This pricing package should have the lowest priority. It is qualified only when all other event qualification criteria fails. It is used to capture such events so that they are not classified as “non-qualified” and redirected to the Error data store. It contains one pricing item that captures all subscriber activities. This pricing item contains:

- Qualification criteria
- Event handlers
- Offer-independent activities requiring rerating

The qualification criteria define all the subscriber activities that require marking for rerate or reguiding. It includes expressions such as:

Event Type = Subscriber Activity

The event handler is a dummy handler, with no functionality.

## **Offer-dependent Activities Requiring Rerating**

### **Conditional Rerating**

Depending on the activity and the subscriber’s offers, certain subscriber events might require subscriber rerating; that is, conditional rerating is required.

An example of this is the Subscriber Cancellation activity. If the subscriber has an offer that includes allowances, then the subscriber requires rerating. This scenario may be refined by adding the following condition: if the subscriber has an offer that includes pro-ratable allowances, then the subscriber requires rerating. Further filters may be added, such as: if the subscriber has an offer that includes pro-ratable allowances, and the allowance granted exceeds the revised amount after prorating.

The complexity of the qualification for determining rerate subscriber marking defines the complexity of the implementation.

### **Pricing Package Support for Conditional Rerating**

To support conditional rerating, a PIT (with one pricing item) captures all the subscriber activities for conditional rerating that might require subscriber marking. This pricing item contains:

- Qualification criteria
- Event handler

The qualification criteria define all the subscriber activities that may require marking for rerate or reguiding. It includes expressions such as:

Event Type = Subscriber Activity

and

Event.Activity = “Cancel Subscriber”

The Event handler performs the Check and Mark Allowances routine, which checks whether the subscriber has allowances, and marks the subscriber for rerating.

If the subscriber participates in an agreement-level offer, then the customer is also marked for rerating.

## Handling Subscriber Events

Subscriber events are handled similarly to the regular network events:

1. The event is passed to the Pricing Engine as an event object.
2. The Pricing Engine receives this object as an input parameter through its API.
3. The customer object is loaded from the IMDB according to the customer identification attributes populated by Guiding.
4. Qualification of pricing packages and pricing items is performed according to priority.
5. Handlers provided by the qualified items are evaluated for the specific event type.
6. The event object is written to the IMDB for further offline synchronization with Oracle (if required).

## 5. PROCESSING SUBSCRIBERS FOR RERATING

---

All the subscribers to be reguided and rerated are stored in the Subscriber Rerate data storage table.

Once all the subscribers and customers to be reguided and rerated are known and recorded, the final sets of subscribers must be determined for:

- Reguiding
- Rerating

Subscribers that require reguiding are rerated immediately afterwards. Therefore, the reguiding and rerating lists do not overlap (include no common subscribers).

Two options for rerating events are discussed in this chapter:

- Rerating events after the bill cycle is locked
- Rerating events during the bill cycle

### Rerating Events After Bill Cycle is Locked

The first option is to reprocess the subscribers' past events in the requested bill cycle (that is, to rerate) after that bill cycle has been locked. This means that, while no additional events enter the locked cycle, existing events and accumulations can be updated. This is the standard mode for Rerating.

#### Advantages

There are two major advantages for waiting for the locking of the bill cycle prior to beginning the rerating:

- Event is rerated only once – If the bill cycle is closed for new events when rerating the subscriber, the events are not rerated again.
- The Rating process is not halted due to rerating – While rerating a subscriber, new events for the same subscriber do not require special handling. If the subscriber is currently being rerated, all new events are targeted to the next bill cycle month, as the period, which is being rerated for the subscriber, is locked for new events.

#### Disadvantages

The simplest solution is performing rerating at the end of the bill cycle. However, the overhead of processing a potentially large amount of events in a short timeframe may be too great.

## Rerating Events While Cycle Is Open

The second option is to rerate events during an open bill cycle. This mode can be selected manually at any time in the cycle through the Application Monitoring & Control (AMC).

### Advantages

The main advantage is that Billing would not have to be halted for a large volume of events that require rerating, as when the bill cycle is locked.

### Disadvantages

If rerating is performed during a bill cycle, events pertaining to the subscriber entering the system after rerating is completed may also require rerating. Furthermore, if a subscriber is currently being rerated, and new events for that subscriber enter the system for the period being rerated, they need to be deferred until the rerating of the subscriber is complete. This requires downtime of the Rater process.

## Recommended Solution

The recommended solution is to identify reasons for rerating where the probability of rerating the same subscriber again is minimal. An example is the Subscriber Cancellation activity. When subscribers cancel their subscriptions during a bill cycle, the probability of receiving additional events that trigger rerating is very low. Subscribers that fall into this category may be processed during the bill cycle. The rerating of the remaining subscribers would be processed at the end of the bill cycle.

The Rater processes of the bill cycle being rerated are paused during Rerating. This ensures that events received for subscribers that are rerated during the bill cycle, and are destined for the period currently being rerated, do not cause a conflict with the Rerated events.

## 6. EXTRACTING DATA FOR REGUIDING AND RERATING

---

This chapter covers the various extraction processes in preparation for reguiding and rerating.

### Subscriber Pre-rating Job

The Subscriber PreRer job builds the groups of subscribers that require reguiding and rerating. This process is an initial part of the pipelined process for rerating, which allows Billing to perform its processing on some groups simultaneously with rerating of other groups.

The job is run with the following parameters:

- Map\_key
- Partition\_ID
- Run\_mode
- Run\_ID
- Map\_mode

The Subscriber Prepare job performs the following operations:

1. Moves all subscribers that meet the input criteria from the Subscriber Rerate data storage table to the History Subscriber rerate database table.
2. Divides the subscribers into logical groups based on their subpartition and cycle. Each group is assigned a run\_id.
3. Sends a report to Billing containing the customer IDs to be rerated.

Subscribers with Customer IDs marked for rerating due to agreement-level plans are not included in this file, because all subscribers related to the marked customer are rerated by default.

### Subscriber Extract Job

The Subscriber Extract job builds the PI and Event files for rerating and reguiding. Once all subscribers have been assigned to the corresponding files (PI and Event), the process terminates.

The job is run with the following parameters:

- Map\_key
- Partition\_ID
- Run\_mode
- Run\_ID
- Map\_mode
- Mode:

- Cycle – Performs rerating of the marked population for the whole cycle
- Undo population – Performs rerating for specified customers
- List of Activity Sources:
  - Reason for subscriber marking, or
  - All

The Subscriber Extract job performs the following operations:

1. Depending on the target system (reason for marking) recorded in the Subscriber Rerate data storage table, either:
  - Adds the subscribers to be reguided to the appropriate Event file, and sends the file for reguiding. Each Event file has PI files associated with it. These PI files are sent to rerating upon completion of the reguiding associated with their Event file.
  - Adds the Event file of subscribers that need only to be rerated to the PI files, and sends them directly for rerating.
2. Sorts the above two files by Customer ID and Event Date/Time.

## Background Information

Once all the subscribers selected for reguiding and rerating have been extracted, it is required to extract all the data for the Reguiding and Rerating processes. The requirements for reguiding and rerating are:

- To “roll back” the system to the end of the previous bill cycle, and
- To process all events for the bill cycle

In order to “roll back” the system, the following activities are performed:

1. Extract the previous cycle months’ PIs to re-initialize them for the new cycle
2. Extract all rated events for the current bill cycle

## Previous Bill Cycle Rerate Extract

In order to perform the Rerate Extract process, the PIs for the previous bill cycle must be located in the Performance Indicator History database table. In some cases, in the early stages of the bill cycle, the PIs for the previous bill cycle may reside in the Performance Indicator tables (i.e., transient tables). The PI Archive process is used to transfer the PIs for the previous bill cycle to the Performance Indicator History database table.



There are two categories of PIs:

- Self-contained – These PIs have no effect on the future bill cycle. For example, Total Volume and Charges for a data session are measured in the context of a bill cycle, and the information is neither needed nor rolled over to the next bill cycle.
- Cross-cycle – These PIs have an effect on future bill cycles. For example, in the Allowance PI the balance is rolled over to the next bill cycle.

The Extract process connects to the history database, and extracts all PI information that has a “cross-cycle” attribute for the customers and subscribers found in the “Reguiding” and “Rerating” files.

A file, sorted by Customer ID, containing all PIs found in the history database tables, is created. This file is recorded in Audit & Control, ready for processing by Rerating.

### Cycle Rerate Extract

Usage data is retrieved from the Rated Event Oracle database. The following steps are performed:

- For the Reguiding customer or subscriber file:
  - a. All rated events for the subscribers are retrieved from the Rated Event database table into a Reguiding file.
  - b. The file is sorted by Customer ID and Network Event Date.
  - c. The file is recorded in Audit & Control as ready for processing by Acquisition & Formatting.
- For the Rerating customer or subscriber file:
  - a. All rated events for the customers/subscribers are retrieved from the Rated Event database table into the Rerating file, and the file is marked as “being rerated/reguided”.
  - b. The file is sorted by Customer ID and Network Event Date.
  - c. The file is recorded in Audit & Control as ready for processing by Rerating.



note

*The maximum number of records in each file is configurable, which may be used to improve performance. Performance can also be improved by running multiple parallel instances of the extract. It is recommended that you run separate extract jobs to meet the capacity requirements of the rerate and reguide processes.*



## 7. REGUIDING AND RERATING PROCESSES

---

This chapter describes the Reguiding and Rerating processes. The process flow is illustrated in Appendix D, “Rerating Map.”

### Reguiding



*This step is skipped if only rerating is required.*

Acquisition & Formatting receives the rated event files prepared by the Data Extract process. These records are formatted as regular records, and are guided to the relevant rater.

Acquisition & Formatting updates Audit & Control upon completion of the file, indicating that the target system is Rerating.

### Preparing Files for Rerating

Once Reguiding has completed guiding the rated events, these events are ready for rerating.

Rerating retrieves these files, along with the corresponding PI files, and begins processing.

### Rerating Flow

The general rerating flow is as follows:

1. The PI file is read. This file contains:
  - Headers for each customer/ or subscriber.
  - PI data required to initialize the PIs to their original states at the cycle's beginning (Cross-cycle PIs):
    - If the PI read is the first PI for the customer, all PI records for the customer are read and stored in memory.
    - All PI records are then sent to the Pricing Engine ReInitializePI API. The Pricing Engine constructs and initiates the PI records for the customer or subscriber in that bill cycle, ensuring that the data in the PI is in the same state as when the PI bill cycle was created.
2. When Rerating recognizes that the Customer ID has changed, the Rated Event file is read.
3. All rated events for this customer or subscriber are sent to the Pricing Engine in the same way as for regular rating.

4. Upon recognizing that all the rated events for the customer or subscriber have been processed, Rerating performs the following:
  - a. Sends all PI records to the Pricing Engine FinalizeCrossCyclePI API. This API determines whether any cross-cycle PIs were created during rerating of the current bill cycle. If so, it rewrites to the database any cross-cycle PIs that were deleted during the ReInitializePI step.
  - b. If rerating of the current bill cycle triggered the need for rerating the next bill cycle, runs the Mark Next process. This process marks the subscriber for rerating in the next cycle. This is further explained in Appendix B. Additional causes for rerating the next cycle are explained in Appendix C.

The PI file is then ready to process the next customer.
5. The Rerating process accesses the Rated\_Event table, which is subpartitioned. This enables simultaneous rerating of several files, where each process addresses a separate subpartition. When rerating is completed, and no more files are left to be rerated, a signal is sent to the PI Extract process and Event Extract to indicate the completion of rerating.
6. Once all the events have been processed and rerating completed, Audit & Control is updated with the completion status.

## Rerating Completion

Once all the available PI files have been processed by the rerating processes, the PostRer job is activated. This job:

- Ensures that all rerating processes terminated without errors
- Sends a report to Billing specifying subscribers that need to be further rerated.
- Sends a notification that the rerate process has concluded.



note

*In order to improve rerating performance, several instances can be run in parallel..*

## 8. RERATING DUE TO BILLING REQUEST

---

This chapter describes the continuation of the Rerating process life cycle after the billing cycle is closed and Billing takes control of the cycle for bill generation.

The Billing QA process might identify that a certain population requires recalculation. The recalculation may require rerating of all events associated with the selected customers.

### Billing – Preparation of Rerate Population File

The Billing Undo Preparation process creates text files containing a list of all the customers requiring rerating.

Once these files are prepared, the Billing process initiates the Rerating process in Customer mode.

### Rerating in Customer Mode

The Customer mode of Rerating is invoked by Billing, and is very similar to the Cycle mode. The following steps are performed:

1. The cycle period is opened for rerating.
2. The Prepare and Report process is invoked with the specific population data group, which it receives from the Audit & Control files (not from the Subscriber Rerate data storage table).
3. The Prepare and Report process retrieves the file from Audit & Control, and then moves all the customers included in the input file to the History Subscriber Rerate database table.
4. The Rerate Extract:
  - a. Extracts all the PIs from the previous month to a file.
  - b. Extracts all the rated events of the current month to a file.
  - c. Deletes the rated events from the database.
  - d. Closes the event and PI files, providing a special data group.
  - e. Sends the files for rerating via Audit & Control.
5. Rerating processes the files received from the Rerate Extract process.
6. The control of the cycle is passed back to Billing by sending Billing a message stating that rerating is complete.



amdocsrating

# **Rating 6.0 Maintenance Specification (Internal)**





# 1. INTRODUCTION

---

Amdocs Rating performs various operations for maintaining billing cycles, performance indicators, and usage databases.

## Purpose and Scope

This part of this book describes the Amdocs Rating operations for maintaining billing cycles, performance indicators, and usage tables.

## Main Functions

Amdocs Rating's principal maintenance functions for rating-oriented information are:

- Billing cycle control, including:
  - Maintaining Cycle Definition and Cycle State tables
  - Controlling the billing cycle's influence on Rating usage entities throughout the cycle's life cycle
- Partition control, including maintaining Partition Definition table
- Performance indicator (PI) maintenance, including:
  - Downloading PIs from the in-memory database table to the Oracle table
  - Rolling over cross-cycle PIs from the previous cycle to the current cycle
- Table truncation, including:
  - Backing up historical usage data
  - Truncating usage table partitions



## 2. CYCLE CONTROL

---

A billing cycle code identifies a group of customers billed on the same day of the month. The storage of usage data (rated events and performance indicators) is determined according to the customer's cycle code and run month.

Each cycle opens on a particular day of the month and closes on another day of the month. All events that occur prior to the cycle close date are included in that cycle's bill, whereas events that occur after the close date are billed in the following cycle.

The cycle state drives the maintenance of the Rated Event table, the PI table, and other data (described in the chapters that follow). This chapter describes the cycle tables and cycle operations that control the life cycle of cycles.

### Cycle Tables

The cycle tables contains cycle parameters, such as cycle dates and statuses.

#### Cycle Definition Table

The Cycle Definition table defines each cycle's parameters. When a new cycle is introduced, a new entry is added to the table manually.

The table's fields are:

- Cycle Code – A unique identifier of the cycle.
- Cycle Close Day – the day of the month on which the cycle is closed. All events occurring after this day are guided to the next instance of the cycle. For example, a monthly cycle that closes on the 15<sup>th</sup> of each month is a logical day that specifies a series of dates: 15 January v2005, 15 February 2005, 15 March 2005, and so on.



*If the specified date is the 31<sup>st</sup>, the cycle close date is defined as the last day of the month.*

- Frequency – A customer's bill cycle frequency can be specified in relation to either weeks or months:
  - W – Weekly, biweekly, or any other factor of a week
  - M – Monthly, bimonthly, quarterly, or any other factor of a month up to a year
- Frequency Multiplier – The frequency at which bills are produced can be specified for each Billing Arrangement as a multiple of the related customer's cycle frequency. Thus, the BA can be billed at a different frequency from the customer.

## Cycle State Table

The Cycle State table controls the life cycle of each cycle instance, which is one occurrence of a cycle code, cycle run month, and cycle run year. The New Cycle and New Cycle State jobs create an entry in the table for each cycle instance. (These jobs are described in the next section.)

The table's fields are:

- Cycle Code – The cycle's unique identifier.
- Partition ID – The partition unique identifier.
- Cycle Run Month – The month in which the cycle is closed.
- Cycle Run Year – The year in which the cycle is closed.
- Cycle Start Date – The date on which the cycle starts, which is the date after the close date of the previous cycle.
- Cycle Close Date – The date on which the cycle closes.
- Cycle State – Indicates whether the cycle is locked or open for new events. When the cycle is locked, new events are guided to the next instance of the cycle. The Pricing Engine refers to this field.
- Lock for New indicator – Indicates whether the cycle is locked or open for new events. This field's functionality is the same as the Cycle State field, but all applications other than the Pricing Engine refer to this field.
- Lock for Update indicator – When locked, cycle information cannot be updated, meaning that Rerating cannot run on the cycle. The cycle is locked for updates after the Billing Rerate and before the Events and PI extract. When the cycle is locked for updates, Billing controls the cycle.
- Archive indicator – This indicator is set after the cycle's PIs are downloaded from the IMDB to the Oracle history tables.
- Maintenance Status – Shows the cycle's data maintenance status for database administration purposes. The possible statuses are:
  - A (Available) – The cycle is available for queries, that is, the data exists in Oracle or the IMDB.
  - B (Backed up) – The data was backed up from Oracle to a tape or other backup device.
  - C (Cleaned) – The cycle month partition was truncated from the database. This takes data for the cycle offline, so it needs to be reloaded from the archive if access to the data is necessary.

## Partition Definition Table

The rating partition (a usage host in which the events are rated and stored) is independent of the cycle. Customers belonging to a particular cycle can be distributed across multiple partitions according to their Partition IDs. Rating maintains the Partition Definition table, which identifies the partition assigned to each customer.

The table includes these fields:

- Partition ID – The partition unique identifier

- Partition Type – Regular (R) or Outcollect (O)
- Description – Text description of the partition

The entire population of a billing cycle can be restricted to a single rating partition, if desired. This cycle compatibility mode enables maintaining billing cycles that were assigned to a single partition in earlier versions of Amdocs Rating.

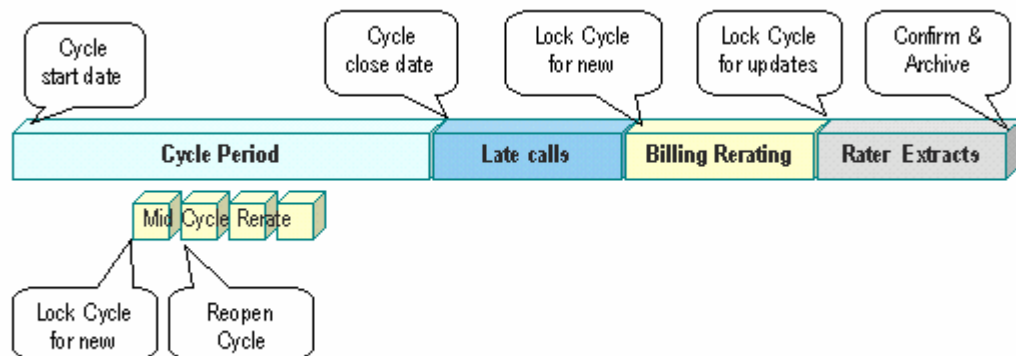
## Cycle Life Cycle

The billing cycle can be divided into these logical periods (each period is described in detail in the subsections below):

- Midcycle – The period between cycle's start and close dates.
- Midcycle rerating – A rerating process that runs during the cycle.
- Late calls – The period after the cycle close date and before Billing Rerating starts running. Late-arriving usage events can still be billed in this stage.
- Billing Rerating – A Rerating process that runs after the cycle is locked to new events.
- Rating extract – The period after the Billing Rerating run is completed, when PIs and rated events are extracted.
- Cycle confirmation – After Billing has successfully completed its run and is confirmed, and billing information is posted to Accounts Receivable.

The Cycle State table indicators are updated to open and close usage tables for new inserts or updates. The update is done at the start and end of each period by dedicated cycle operations that run as part of the operational maps of each process. For example, when Billing Rerating starts running, the Billing Rerating map activates the Lock Rerate operation to close the cycle to new events.

The figure below illustrates the life cycle of one billing cycle and its cycle operations:



**Figure 2-1: Billing Cycle Life Cycle**

The following sections describe the behavior of Rating in each period.

## **Midcycle**

All events that reach the system during this period are included in the current cycle's bill.

The Rated Event and Performance Indicator tables are open for updates and inserts in this period.

## **Midcycle Rerating**

A midcycle Rerating can be run on the current cycle. While the Rerating process runs, the current cycle's Rating process should not run.

The Midcycle Rerating map invokes the Lock Rerate job, which:

- Updates the Cycle State to Lock.
- Updates the Lock for New indicator to Yes.

At the end of the Rerating process, the Cycle Open operation reopens the cycle with the Reopen New run mode, and:

- Updates the Cycle State to Open.
- Updates the Lock for New indicator to No.

## **Late Calls**

The cycle is closed at midnight at the end of the cycle close date. New events that occur after midnight are guided to the new cycle. However, "late calls" (new events that occur before midnight but reach the system after midnight) are guided to the just-closed cycle until the cycle is locked.

The date change triggers the cycle closing and opening of the next cycle; no job is run.

## **Billing Rerating**

The Billing Rerating map locks the closed cycle to new events. It executes the Lock Rerate cycle operation, which:

- Updates the Cycle State to Lock.
- Updates the Lock for New indicator to Yes.

From this point, the cycle cannot be reopened for inserting new events.

## **Rating Extract**

The Rated Events and PI extracts are run after Rerating, to ensure that the extracted data is accurate. The extracts can select information for a Billing Arrangement from more than one cycle instance.

The Lock Bill cycle operation is activated, which Updates the Lock for Update indicator to Yes.

If a correction to usage is required and Rerating should run, the cycle is reopened for updates. The Cycle Open operation is invoked in the Reopen run mode, which updates the Lock for Update indicator to No.

## Cycle Confirmation

The bill cycle confirmation process seals the bill. The cycle's Performance Indicator table can be downloaded from memory to the Oracle history tables (when the table is in memory rather than in Oracle). The Archive indicator is set to Yes when the PIs are loaded to the Oracle PI table (for more details, see "Performance Indicator Maintenance").

## Cycle Operations

The operations, or jobs, that update the Cycle State table are described in the subsections that follow.

### Cycle Open

The Cycle Open job operates in two run modes:

- **Reopen New** – Opens the cycle for insertion of new records. The job is operated from the Midcycle Rerating map after Midcycle Rerating finishes.  
In this mode, the Lock for New indicator is set to No and the Cycle State attribute is set to Open.
- **Reopen** – Opens the cycle for updates only. This job runs before Billing Rerating is rerun. This is necessary when Billing Rerating already has run once and the cycle has been closed for Rerating, but errors are discovered during Billing Quality Assurance that require running Billing Rerating again.

In this mode, the Lock for Updates indicator is set to No.

### Lock Rerate (Lock for New)

This job locks the cycle to new events. It sets the Lock for New indicator to Yes and the Cycle State attribute to Lock. The cycle still can be updated by Rerating.

### Lock Bill (Lock for Update)

This job locks the cycle to updates, and prevents Rerating from running, as well as preventing new records from being inserted. It sets the Lock for Update indicator to Yes. (At this point, the Cycle State already is Locked and the Lock for New indicator already is set to Yes.)





## 3. PERFORMANCE INDICATOR MAINTENANCE PROCESS

---

This chapter describes the PI Maintenance process, which is responsible for:

- Copying PIs from the IMDB to the Oracle database after bill confirmation
- Creating new PIs in the current cycle for cross-cycle PIs (if not already created)

### General

The PI Maintenance process performs three operations regarding the Performance Indicator table in the IMDB. According to flags defined in the configuration file, the process:

- Archives – Reads PI information from the Performance Indicator usage table in the IMDB and loads it into the Performance Indicator table in Oracle.
- Purges – Cleans unneeded memory by removing records or setting columns holding dynamic data to null. Note that the presence of the “reconstruct” verb enables the processing of cross-cycle PIs in memory.
- Reconstructs cross-cycle PIs – Copies the values of cross-cycle PIs from the previous cycle to the new cycle and invokes initialization handlers.

The process is subject to these restrictions:

- Archive mode – A month should not be archived unless the preceding month already has been archived. The usage database must be in the IMDB.
- Archive and Purge – These tasks are mutually exclusive. While the cycle month is being archived, the Cycle State archived indicator directs Customer Management queries to the IMDB. Since the archiving process might fail before completion, purging while archiving could result in part of the cycle data being sent to the IMDB and part to Oracle. The two operations therefore are not performed concurrently.
- Reconstruct – A month should not be reconstructed unless the preceding month already has been archived, because reconstruction can delete or overwrite the preceding month’s data.
- Purge – A month should not be purged unless that month already has been archived.

Rating supports the Product Catalog’s global flag for PI attributes, so the Pricing Engine can distinguish between “global” and “partitioned” PI attributes. “Global” attributes are attributes that exist once in a partitioned PI, whereas “partitioned” attributes have a different instance in every partition. (“Partitioned” is the default value.)

## Reconstructing Performance Indicators

Cross-cycle PIs (also known as rolling PIs) pass usage information from one cycle to the next cycle. For example, the remaining amount of a rolling allowance from a previous cycle's PI is rolled over to the PI for the current cycle's allowance. (Cross-cycle PIs are defined in the Product Catalog.)

The Reconstruct operation of the PI Maintenance process handles cross-cycle PIs. The process can be run at any time after the cycle close date and before the next cycle is closed.

In the regular Rating process, the Pricing Engine identifies all the PIs that influence or are affected by an event it receives. If a PI record does not exist in a particular cycle, the Pricing Engine creates a record for the PI. In case of a cross-cycle PI, the Pricing Engine:

1. Retrieves the PI from the previous cycle.
2. Creates a new record with the same data but a different cycle ID (using the new cycle's run month and year).
3. Activates initialization handlers of the PIT that relates to the PI.

If no events occurred for a subscriber during the current cycle, the PI Maintenance Reconstruct operation creates the cross-cycle PIs instead of Rating, using the same Pricing Engine modules. This ensures that the correct information is displayed on the bill, and no usage information is lost if the subscriber does not generate any events during some cycles.

## PI Maintenance Process Description

This subsection describes the parameters and flow of the PI Maintenance process.

### Configuration File Parameters

The parameters of the configuration file are:

- Cycle code
- Cycle month
- Cycle year
- Run mode:
  - Archive (true or false)
  - Purge (true or false)
  - Reconstruct (true or false)
- Commit parameters (for Archive mode):
  - Maximum number of records in file
  - Records to commit

## High-Level Flow

The main flow of the PI Maintenance process includes these steps:

1. Read PI Maintenance configuration data and perform the validity checks according to the process restrictions.
2. Create query parameters (PI, cycle code, month, year, etc.).
3. Using the Query API, retrieve the PI records for the given parameters.
4. While fetching the PI entity:
  - In Archive mode:
    - a. Insert the entity to the Oracle connection.
    - b. If the insert fails, try to perform an update in Oracle.
    - c. If the limit on the number of records per file is reached, commit the changes to Oracle.
  - In Purge or Reconstruct mode:
    - a. Call the Rater API to purge.
    - b. If the limit on the number of transactions to commit is reached, commit the changes to Oracle.
5. Update the cycle state of the current cycle to Archive mode (by changing the Archive indicator to Y).



## 4. TRUNCATE CYCLE PROCESS

---

Billed cycles are kept in the Rating usage data storage area for a certain period after billing, for use with online queries and adjustments. The amount of information is huge, however, and consumes a vast amount of disk space. Therefore, after a certain number of bill cycles, the history cycles are backed up and deleted from the Rating tables.

This chapter describes the Truncate Cycle process, which truncates history partitions from the Rated Event, Rejected Event, Performance Indicator, and Subscriber Rerate usage tables in Oracle.

### Usage Partitioning Background

The usage tables are huge because of the large numbers of events rated. To maintain this amount of data, the tables are divided into smaller table areas, according to criteria defining the basis of the division. The natural division criterion is the Cycle ID, which is the cycle code for a specific cycle month and year.

The Oracle partitioning mechanism is used to divide the usage tables into smaller data volumes. This mechanism permits:

- Placement of Oracle tables and indexes in multiple table spaces
- Reduction of maintenance by Operations on high-volume tables, because:
  - It operates on an individual partition rather than on the entire table or index
  - Maintenance can be performed concurrently on different partitions

The Subpartition ID is based on the cycle code and cycle month; therefore, 12 partitions are created for each cycle code. (The Rated Event table also is partitioned within the cycle.) The population of a billing cycle can span more than one partition. Therefore, Rating maintenance jobs can run across multiple partitions.

### Number of History Cycles To Keep

In general, storage of a three-month history (three billed cycles) is recommended, with earlier cycles being backed up and their partitions truncated. In this case, five partitions per cycle are active: the current cycle, the next cycle, and three previous cycles.

However, the CSP may decide to retain more history partitions. Since the maximum number of cycle partitions is 12, no more than 10 history cycles are allowed (plus the current and next cycles).

### Truncate Cycle Process Flow

This section describes the Truncate Cycle process's parameters and flow.

## **Input Parameters**

The input parameters of the Truncate Cycle process are:

- Cycle code
- Cycle month
- Cycle year

## **Process Flow**

The Truncate Cycle process follows this flow:

1. Connect to the Oracle database.
2. Check the Maintenance Status field in the Cycle State table. The truncation is performed only if the Maintenance Status indicator has been set to B (backed up) by the data administrator who backed up the partition.
3. Truncate the Rated Event, Rejected Event, Performance Indicator, and Subscriber Rerate tables according to the cycle code and cycle month.
4. Update the Maintenance Status indicator to C (clean).

amdocsrating

# **Rating 6.0 Recovery Specification (Internal)**





# 1. INTRODUCTION

---

Amdocs Rating Recovery provides a mechanism for restoring the Rating system to a valid state, from which it can resume processing after an unexpected shutdown.

## Recovery Overview

When Amdocs Rating receives a shutdown request from an external source and terminates smoothly, it continues performing the current operation and then performs a durable commit on all processed transactions. That is, on receipt of a shutdown request, Rating shuts down after completing processing of the file that it is currently working on.

Unfortunately, Rating does not exist in a sterile environment, and it cannot always guarantee that the process ends smoothly. If Rating terminates abruptly for any reason, a file may be undergoing processing. Some events may yet need to be processed.

The Rating Recovery mechanism ensures that the system is restored to a valid state, that is, it updates the system with all information required by Rating to reprocess the aborted file correctly.

During the Rating process, the Recovery mechanism updates a table that holds the last processed event in the file. The updates occur when the Rater commits the results to the database. When recovering from a system crash, the utility assists Rating in:

- Retrieving the last event processed in the recoverable file from the table
- Starting processing from the point of failure (skipping all already processed events)

## Purpose and Scope

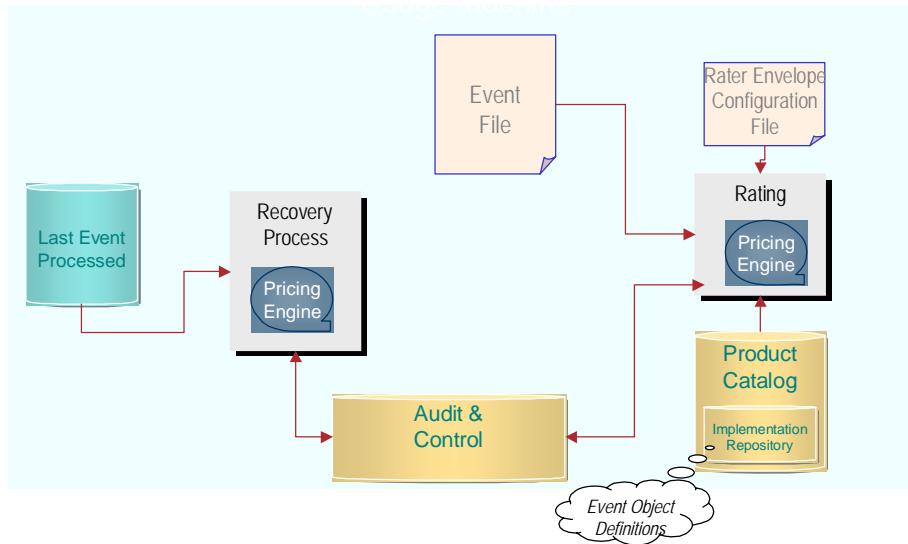
This part of this book provides a functional and technical description of the Amdocs Rating Recovery mechanism. It details how Rating recovers from an unexpected shutdown.

## Interaction with Other Components

The Rating Recovery mechanism interfaces with the Audit & Control module when determining what files need to be recovered and processed.

## Process Flow

The Rating Recovery process flow is depicted in the following diagram.



**Figure 1-1: Rating Recovery Flow**

The following chapters provide more detailed descriptions of the Rating Recovery process flow.

## 2. RATING FLOW

---

This chapter provides a brief overview of the Rating process and reviews the recovery methodology, to better explain the Recovery mechanism.

Normally, the Rating process performs the following actions:

1. Receives files from Audit & Control.
2. Marks the files In Use.
3. Loads the files to memory and builds buckets of events.
4. Loops on all the events in each bucket:
  - a. Reads each event.
  - b. Updates the Performance Indicator table in TimesTen or Oracle.
  - c. Updates the Rated Events and Rejected Events tables in TimesTen.
  - d. Performs a nondurable commit.
5. Verifies that all events have been processed.
6. Performs a durable commit in TimesTen.
7. Updates Audit & Control to indicate that the files has been processed.

Dedicated processes (Replicators) retrieve rated events or rejected events from the TimesTen logs and load them into the Oracle database.



## 3. PREPARATION FOR RECOVERY

---

This chapter describes changes to the Rating flow to support the Recovery mechanism.

### Recovery Methodology

Rating may terminate unexpectedly (crash) at any point in its flow. If some events have been processed, the files containing the events cannot be sent again for reprocessing, because already-processed events have affected the performance indicators, and reprocessing would duplicate the effect on the PIs. However, the file might contain unprocessed events that, if disregarded, may result in a loss of revenue.

The Recovery process, therefore, is responsible for identifying unprocessed events that affect the PIs, and assuring that these events are fully processed.

However, the Recovery process cannot identify the last processed event based on the data in Oracle or in TimesTen.

### Rating Flow – Preparing the Data for Recovery

Since the Recovery process cannot rely on the tables to identify the last processed event, Rating must record the last event it processed for the Recovery process.

Therefore, the Rating flow described in the preceding chapter incorporates additional steps (indicated below in *italics*) to record the last event processed.

Rating:

1. Receives a files from Audit & Control.
2. Marks the file as In Use.
3. Loads the files to memory and builds buckets of events.
4. *Inserts a record into the Postpaid Recovery (RPR1\_POSTPAID\_RECOVERY) table containing the file, the bucket and process details.*
5. Loops on all the events in each bucket:
  - a. Reads each event.
  - b. Updates the PIs in TimesTen or Oracle.
  - c. *Updates the Postpaid Recovery table with the record counts of both successfully handled and rejected events for each bucket, as well as additional auditing information (only in TimesTen mode).*
  - d. Performs a non-durable commit (only in TimesTen mode).
6. Verifies that all the events in the bucket have been processed.
7. *Deletes the record in Postpaid Recovery table with the bucket details.*

8. Performs a durable commit in TimesTen.

9. Updates Audit & Control to indicate that the file has been processed.

The additional Rating steps enable the Recovery mechanism to identify the last processed event in a file and bucket if Rating exits abnormally. The last event recorded in the table is the last event that affected the Performance Indicator table.

## 4. RECOVERY PROCESS

---

This section describes the Recovery process, which an administrator invokes manually. The administrator recognizes which files require recovery, and provides the Recovery process with their file identifiers.

If multiple files are left in an aborted state, the Recovery process receives the ID of the partition on which to perform recovery.

### Recovery Flow

For single or multiple files, the Recovery process:

1. Gets files that are In Use in Audit & Control
2. Loads the files to memory and builds buckets of events.
3. Deletes all previously processed events from the buckets.
4. Loops on all the remaining events in the buckets:
  - a. Reads each event.
  - b. Updates the Performance Indicator table in TimesTen or Oracle.
  - c. Updates the Rated Events and Rejected Events tables in TimesTen.
  - d. Performs a nondurable commit.
5. Verifies that all events have been processed.
6. Performs a durable commit in TimesTen.
7. Updates Audit & Control to indicate that the files has been processed.



note

*The same recovery flow is used in both Oracle and TimesTen mode.*





## 5. POSTPAID RECOVERY TABLE

---

The Rating Recovery mechanism uses the Postpaid Recovery (RPR1\_POSTPAID\_RECOVERY) table for recovering groups of postpaid events. It inserts an entry into the table for each bucket in a transaction. This entry is updated for each event processed using the save point mechanism and is deleted when all transactions are completed. Rating creates one entry for each A&C file (or bucket in many to many mode) in a transaction to identify the file as part of a group of files. Entries inserted by an acceptor will be removed by the Transaction Manager. If a failure occurs before the group file is created, then there will be no entries in the Postpaid Recovery table. If the failure occurs during processing of a group of files, the files will stay in In Use (IU) status. (the original file in many to many mode will be in status CO"). The postpaid recovery mechanism fetches the group files in IU status and retrieves the information for each one of the files from the Postpaid Recovery table. Any files that are not part of a group, and which are in IU status are also retrieved but are processed as usual since no recovery information exists for them.

The table contains the following information:

- Audit & Control File details – These include the data required to read the file:
  - IDENTIFIER - The unique identifier of the file in Audit & Control
  - FILE\_NAME - The location of the corrupt file
  - FILE\_PATH - A&C events input file path
- Rated Event Temporary Files – In an Oracle environment, the previously processed rated events are written to output files:
  - OUTPUT\_FILE - Output file name for the dispatcher
  - OUTPUT\_FILE\_PATH - Output file path for the dispatcher
- Process Details – These include the process that processed the file:
  - CYCLE\_CODE - The bill cycle code of the corrupt file (this allows the recovery of all files related to a specific bill cycle)
  - TASK\_ID - Partition ID
  - SUBTASK\_ID - Subpartition ID (unique ID to differentiate between Raters working on the same partition)
  - BUCKET\_ID - Bucket ID (each bucket may contain more than one event)
  - TOTAL\_BUCKETS\_NUM - Number of buckets in one transaction
- Last Event Processed Details:
  - NUMBER\_SUCCESS - The number of events successfully processed in the previous run

- NUMBER\_REJECTS - The number of events rejected in the previous run
- NUMBER\_GENERATED – The number of generated events (dispatched events)
- NUMBER\_DROPPED – The number of dropped events (events which were processed but with no output to the Rated Event table)
- AUDIT\_MONEY - The money value of the events successfully processed in the previous run
- AUDIT\_DURATION - The audited duration value of the events successfully processed in the previous run

amdocsrating

# **Rating 6.0 Usage Queries Specification (Internal)**



# 1. INTRODUCTION

---

This part of this book describes the Amdocs Usage Queries system.

## Overview

The Usage Queries system is the tool provided by the Amdocs Pricing Engine for viewing the customer's usage records after they have been rated by Amdocs Rating. Usage Queries offers access to two types of rated information:

- Event – The basic data element that represents usage of telecommunications network services. Events are received from the network through various machines and servers, and processed during the event processing flow by Acquisition & Formatting and Rating.
- Performance Indicator (PI) – Rated events accumulated into one performance indicator (accumulated usage counter), according to predefined qualification criteria and handling methods.

This information can be retrieved for either billed or unbilled information:

- Billed data – The Billing process has been confirmed and the bill is considered sent to the customer.
- Unbilled information – The Billing process has not been run or confirmed.

From this classification, it follows that the basic unit of data that can be defined as billed or unbilled is the cycle. Indeed, the cycle period is the basic unit of information containing all its rating data (PIs and events), and it is stored as one unit, billed or unbilled, in a data storage device. Due to the large volume of data, cycles are often maintained in different data storages (e.g., database table partitions) and on different hosts (e.g., different remote UNIX servers). Amdocs Rating maintains current PI information either in an in-memory database (TimesTen), or in a relational database (Oracle), while the previous cycles' PIs are maintained in a relational database.

Amdocs Usage Queries provide answers to queries about customer usage in a simple manner, encapsulating this complex structure. They offer a simple interface for viewing usage, while incorporating all the parameters mentioned above (billed or unbilled, data storages, remote hosts, etc.).

The Usage Queries system can be used by:

- Call center customer service representatives (CSRs), for viewing a customer's accounts
- Customers of the communications service provider (CSP), using a self-service portal to view their accounts over the Internet
- Interactive Voice Response (IVR) machines, providing the customer with information for a specific PI (e.g., the number of free minutes used in the current cycle)

Usage Queries are designed as Application Program Interfaces (APIs) based on Enterprise Java Beans (EJBs). As a layer on top of these APIs, the Amdocs Web-enabled client (JSP based) reads the information and displays it conveniently.

## Architectural Design Standards

Usage Queries was designed under these architectural principles:

- EJB APIs comply with Amdocs API standards.
- APIs are stateless, to reduce resource allocation when high volumes of data are queried by large numbers of users.
- Results are formatted flexibly, according to the implementation for each CSP. Whenever a new event or PI data structure or format is introduced, no additional development is required in the API layer or in the presentation layer (JSP client).
- Events are defined in the Product Catalog only, and not in the API area.
- Various types of deployments and configurations are supported, such as distributed hosts and different data storage methods (Oracle and TimesTen).

## Purpose and Scope

This part of this book describes the Amdocs solution for Usage Queries. It describes both the presentation layer and the API layer. It addresses both business and architectural aspects of Usage Queries.

## 2. USAGE QUERY DESCRIPTION

---

This chapter provides a comprehensive description of the Usage Queries system, including the business logic involved in querying events and performance indicators.

### Query Flow

This section describes the flow for querying rated events and PIs. The Amdocs front end (FE) implements the flow described here. Third-party systems (applications that are designed as clients of the APIs) also can implement this flow.

#### Rated Events

To query and view rated events:

1. Select View Rated Events from the Subscriber folder.  
The Available Cycles list is displayed (through the Billing API).
2. Specify the desired date range.
3. Select the event type, if desired. If none is selected, the default is “All Events”.
4. Invoke the query.  
A result page displays a common view for all events. The view is derived from the definitions in the Product Catalog.
5. Use the pagination mechanism to navigate among the result pages.
6. Select Zoom Event to view detailed information about a selected event (through the Get Event Details API).

#### Performance Indicators

To query and view PIs:

1. Select View Performance Indicators from the Subscriber or Agreement folder.
2. Specify the desired cycle month and year.
3. Invoke the query.  
A result page displays a common view for all PIs. The view is derived from the definitions in the Product Catalog.
4. Use the pagination mechanism to navigate among the result pages.
5. Select Zoom PI to view detailed information about a selected PI (through the Get PI Details API).

## Query Definition

This section provides a detailed definition of each of the queries. Although the input is well-defined, the output is flexible and can be customized during implementation.

### Rated Event List

#### Input

- `customerIdInfo` (mandatory)
  - *customerId* (mandatory) – The customer identifier.
  - *partitionId* (optional) – The partition identifier. This field is mandatory if Usage Queries machines are installed in customer partition mode, and optional if Usage Queries machines are installed with the old cycle mode.
- `baIdInfo` (optional)
  - *baId* (mandatory) – Identifies the billing arrangement.
- `subscriberIdInfo` (optional)
  - *subscriberId* (mandatory) – The subscriber identifier.
- `flexibleCycleInfo` (mandatory)
  - *cycleCode* (mandatory) – The cycle code from the reference database.
  - *cycleYear* (mandatory) – The year in which the cycle end date occurs.
  - *cycleInstance* (mandatory) – Identifies the serial occurrence of the cycle instance in a calendar year.
- `eventTypeInfo` (optional)
  - *type* (mandatory) – The event type.
- `dateRangeInfo` (optional)
  - *startDateTime* (optional) – Represents the start date.
  - *endDateTime* (optional) – Represents the end date.
- `externalStructureInfo` (optional)
  - *externalstructure* (optional) – List of the external structure.
- `additionalQueryParameters` (optional)
  - *parameter1* (optional) – Additional parameter used for customization purposes.
  - *parameter2* (optional) – Additional parameter used for customization purposes.
  - *parameter3* (optional) – Additional parameter used for customization purposes.
  - *parameter4* (optional) – Additional parameter used for customization purposes.
- `paginationInfo` (mandatory)
  - *pageSize* (mandatory) – Number of records in a page.



- *pageNumber* (mandatory) – The required/retrieved page number.

### Output

- Event list (limited by page size)
- Pagination information (*has more pages* indication)

## PI List

### Input

- *customerIdInfo* (mandatory)
  - *customerId* (mandatory) – The customer identifier.
  - *partitionId* (optional) – The partition identifier. This field is mandatory if Usage Queries machines are installed in customer partition mode, and optional if Usage Queries machines are installed with the old cycle mode.
- *agreementIdInfo* (optional)
  - *agreementId* (mandatory) – The agreement identifier
- *flexibleCycleInfo* (mandatory)
  - *cycleCode* (mandatory) – The cycle code from the reference database.
  - *cycleYear* (mandatory) – The year in which the cycle end date occurs.
  - *cycleInstance* (mandatory) – Identifies the serial occurrence of the cycle instance in a calendar year.
- *piCategoryInfo* – Cycle information (optional)
  - *category* (optional) – The PI type.
- *externalStructuresInfo* – Cycle information (optional)
  - *externalstructure* (optional) – List of the external structure.
- *additionalQueryParameters* – Cycle information (optional)
  - *parameter1* (optional) – Additional parameter used for customization purposes.
  - *parameter2* (optional) – Additional parameter used for customization purposes.
  - *parameter3* (optional) – Additional parameter used for customization purposes.
  - *parameter4* (optional) – Additional parameter used for customization purposes.
- *paginationInfo* – Cycle information (mandatory)
  - *pageSize* (mandatory) – Number of records in a page.
  - *pageNumber* (mandatory) – The required/retrieved page number.

### Output

- PI list (limited by page size)
- Pagination information (*has more pages* indication)

## Event Details

### Input

- `customerIdInfo` (mandatory)
  - *customerId* (mandatory) – The customer identifier.
  - *partitionId* (optional) – The partition identifier. This field is mandatory if Usage Queries machines are installed in customer partition mode, and optional if Usage Queries machines are installed with the old cycle mode.
- `subscriberIdInfo` (optional)
  - *subscriberId* (mandatory) – The subscriber identifier.
- `eventIdInfo` (mandatory)
  - *eventId* (mandatory) – The event identifier.
- `flexibleCycleInfo` (mandatory)
  - *cycleCode* (mandatory) – The cycle code from the reference database.
  - *cycleYear* (mandatory) – The year in which the cycle end date occurs.
  - *cycleInstance* (mandatory) – Identifies the serial occurrence of the cycle instance in a calendar year.
- `eventTypeInfo` (mandatory)
  - *type* (mandatory) – The event type.
- `externalStructuresInfo` (optional)
  - *externalstructure* (optional) – List of the external structure.

### Output

- Event details

## PI Details

### Input

- `customerIdInfo` (mandatory)
  - *customerId* (mandatory) – The customer identifier.
  - *partitionId* (optional) – The partition identifier. This field is mandatory if Usage Queries machines are installed in customer partition mode, and optional if Usage Queries machines are installed with the old cycle mode.
- `piCategoryInfo` (optional)
  - *category* (optional) – The PI type.
- `performanceIndicatorIdInfo` (mandatory)
  - *performanceIndicatorId* (mandatory) – The performance indicator identifier.
  - *cycleInfo* (optional) – Information about the cycle.
  - *flexibleCycleInfo* (optional) – Information about the cycle.

- *agreementId* (mandatory) – The agreement identifier.
- *offerInstance* (mandatory) – The offer instance identifier.
- *externalStructuresInfo* (optional)
  - *externalstructure* (optional) – List of the external structure.
- *paginationInfo* (mandatory)
  - *pageSize* (mandatory) – Number of records in a page.
  - *pageNumber* (mandatory) – The required/retrieved page number.

### Output

- PI details (limited by page size)
- Pagination information (*has more pages* indication)

## Technical Overview

The Rating Usage APIs are developed as EJB components. They can be deployed either on a separate EJB container or on the same container as other Amdocs core components, such as Customer Management and Replenishment Management. The EJB APIs use the Pricing Engine core APIs (written in C++) to retrieve the required information. The EJB layer and the Pricing Engine communicate through a mediation server, implemented as an RMI (Remote Method Invocation) server that runs on the usage hosts (the same hosts where the Rating processes run). The EJB server returns the results to the client in XML format. The client then parses it and formats it in the requested presentation style, and presents the information to the end user.

The Pricing Engine APIs are used to receive an *EntityCursor* object, which provides iteration operations. The objects returned by the *EntityCursor* are *ExternalRecord* instances defined during the implementation stage. These external records are located in the Product Catalog Implementation Repository and represent the desired structure needed by the PI and event queries. A serialization method provides the ability to transform the *ExternalRecord* to an output stream in XML format. This serialized string is stored in the EJB event or PI data type, along with common members that identify the event or PI. The APIs return this result data type to the client, which parses the data and presents it.

More details can be found in the “Usage Query Architecture” chapter.

## Implementation Repository Definition

During the implementation process, external records should be defined in the Product Catalog Implementation Repository and mapped to the various event and PI entity objects. Defining these structures immediately impacts the view that is returned by the Usage Query APIs and displayed by the Amdocs front end.

## Common Event Members (Header)

The following members are required attributes and assumed to be associated with every event regardless of its type:

- Event ID
- Event Type
- Event Start Date and Time
- Subscriber ID
- Customer ID

## Common PI Members (Header)

The following members are required attributes and assumed to be associated with every PI regardless of its type:

- Cycle information (month, year, and code)
- Item ID
- Agreement ID
- Customer ID
- Offer Instance ID

## External Record

The external records mechanism is utilized to extract the required attributes for each event or PI query during the implementation process.

An external record is defined in PC and mapped to an entity object (PI or Event). The mapping rules allow for:

- Creating a simple mapping (attribute to attribute).
- Creating complex, dimension-based mapping rules. Applying such rules result in multiple external records per entity.
- Creating filter rules. Applying these rules results in entity filtering.

## Linking Between Query and External Records

The system enables the linking of external records to an online query. The linkage is performed according to the following attributes:

- Query Name – event list, event details, PI list, or PI details.
- Link Subtype – Used to distinguish between different event and PI types. If an exact match is not found, a default link is used.

This linkage is maintained in a reference entity. It is part of the implementation team's responsibility to maintain these links.

## Examples of External Records

The following examples illustrate how an external Event View record is created and mapped to a Billable Call that is the ancestor of all billable events.

### Event Online View External Record

The Event Online View record is used for displaying a common view for all events of all types.

Based on:

**Attributes**

	Name	Type
▶	Event Type	Event type
	Event ID	Event ID
	Resource Type	Resource type
	Resource ID	Resource ID
	Start Time	Date
	Rated amount	UK currency
	Discount Amount	UK currency
	Attribute #9	Numeric
*		

**Description:**

**Attribute properties**

General

Dimensions: **None**  
Kind: Implementation  
Default value:

Misc

ReadOnly: False

PC properties

Attribute category na:

**ReadOnly**

**Figure 2-1: Event Online View External Record**

### GSM Voice Call Online View External Record

The Voice Call Online View record is based on the Event Online View record and is used when a GSM voice call event is queried.

Based on: Event Online View Go

Attributes

Name	Type
Resource ID	Resource ID
Start Time	Date
Rated amount	UK currency
Discount Amount	UK currency
Attribute #12	Numeric
IMSI	IMSI
Call Source	Call source
Call Destinaion	Call destination
Calling number	Telephone number
Cell ID	Cell ID
Location area	Location area

Description:

Attribute properties

General

Dimensions	None
Kind	Implementation
Default value	

Misc

ReadOnly	False
----------	-------

PC properties

Attribute category na	
-----------------------	--

ReadOnly

Figure 2-2: GSM Voice Call Online View External Record

Applying Mapping Rules

The Billable Event is mapped to the Event Online View external records.

The following figure displays the event definition screen.

Based on:  Go

### Attributes

Name	Type
Event type	Event type
Event ID	Event ID
Guiding start time	Date
Resource type	Resource type
Resource ID	Resource ID
Service filter	Service filter
Start time	Date
Subscriber ID	Agreement ID
Account	Account
Billing arrangement	Billing arrangement
Charge amount	UK currency
Charge including fre	UK currency
Discount amount	UK currency

**Description:**

### Attribute properties

General	
Dimensions	None
Kind	Core
Default value	
Misc	
ReadOnly	True

**ReadOnly**

**Figure 2-3: Billable Event Mapping Rules**

The GSM Voice Base event is mapped to the GSM Voice Base Online View external record.

When the Get Event List query is invoked with no specific event type filter request, a common event list is presented by the client, based on the attributes defined for the Event Online View external record. When selecting a GSM Voice event and invoking the Get Event Details API, the information retrieved and displayed is based on the GSM Voice Online View external record.

## API Definition

The next step after defining external records and mapping cases is to map the External Record names to the API name in the API configuration repository (implemented as a reference table) – for example, to map the Event Online View record to the Get Event List API. More than one external record can be linked to an API name.

The API objects (data types) called EventDetailsInfo and PIDetailsInfo are used to represent events or PIs. Specific members are defined to contain the common header (common to every event or PI) and a String data member containing all the event details serialized in XML format.

Event APIs use the data member as a basis for presentation. The client may invoke parsing logic on the returned string to present it as required.

The Amdocs front end uses a simple parsing logic, assuming the data is presented as serialized by the serialization method (e.g., if the field *Capture Date*=02/02/2005 is written to the XML string by the serialization process, it will be presented in that format by the client).

### XSD Layout

The EntityDataSequence XSD represents a common External Structure XML format when serialized.



EntityDataSequence.xsd

### XML Sample

The following is a sample of the XML data.

```
<?xml version="1.0" encoding="UTF-8"?>
<EntityData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="EntityData.xsd">
  <Event type="Voice call">
    <Attribute name="Subscriber ID" basicType="Numeric">
      <Value value="1000030"/>
    </Attribute>
    <Attribute name="Event ID" basicType="Numeric">
      <Value value="12345678901234567"/>
    </Attribute>
    <Attribute name="Start time" basicType="DateTime">
      <Value value="2001-09-22 19:50:00.000"/>
    </Attribute>
    <Attribute name="Cycle code" basicType="Numeric">
      <Value value="3"/>
    </Attribute>
    <Attribute name="Cycle month" basicType="Numeric">
      <Value value="9"/>
    </Attribute>
    <Attribute name="Cycle year" basicType="Numeric">
      <Value value="2001"/>
    </Attribute>
    <Attribute name="Duration" basicType="Numeric" uom="Seconds">
      <Value value="231"/>
    </Attribute>
  </Event>
</EntityData>
```

## Billing Cycle API

An Amdocs front-end client, or any other client that implements the query flow described above, must use an API to retrieve the applicable cycles per customer. Following is the definition of such an API.

### Get Cycle List

#### Input

- Agreement ID or Subscriber ID, or



- Billing Arrangement ID (optional; if not provided, all the account's data are retrieved)

### Output

- Array of:
  - Cycle code
  - Cycle month
  - Cycle year
  - Status (billed or unbilled)

The API provides a list of applicable cycles per customer. If a cycle change occurs during the customer life cycle, all the related cycles are returned.

## Amdocs Front End Client

The Amdocs front end client supports the flows described above. The client uses the APIs to retrieve the information.

The XML file is parsed and displayed as written in the XML document. Currently, no additional processing is done on the attributes. The client may impose any formatting rules required. These rules can be associated with the attribute types, which are published among other information in the XML document retrieved by the queries (external record).

## Rated Events

This section describes the Amdocs front end for queries of rated events.

### Description

When Usage Queries are entered from the Subscriber folder, the Subscriber is specified. The list of billed documents is supplied by the Billing API for the Billing Arrangement. The Billing Arrangement contains the Pay Channel of the owner that is mapped to the Primary Event Group Item (EGI) of the current subscriber. In addition, there are fields for specifying the cycle code, month, and year.

### Layout

A Query parameter is prompted when selecting Rated Events. The query parameters are:

- Date range – Optional filtering; the default is the period of the specified cycle.
- Event type – Optional filtering; the default is all events.

Query Details	
Billing Cycle	Code: Cycle3, Month: Sep/2002
Subscriber ID	200003080, 423534 LTD
Date Range	13/08/2002  - 23/08/2002 
Event Type	GSM Voice 

cancel
back
submit

Figure 2-4: Query Definition for Rated Events

The next page is displayed after issuing the query. The page is generated dynamically, according to the query XML result. The following is an example:

Calls Rated Event : FE event query list					
Event type ID	Event ID	Network start time	Start time	Subscriber ID	Custom
910	<a href="#">760967531490900004</a>	2002-03-15 08:30:00	2002-03-15 08:30:00	200000140	200013
910	<a href="#">760967531490900005</a>	2002-03-16 07:00:00	2002-03-16 07:00:00	200000140	200013

Calls Rated Event : FE event query list					
Event type ID	Event ID	Network start time	Start time	Subscriber ID	Custom
913	<a href="#">760967531490900001</a>	2002-03-15 09:00:00	2002-03-15 09:00:00	200000140	200013
913	<a href="#">760967531490900002</a>	2002-03-15 10:00:00	2002-03-15 10:00:00	200000140	200013
913	<a href="#">760967531490900003</a>	2002-03-16 10:05:00	2002-03-16 10:05:00	200000140	200013



Query Details	
Billing Cycle Code	Cycle1
Billing Cycle Month	Mar/2002
Subscriber ID	200000140
Date Range	<input type="text"/>  - <input type="text"/> 

Figure 2-5: Query Results for Rated Events

The Event ID and Event Type ID fields are retrieved for all event types. All other fields are dynamically created according to the XML document returned by the API. Records are grouped by event type.

Clicking on an Event ID link opens the Event Details window (Zoom Event). Usually, this provides more details than are presented in the event list. The page elements depend on the external structure associated with the Get Event Details query. The following is an example:

Calls Rated Event Details : FE event query GSM voice detail	
Attribute name	Attribute value
Event type ID	910
Event ID	760967531490900004
Network start time	2002-03-15 08:30:00
Start time	2002-03-15 08:30:00
Subscriber ID	200000140
Customer ID	200013009
Guiding resource ID	0011112220001
Guiding resource type	C
Target customer ID	200013009
Account	200001403
Billing arrangement	20000040
Pay channel	200000100
Event group	R
Equipment ID	111111111111111
Pricing item ID	6
Source agreement ID	0
Charge amount Pounds	6.150
Charge including free allowance Pounds	6.150
Discount amount Pounds	0.000
Additional charge Pounds	0.000
Free charge amount Pounds	0.000
Payment category	POST
Event type name	GSM voice

Figure 2-6: Zooming a Rated Event Displayed in the Query Results

## Performance Indicators

This section describes the Amdocs front end for queries of performance indicators.

### Description

The Performance Indicators part of Usage Queries can be reached from both the Subscriber folder and the Agreement folder. The flow is the same for both cases.

### Layout

Query parameters are presented, allowing the user to select a cycle from the available cycle list.

Query Details	
Customer Billing Cycle	4
Month	Nov
Year	2003
Performance Indicator Category	Allowance

**Figure 2-7: Query Details for Performance Indicators**

The same mechanism described previously for events is used for PIs. The result page is dynamically generated, based on the XML document received by the query API. The following is an example:

Performance Indicators : FE PI query rate list						
Item ID	Agreement ID	Customer ID	Role	Accumulated charge Pounds	Number of events	Offer ID
6	<a href="#">200000180</a>	200005009	Rate	6.150	2	29

Query Details	
Customer Billing Cycle	15
Month	April
Year	2002
Performance Indicator Category	Default

**Figure 2-8: Query Results for Performance Indicators**

Clicking on the PI link opens the Performance Indicator Details window (Zoom PI), as shown in the example below:

Performance Indicators Details : FE PI query rate detail	
Name	Value
Item ID	6
Agreement ID	200000180
Customer ID	200005009
Role	Rate
Accumulated charge Pounds	6.150
Number of events	2
Offer ID	29
Account	200002200
Billing arrangement	200000220
Pay channel	200000550
Event group	R
Basic service type	M
First event date	2002-03-15 18:30:00
Last event date	2002-03-15 19:30:00
Accumulated charge including free allowance Pounds	6.150
Number of free events	0
Chargeable attribute	Duration
Accumulated quantity	30.000
Billed accumulated quantity	30.000
Free accumulated quantity	0.000
Period name	

**Figure 2-8: Zooming a Performance Indicator Displayed in the Query Results**

Moving among pages, if needed, is enabled by the Next and Previous links.



### 3. USAGE QUERY ARCHITECTURE

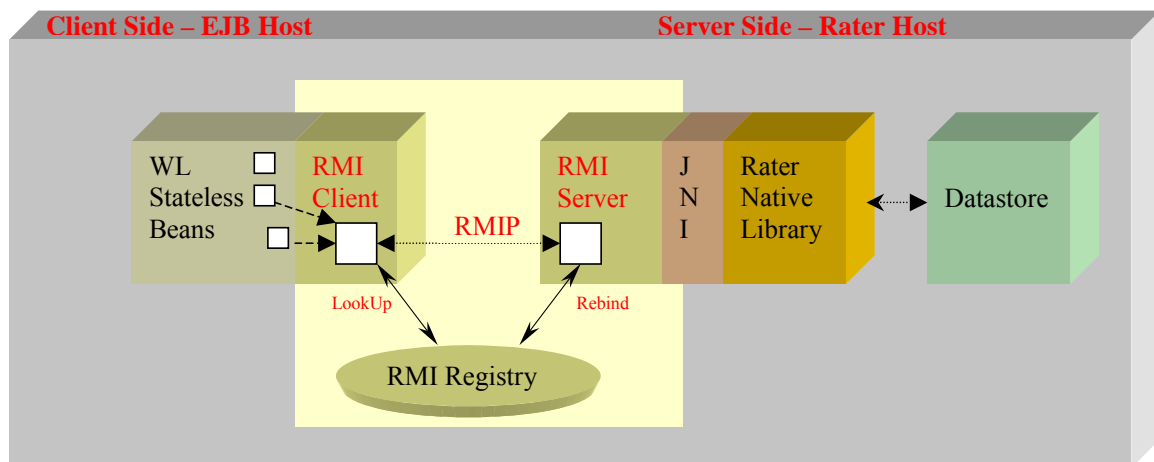
This chapter provides architectural details regarding the Usage Query APIs. It deals with the following issues:

- Development platforms
- Deployment platforms
- Distributed machine support
- External-to-core API connectivity
- Database-to-TimesTen connectivity

#### EJB and Rating API Connectivity

The usage queries include two layers. The core is the Pricing Engine APIs that are developed in C++ and wrapped as a UNIX shared library. The interface is implemented as EJB APIs developed in Java and deployed in a WebLogic application server. The core area is responsible for issuing the query to the database or TimesTen, invoking the mapping cases, and iterating over the results. The interface is responsible for handling the requests sent by the client, routing the query to the relevant server (according to cycle information), and receiving the query results from the core layer.

As mentioned, these two parts are developed in different technologies and architectures. The connection between these two layers is achieved by a third layer utilizing an RMI server. The following diagram illustrates this approach.



**Figure 3-1: Usage Query Architecture**

The RMI server is developed in Java and C++. Its objective is to provide a convenient wrapper for the Pricing Engine APIs to be used by the EJB layer whenever a Rating API is called. Since the external APIs are implemented in Java over EJBs, the adapter object should provide these modules with an easy way to access the Pricing Engine APIs (in C++).

The adapter's responsibilities are:

- Overcome the difference in the design approaches between the two layers; while the C++ APIs maintain state (e.g., the query API returns a reference to an iterator object that maintains a state). The EJB API is designed to be a stateless session bean in order to reduce system resources when serving many clients.
- Provide an envelope to the Rating core API (e.g., to wrap the Rating initialization methods responsible for module initialization, database connection, etc).
- Ensure full independence of the EJB layer, in which many other components are deployed. A socket-based RMI connection meets this requirement.

## Routing Requests to Responsible Rating Data Store

The Rating configuration supports a distributed process over multiple remote machines. The configuration is determined during deployment rather than development. This architecture was designed to enable the processing of huge volumes of data while maintaining high performance levels. Hence, whenever a PI or event query is requested, the result may reside on a different host, either in an Oracle account or in a TimesTen account.

The key to finding the target host derives from the cycle code, month, and year. The Usage APIs maintain the logic of how to find the responsible Rating machine (Usage Queries RMI server) and data store server that maintains the cycle information. Whenever a request is handled, the target host parameters are verified according to the cycle information and a reference entity that maintains the connection between the cycle and the remote server. According to this result, the request is forwarded to the target RMI Server.

The systems support flexible configurations. One possibility is to install an RMI server for each cycle. Adopting this configuration, a "main" application server that handles the client's requests is installed, finds the target server's URL, and routes the request to this server (as an RMI client). Another configuration option involves having one RMI server, responsible for multiple cycles. This server forwards the request to the Rating API adapter (written in C++). The server invokes the Pricing Engine Query API using a free connection from the connections pool. The minimum RMI server number is the number of usage hosts (one per host). Any other configuration combining these two types is supported. For example, an operator might choose to install a dedicated RMI server only for the busiest usage data store (for example, the one containing the latest confirmed cycles), while all the other databases are handled by another RMI server.

The following figure illustrates the architecture and possible installations:



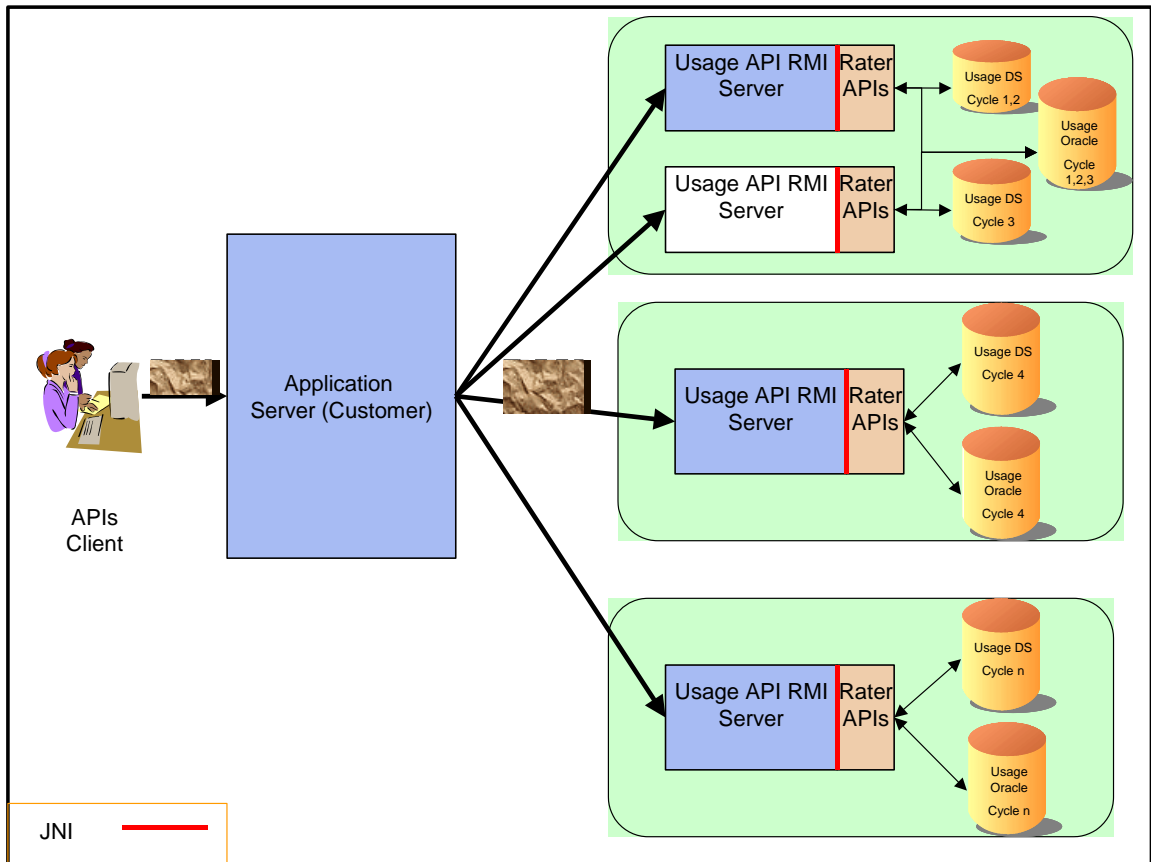


Figure 3-2: Usage Query Deployment

## Database Connectivity

Event data is located in the Oracle Rated Events table (**RATED\_EVENTS**), which can reside in different Oracle accounts and hosts.

While PI information for the current opened cycle may reside in a database in memory (TimesTen) or in an Oracle account, PI history is located in Oracle databases.

Connectivity to TimesTen or Oracle is accomplished using C++ connection objects. The Rating API host maintains a pool of connections for the various TimesTen and Oracle accounts. The pool parameters (minimum size, maximum size) are configurable.



## 4. IMPLEMENTATION EXAMPLES

---

This chapter provides several examples of implementing usage queries. The scenarios covered are:

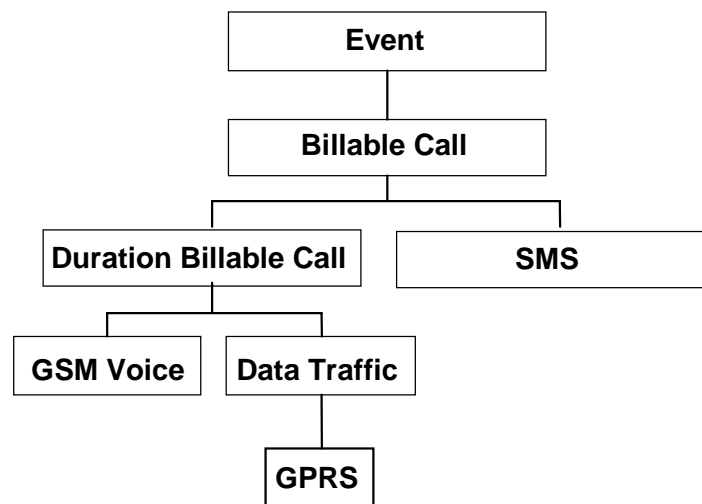
- Event list query, regardless of its type
- Event list query for data events
- PI details for rate-based PIs

Each scenario describes the implementation steps to be followed in order to support the requirement. The steps cover the following items:

1. External record definition in Product Catalog
2. Linkage between records and queries
3. Flow description

The examples assume the existence of the following Implementation Repository data:

- Actual event types:
  - GSM
  - GPRS
  - SMS
- Event hierarchy, as illustrated below:



**Figure 4-1: Event Hierarchy**

The examples also illustrate how to define new query outputs for rate-based PITs.

## Event List

This section describes the establishment of a generic event list query, regardless of event type, and a data traffic list query record. Any client can use the base record when a common list of events is presented, while the second record is used when the GPRS event list is invoked.

### Implementation Steps

To implement a generic event list query:

1. Create two new external records:

- FE Usage Query Basic List

The table below provides the information for this list.

Record Attribute Name	Event Type	Event Attribute	Default Value
Event Type ID	Event	Event type ID	
Event ID	Event	Event ID	
Network Start Time	Event	Network start time	Historical date
Start Time	Event	Start time	Historical date
Subscriber ID	Event	Service filter	Space
Customer ID	Event	Subscriber ID	
Guiding Resource ID	Event	Customer ID	
Guiding Resource Type	Billable event	Guiding resource ID	
Start Time	Billable event	Guiding resource type	
Target Customer ID	Billable event	Billing resource ID	
Account	Billable event	Billing resource type	
Billing Arrangement	Billable event	Target customer ID	
Pay Channel	Billable event	Account	
Event Group	Billable event	Billing arrangement	
Equipment ID	Billable event	Pay channel	
Pricing Item ID	Billable event	Event group	
Source Agreement ID	Billable event	Equipment ID	
Charge Amount	Billable event	Pricing item ID	
Charge Including Free Allowances	Billable event	Source agreement ID	
Discount Amount	Billable event	Charge amount	0

Record Attribute Name	Event Type	Event Attribute	Default Value
Additional Charge	Billable event	Charge including free allowances	0
Free Charge Amount	Billable event	Discount amount	0
Payment Category	Billable event	Payment category	
Roam Amt Air	Billable call	Roam amt air	0

- FE Usage Query Data Traffic List

The table below provides the information for this list.

Record Attribute Name	Event Type	Event Attribute	Default Value
Event Type ID	Event	Event type ID	
Event ID	Event	Event ID	
Network Start Time	Event	Network start time	Historical date
Start Time	Event	Start time	Historical date
Subscriber ID	Event	Service filter	Space
Customer ID	Event	Subscriber ID	
Guiding Resource ID	Event	Customer ID	
Guiding Resource Type	Billable event	Guiding resource ID	
Start Time	Billable event	Guiding resource type	
Target Customer ID	Billable event	Billing resource ID	
Account	Billable event	Billing resource type	
Billing Arrangement	Billable event	Target customer ID	
Pay Channel	Billable event	Account	
Event Group	Billable event	Billing arrangement	
Equipment ID	Billable event	Pay channel	
Pricing Item ID	Billable event	Event group	
Source Agreement ID	Billable event	Equipment ID	
Charge Amount	Billable event	Pricing item ID	
Charge Including Free Allowances	Billable event	Source agreement ID	
Discount Amount	Billable event	Charge amount	0
Additional Charge	Billable event	Charge including free allowances	0
Free Charge Amount	Billable event	Discount amount	0
Payment Category	Billable event	Payment category	
Roam Amt Air	Billable call	Roam amt air	0

Record Attribute Name	Event Type	Event Attribute	Default Value
Duration	Duration billable call	Duration	0
Billed Duration	Duration billable call	Billed duration	0
Free Duration	Duration billable call	Free duration	0
Rounded Duration	Duration billable call	Rounded duration	0
Total Volume	Data traffic	Total volume	0
Rounded Total Volume	Data traffic	Rounded total volume	0
Billed Total Volume	Data traffic	Billed total volume	0
Free Total Volume	Data traffic	Free total volume	0
Total Volume Charge Amount	Data traffic	Total volume charge amount	0

Map the records as follows:

Record Name	Event to be mapped
FE Usage Query Base List	Billable Call
FE Usage Query Data Traffic	GPRS

Link the external records by populating a reference table (RPR1\_QRY\_OUT\_FORMAT) to the list queries as illustrated in the following table:

Usage Type	Query Name	Query Subtype	External Records
Events	Get List	Default	FE Event Query Basic List
Events	Get List	GPRS	FE Event Query Data Traffic List

After the first three steps are completed, the system is ready for querying events. Whenever invoked, the Get Event List API returns an XML result based on the FE Usage Query Basic List record (if the event type is other than GPRS). For GPRS, the FE Usage Query Data Traffic List is reflected in the output result.

## PI Details for Rate-Based PIs

This section illustrates how a PI based on rate PITs is implemented and queried.

### Implementation Steps

To implement a query for PI details for rate-based PITs:

1. Create and map new external records as described in the following tables.
  - FE PI Query Rate List

The table below provides the information for this list.

Attribute Name	PIT	Attribute, Dimension, or Variable	PI Attribute or Dimension Name	Default Value
Item ID	Base rate	Attribute	Item ID	0
Agreement ID	Base rate	Attribute	Agreement ID	0
Customer ID	Base rate	Attribute	Customer ID	0
Role *	Base rate	None	None	“Rate”
Accumulated Charge	Base rate	Attribute	Accumulated charge	0
Number Of Events	Base rate	Attribute	Number of events	0
Offer ID **	Base rate	Attribute	Offer ID	0

\* Role – Should be mapped as hard-coded string equal to the PIT role. In this example, it contains the value “Rate.”

\*\* Offer ID – In this example, the offer ID is written. An alternative option is to invoke an external function decoding the offer name.

- FE PI Query Rate Details

The table below provides the information for this list.

Attribute Name	PIT	Attribute, Dimension, or Variable	PI Attribute or Dimension Name	Default Value
Item ID	Base rate	Attribute	Item ID	0
Agreement ID	Base rate	Attribute	Agreement ID	0
Customer ID	Base rate	Attribute	Customer ID	0
Role *	Base rate	None	None	“Rate”
Accumulated Charge	Base rate	Attribute	Accumulated charge	0
Number Of Events	Base rate	Attribute	Number of events	0
Offer ID	Base rate	Attribute	Offer ID	0
Account	Base rate	Dimension	Account	0
Billing Arrangement	Base rate	Dimension	Billing arrangement	0
Pay Channel	Base rate	Dimension	Pay channel	0
Event Group	Base rate	Dimension	Event group	0
Basic Service Type	Base rate	Dimension	Basic service type	Space
First Event Date	Base rate	Attribute	First event date	Historical date
Last Event Date	Base rate	Attribute	Last event date	Historical date

## Rating 6.0 Usage Queries Specification (Internal)

Attribute Name	PIT	Attribute, Dimension, or Variable	PI Attribute or Dimension Name	Default Value
Accumulated Charge Including Free Allowance	Base rate	Attribute	Accumulated charge including free allowance	0
Number Of Free Events	Base rate	Attribute	Number of free events	0
Chargeable Attribute		None	None	Space
Accumulated Quantity	Base duration/ volume/ occurrence rate	Attribute	Accumulated duration/ volume/ occurrence	0
Billed Accumulated Quantity	Base duration/ volume/ occurrence rate	Attribute	Billed accumulated duration/ volume/ occurrence	0
Free Accumulated Quantity	Base duration/ volume/ occurrence rate	Attribute	Free accumulated duration/ volume/ occurrence	0
Period Name	Period sensitive duration/ volume/ occurrence rate	Variable		Space

The external records should be linked to the list queries as illustrated in the following table:

Usage Type	Query Name	Query Subtype	External Records
PI	Get List	Rate	FE PI Query Rate List
PI	Get Details	Rate	FE PI Query Rate Details

After the first two steps are completed, the system is ready for querying PIs. Whenever invoked, the Get PI List API returns an XML result based on the FE PI Query Rate List record for rate-based PIs. The Get PI Details API returns an XML based on the FE PI query rate details record.



# Appendixes



# Appendix A. Rerating Data Storage Table

---

This appendix provides the technical information about the Rerating data storage table.

## Subscriber Rerate Data Storage

The Subscriber Rerate data storage table maintains the data for all subscribers marked for rerating.

The table has the following structure:

Field	Description
Bill Cycle Code	The customer bill cycle. A customer is associated with a unique bill cycle at any time.
Bill Cycle Month	The rated events for this month must be rerated.
Bill Cycle Year	The rated events for the year of the cycle month must be rerated.
Customer ID	The events of this customer must be rerated.
Activity Source	The activity that demanded rerating, e.g., Cancellation, Price Plan Change, or Out-of-Sequence Event.
Subscriber/ Agreement ID	The events of this subscriber must be rerated. If this value is zero, it implies that the entire customer is marked for rerating.
Activity Code	Technical cause for rerating, for example, a misplaced call in rating, manual request, or Customer Management activity.
Target System	Reguiding or Rerating.
Process Status	The current status of the Marked Subscriber, such as New or Obsolete.

The primary key of the table is:

- Bill Cycle Code
- Bill Cycle Month
- Bill Cycle Year
- Customer ID
- Subscriber/Agreement ID
- Activity Source

## Mark Subscriber for Rerate

This function is called to insert a record into the Subscriber Rerate data storage table with the following event data:

- Bill Cycle Code

## Rating 6.0 Specification (Internal)

---

- Bill Cycle Month
- Bill Cycle Year
- Customer ID
- Subscriber/Agreement ID
- Activity Source
- Activity Code

The function also calculates the value of the Target System field –Reguiding or Rerating.

If a record exists in the table with the same primary key (Bill Cycle data, Subscriber, and Activity), then the instruction to insert the record is ignored.

# Appendix B. Rolling Allowance Handling

This appendix describes a potential problem between Rerating and rolling allowances, and a recommendation for solving this problem.

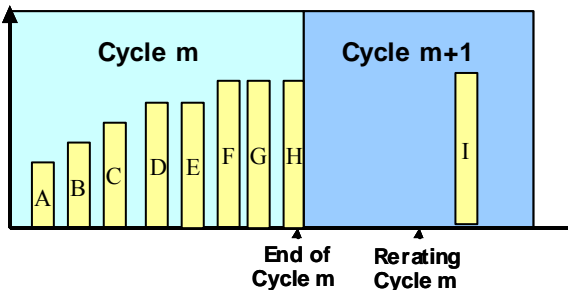
## Rolling Allowances and Rerating

If a subscriber has an offer that includes rolling allowances, and the subscriber events are rerated, it is possible that the subscriber may require rerating in the subsequent month due to incorrect rolling.

The sections that follow present two scenarios illustrating how Rerating handles rolling allowances.

### Rolling Allowances – Scenario 1

The diagram below depicts the situation in this scenario.

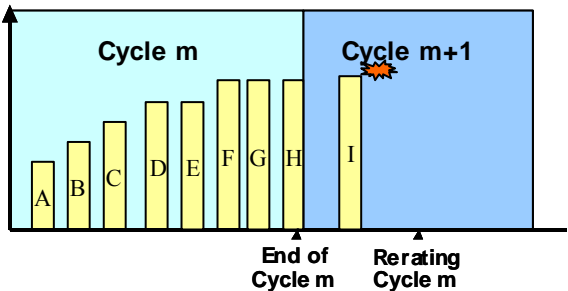


**Figure 4-2: Rolling allowances – Scenario 1**

Rerating is performed prior to any event entering the system for the same subscriber in the next bill cycle. In this case, the Pricing Engine identifies that the amount to be rolled over to the next cycle has not been rolled. Therefore, rerating of the next cycle is not required.

### Rolling Allowances – Scenario 2

The diagram below depicts the situation in this scenario.



**Figure 4-3: Rolling allowances – Scenario 2**

Rerating is performed, and at least one event has entered the system for the same subscriber in the next bill cycle. In this case, the Pricing Engine identifies that the amount rolled over to the next period was based on the results of the period being rerated. Therefore, rerating of the next cycle is required. The Pricing Engine automatically marks the subscriber for rerating in the next cycle.

## Recommendation

In order to prevent rerating of the same subscriber for every future bill cycle (due to the phenomenon shown in Scenario 2), it is recommended that rerating of the next bill cycle be performed immediately after rerating of the current bill cycle. This occurs naturally during the bill cycle (thereby requiring additional Rater downtime). Rerating of the next bill cycle should encounter Scenario 1 only.

# Appendix C. Mark Next Cycle for Rerate

---

This appendix describes a potential problem in the allocation of rolling allowances (cross-cycle PIs), and a recommendation for solving this problem.

## Allocation of Rolling Allowances

When a cross-cycle PI is rerated after it has already been reconstructed in the next cycle, the new cycle PI is incorrect because it was initialized with the wrong (that is, un-rerated) value. The following example illustrates the problem:

1. Cycle 4 is closed, and the cross-cycle PI is reconstructed in cycle 5. The initial value for cycle 5 is equal to the final value for cycle 4.
2. Cycle 4 is rerated, resulting in a new final value for cycle 4.
3. The PI for cycle 5 has already been initialized with the old final value for cycle 4 (i.e., not rerated). This is incorrect.

## Recommendation

To eliminate the problem, the following fields are included in the PI:

- Time stamp indicating the date when it was created
- Update date indicating the last time it was updated (e.g., by accumulation or rerating)

Normally, the time stamp of the new cycle PI should be equal to the update date of the previous cycle PI. A separate process (Mark Next Cycle for Rerate) verifies that this is so for each cross-cycle PI. If it encounters a discrepancy, the new cycle PI is marked for rerate, and thus its values are eventually corrected.





## Appendix D. Rerating Map

The following diagram depicts the Rerating job map.

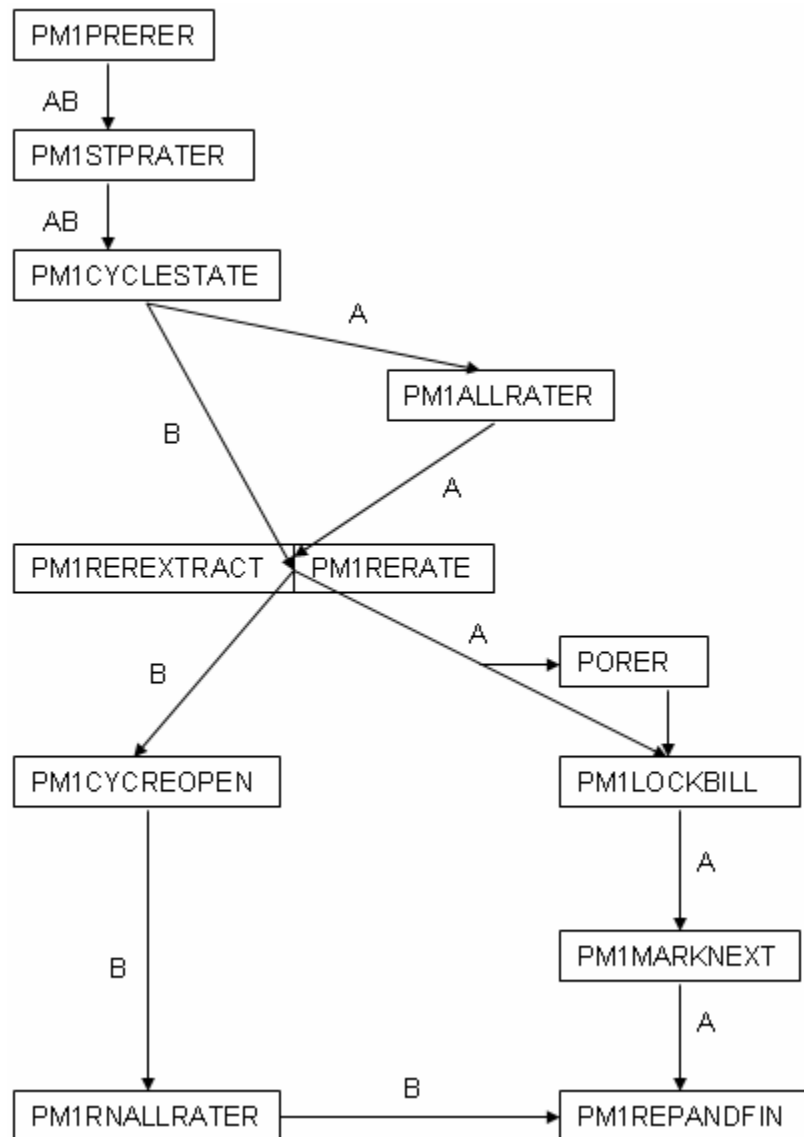


Figure D-1: Rerating map (A - end, B - mid)



amdocsrating

# Index



<b>A</b>	
A&F Flow .....	7
Acronyms.....	5
Additional charges role.....	18
Agreement-Level Handling, for Rerating .....	71
Allowance Evaluation.....	68
Allowance role .....	17
API Definition.....	128
APIs	
Billing Cycle .....	129
Archiving	
Performance Indicators .....	40
Auxiliary Repository.....	47

<b>B</b>	
Benefit role .....	18
Benefits .....	5
Billing	
Request for Rerating .....	57, 85
Undo Bill Preparation Rerating Scenario .....	61
Billing Cycle API.....	129
Billing cycle control.....	91
Billing Rerating.....	95
BOH.....	<i>See Business Organization Hierarchy</i>
Budget control role .....	18
Business Organization Hierarchy.....	14

<b>C</b>	
Combined Mode .....	38
Completion of Rerating.....	84
Customer data extract .....	44
Customer information flow .....	45
Customer information input.....	44
Customer Management	
Identifying Subscribers for Rerating .....	63
Marking Subscribers for Rerating .....	65
Rerating Scenarios, Allowance Activities .....	59
Rerating Scenarios, Retroactive Activities .....	59
Customer Mode, Rerating In.....	85
Cycle	
History.....	101
Truncate .....	101
Cycle Confirmation.....	95
Cycle control.....	91
Cycle Definition table .....	91
Cycle life cycle .....	93
Billing Rerating.....	95

Cycle Confirmation.....	95
Late Calls .....	94
Midcycle.....	94
Midcycle Rerating.....	94
Rating Extract.....	95
Cycle Open operation.....	95
Cycle operations.....	95
Cycle Open.....	95
Lock Bill.....	96
Lock Rerate .....	96
Cycle Rated Events Extract.....	81
Cycle State table .....	92
Cycle tables	
Cycle Definition table .....	91
Cycle State table.....	92
Partition Definition table .....	93
Cycle Tables.....	91

<b>D</b>	
Data Extract, for Reguiding and Rerating .....	79
Cycle Rated Events Extract.....	81
Previous Bill Cycle PI Extract.....	81
Subscriber Extract Job.....	80
Data Extract, Reguiding and Rerating	
Subscriber Extract Job.....	79
Data storage .....	43
Data Storage Tables	
Postpaid Recovery.....	113
Data Storage Types .....	37
Combined Mode.....	38
Oracle Mode.....	38
TimesTen Mode .....	38
Discount role.....	18
Dispatch	
Pricing Engine flow .....	30
Dispatcher .....	21, 40, 50
Document scope.....	3

<b>E</b>	
Envelopes.....	4, 31
Event Handler	
Invocation.....	29
Segmentation.....	29
Event handler evaluation	
Pricing Engine flow .....	29
Event handling	
Pricing Engine flow .....	26
Event Processing	
Flow .....	7

## Rating 6.0 Specification (Internal)

---

Overview .....	7
Event Processing Flow	
A&F to Rating .....	7
Rating .....	8
Event rating .....	15
By duration .....	15
By period and duration .....	15
Flexible rating schemes .....	16
Voice rating examples .....	15
Event Rating	
By accumulated usage, period, and duration .....	16
Event types .....	11
Events .....	11
Determining Rerating Requirements .....	73
Out-of-Sequence, Marking for Rerating .....	67
Execution Policy .....	28
External system interfaces .....	49
Extract	
Customer data .....	44

## F

Flexible rating schemes .....	16
Flow .....	107
A&F to Rating .....	7
Customer information .....	45
Event Processing .....	7
Online Charging .....	34
Postpaid Rater .....	32
Pricing Engine .....	24
Rating .....	8
Rerating .....	33
Usage, Rating to Oracle .....	39
Front End Client .....	130
Functionality overview .....	11
Functionality Overview .....	105
Functions .....	89
Main Rating functions .....	3

## G

Guiding to Service	
Pricing Engine flow .....	27
Qualification and priorities .....	11

## H

Hierarchy .....	14
-----------------	----

## I

Implementation Repository .....	46
Implementation Repository Definition .....	124
Common Event Members (Header) .....	124
Common PI Members (Header) .....	124

External Record .....	125
External Records, Examples .....	125
Linking Between Query and External Records .....	125
Implementation Repository, Rerating .....	74
Offer-Dependent .....	75
Offer-Independent .....	74
Initialization	
Pricing Engine flow .....	25
Input	
Auxiliary Repository .....	47
Customer information .....	44
Data storage .....	43
Implementation Repository .....	46
Pricing Package, Service Package, and Offer Catalogs .....	46
Product Catalog .....	45
Rating .....	43

Inputs	
Operational data .....	24
Pricing Engine .....	23
Pricing Engine configuration .....	23
Product Catalog data .....	23
Interaction with Other Components .....	105
Interactions with Other Components .....	4
Interfaces .....	4
Dispatcher .....	50
External systems .....	49
Online usage queries .....	50
Output .....	49
Rating Extract .....	50
Introduction .....	3, 89, 105

## L

Late Calls .....	94
Lock Bill operation .....	96
Lock Rerate operation .....	96

## M

Main functions .....	89
Manual Marking of Subscribers for Rerating .....	70
Map, Rerating .....	155
Mark Next Cycle for Rerate .....	153
Mark Subscriber for Rerate table .....	150
Methodology	
Recovery .....	109
Midcycle .....	94
Midcycle Rerating .....	94
Modules	
Envelopes .....	4
Main Rating modules .....	4
Pricing Engine .....	4

**O**

Offline Charging .....	<i>See</i> Postpaid Rater
Online Charging .....	4, 33
Flow .....	34
Interface to Pricing Engine .....	34
Online usage queries .....	50
Operational data .....	24
Oracle Mode .....	38
Output	
Dispatcher interfaces .....	50
Interfaces .....	49
Online usage query interfaces .....	50
Postpaid Rating .....	37
Rating Extract interfaces .....	50
Overview .....	3, 105
Event Processing .....	7
Functionality .....	11

**P**

Package priority .....	14
Partition Definition table .....	93
Performance indicators .....	19
Classification .....	21
Global and Partitioned .....	20
Maintenance .....	97
Objectives .....	19
Structure .....	19
Performance Indicators	
Archiving .....	40
Creation .....	40
Maintenance .....	40
Query .....	119
Reconstructing .....	98
PI Maintenance process .....	97
Configuration file parameters .....	98
Description .....	98
Flow .....	99
General .....	97
PI Reconstruct, Inaccurate accumulation .....	61
PI \t .....	19
Postpaid Rater .....	4, 31
Recovery Mechanism .....	32
Postpaid Rater Flow .....	32
Postpaid Rating Output .....	37
Postpaid Recovery Table .....	113
Preparation for recovery .....	109
Previous Bill Cycle PI Extract .....	81
Pricing Engine .....	4, 23
Pricing Engine flow .....	24
Dispatch .....	30
Event handler evaluation .....	29
Event handler invocation .....	29
Event handler segmentation .....	29

Event handling .....	26
Execution policy .....	28
Guiding to Service .....	27
Initialization .....	25
Qualification .....	27
Write results .....	30
Pricing Engine inputs .....	23
Configuration .....	23
Operational data .....	24
Product Catalog data .....	23
Priorities	
Guiding to Service .....	11
Priority in Selecting and Executing Services .....	13
Process Flow .....	111
Process, Recovery .....	111
Processes	
Truncate Cycle .....	102
Processes, Reguiding .....	83
Processes, Rerating	
Flow .....	83
Preparing the Files .....	83
Product Catalog data .....	23
Product Catalog input .....	45
Auxiliary Repository .....	47
Discount Package Catalog .....	46
Implementation Repository .....	46
Offer Catalog .....	46
Pricing Package Catalog .....	46
Service Package Catalog .....	46

**Q**

Qualification	
Event type and service filter .....	12
Guiding to Service .....	11
Pricing Engine flow .....	27
Qualification criterion .....	13
Queries, Usage	
API Definition .....	128
Architecture .....	135
Billing Cycle API .....	129
Database Connectivity .....	138
Definitions .....	120
Definitions, Event Details .....	122
Definitions, PI Details .....	123
Definitions, PI List .....	121
Definitions, Rated Event List .....	120
EJB and Rating API Connectivity .....	135
Front End Client .....	130
Implementation .....	139
Implementation Repository Definition .....	124
Implementation, Event List .....	140
Implementation, PI Details for Rate-Based PIs .....	143

Routing Requests to Responsible Rating	
Data Store .....	136
Technical Overview .....	123
Query Flow .....	119
Performance Indicators .....	119
Rated Events.....	119
 <b>R</b>	
Rated Events	
Query.....	119
Rater Process	
Flow, Preparing the Data for Recovery..	109
Rating	
Detection .....	64
Identifying Subscribers for Rerating .....	64
Inputs.....	43
Main functions .....	3
Marking Out-of-Sequence Events for	
Rerating .....	67
Outputs .....	37
Rerating Scenarios, Chronologically	
Inaccurate Events.....	60
Rerating Scenarios, Inaccurate Product	
Catalog Entries .....	60
Rating an event .....	15
Rating Extract .....	50, 95
Rating Flow.....	8, 107
Rating role.....	17
Rating to Oracle usage flow.....	39
Reconstructing Performance Indicators .....	98
Recovery	
Data Storage Tables .....	113
Interaction with Other Components .....	105
Methodology .....	109
Overview .....	105
Preparation .....	109
Process Flow .....	106, 111
Recovery Mechanism.....	32
Recovery Process .....	111
Reguiding and Rerating	
Background Information .....	80
Cycle Rated Events Extract.....	81
Extracting Data For .....	79
Previous Bill Cycle PI Extract.....	81
Processes .....	83
Subscriber Extract Job.....	79, 80
Reguiding, Processes .....	83
Replicator .....	39
Rerate Population File.....	85
Rerating	
Mark Next Cycle for .....	153
Rerating.....	4, 33, 55
Billing Rerating .....	95
Data Storage Tables .....	149
due to Billing Request .....	85
Due to Billing Request .....	57
due to Billing Request, Rerate Population	
File.....	85
During an Open Cycle.....	78
Events, Determining Requirements .....	73
Extracting Data for .....	79
Flow.....	83
Identifying Subscribers for .....	63
Identifying Subscribers for, Customer	
Management .....	63
Identifying Subscribers for, during Rating	64
In Customer Mode.....	85
Interaction with Other Components .....	55
Main Functions.....	55
Marking for, With Investigation.....	68
Marking for, Without Investigation.....	67
Marking of Subscribers, Agreement Level	
Handling .....	71
Marking of Subscribers, Manual .....	70
Marking Out-of-Sequence Events via	
Rating .....	67
Marking Subscribers for .....	65
Midcycle Rerating .....	94
Process Flow .....	55
Processes .....	83
Processes, Preparing the Files .....	83
Processing Subscribers for .....	77
Recommended Solution .....	78
Rolling Allowance.....	151
Subscribers, after Locking of Bill Cycle ..	77
Tables, Mark Subscriber for Rerate.....	150
Tables, Subscriber Rerate Data Storage .	149
Rerating Completion .....	84
Rerating Events	
Offer-Dependent.....	75
Offer-Independent .....	74
Rerating Events, Determining Requirements	
Handling Subscriber Activity Events .....	74
Implementation Repository Definitions ..	74
Receiving Subscriber Activity Events .....	73
Subscriber Events .....	76
Rerating flow.....	33
Rerating Map.....	155
Rerating Scenarios .....	59
Billing, Undo Bill Preparation.....	61
Customer Management, Allowance	
Activities.....	59
Customer Management, Retroactive	
Activities.....	59
PI Reconstruct .....	61
Rating, Chronologically Inaccurate Events	
.....	60
Rating, Inaccurate Product Catalog Entries	
.....	60



Roles		Partition Definition.....	93
Additional charges .....	18	Postpaid Recovery.....	113
Allowance .....	17	Subscriber Rerate Data Storage.....	149
Benefit.....	18	Terminology.....	5
Budget control.....	18	TimesTen Mode .....	38
Discount.....	18	Truncate Cycle Process.....	101, 102
Rating.....	17		
Rolling Allowance .....	151	<b>U</b>	
<b>S</b>		Usage Data Storage.....	37
Scenarios		Usage flow	
Rerating.....	59	Rating to Oracle .....	39
Rolling Allowance .....	151, 152	Usage Partitioning.....	101
Scope.....	118	Usage Queries	
Scope of document.....	3, 89	API Definition.....	128
Scope of this Document.....	105	Architectural Principles.....	118
Stepped Evaluation .....	69	Architecture.....	135
Storage		Billing Cycle API.....	129
Combined Data Storage Mode .....	38	Database Connectivity.....	138
Data Storage Types .....	37	Definitions.....	120
Oracle Data Storage Mode .....	38	Description .....	119
TimesTen Data Storage Mode.....	38	EJB and Rating API Connectivity.....	135
Subscriber Events, Rerating.....	76	Flow .....	119
Subscriber Extract Job .....	79, 80	Front End Client.....	130
Subscriber Rerate Data Storage table.....	149	Implementation .....	139
Subscribers		Implementation Repository Definition...	124
Identifying for Rerating.....	63	Implementation, Event List .....	140
Marking for Rerating, Agreement-Level		Implementation, PI Details for Rate-Based	
Handling.....	71	PIs.....	143
Marking for Rerating, Manual .....	70	Overview .....	117
Processing for Rerating .....	77	Routing Requests to Responsible Rating	
Rerating, after Locking of Bill Cycle.....	77	Data Store .....	136
Rerating, During an Open Cycle.....	78	Technical Overview .....	123
Rerating, Recommended Solution.....	78	Usage Queries, Definitions	
Subscribers, Marking for Rerating via		Event Details .....	122
Customer Management .....	65	PI Details.....	123
Customer Management to Subscriber File.....	65	PI List.....	121
Pricing Engine Processing.....	67	Rated Event List .....	120
Subscriber File to Subscription Manager .....	66		
Subscribers, Marking for Rerating via Rating		<b>V</b>	
Allowance Evaluation Mechanism.....	68	Voice event rating	
Stepped Evaluation Mechanism.....	69	By duration.....	15
with Investigation.....	68	By period and duration.....	15
without Investigation.....	67	Voice event Rating	
		By accumulated usage, period, and duration	
<b>T</b>		.....	16
Tables		Voice event rating examples.....	15
Cycle .....	91		
Cycle Definition.....	91	<b>W</b>	
Cycle State .....	92	Write results	
Data Storage.....	149	Pricing Engine flow.....	30
Mark Subscriber for Rerate .....	150		



## Document Release Information

Case No.	Service Pack No.	Description of Change
N/A	SP8	Chapter 5 of the Rating 6.0 Recovery Specification section was updated with information on the Postpaid Recovery table.