amdocs rating

Rating 6.0
Release Notes

Document Information

Software Version: **6.0**

Publication Date: **January 2005**

Catalog Number: 212302

Contents

1. INTRODUCTION

This chapter lists the changes in Amdocs Rating 6.0. The changes are described in greater detail in the next chapter.

What's New in this Version

This new release of Amdocs Rating 6.0 includes the following:

- New functions and enhancements:
  - Rerating support for Bill Preparation
  - Run-time reference data update
  - Rerating decoupled from Product Catalog
  - Performance Indicator reservation support
  - Sourceless product
  - Flexible bill cycle support
  - Flexible bill frequency support
  - Allowance proration support
  - Unicode and internationalization support
  - New extension functions
  - Vertical versioning support
  - Customer offer parameters support
  - Complex attribute support for all core types
  - Performance Indicator global attributes
  - Skip PITs with no PIs
  - Rated and reject event fielded database representation
  - Support for loops on complex attributes
  - Temporary event variables
  - Period set minutes resolution support
  - Rerating recovery
  - Multithreaded offline rater
  - Parallel processing for enhanced performance
  - Compressing rated event BLOB column
- Technical changes

2. New Functions and Enhancements

This chapter describes the new functions, and enhancements of existing functions, included in Amdocs Rating 6.0.

Rerating Support for Bill Preparation Pipelining

The Rerating process supports the new Bill Preparation pipelining functionality, which enables Bill Preparation and Rerating to run concurrently on separate groups of customers.

The Bill Day Initiator splits its customer population into two groups for the PI and event extracts, those who require rerating and those who do not. The rerate map has been enhanced to extract the rerate population as its first step. Billing then can proceed to process the second group while the first group is being rerated. Only customer groups requiring rerating need to wait for Rerating to finish before continuing to the PI and Event extracts.

Two new processes are added to the rerate map to support these enhancements:

- Report and Prepare Rerate Population – Updates the history table with the population marked for rerate and sends a report to Billing.
- Report and Finalize – Reports all the customers that have been marked for rerating as a result of reguiding to Billing. This job is activated only if the whole population was rerated successfully.

Run-Time Reference Data Update

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.

Rating Decoupled from Product Catalog

The new Product Catalog Version Management API does not extract Product Catalog version data to tables in the Rating reference data area. 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 as needed. The callback functions are responsible for extracting the data from the XML files and populating the relevant Pricing Engine reference data tables. This improves Product Catalog performance by eliminating the overhead of updating Rating tables. Some tables that were in the Product Catalog data area also have been moved to the Rating area for greater efficiency.

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, which does not load incrementally. The Product Catalog also now exports period set, special days, and scale data, too.

The new API also recognizes Product Catalog baseline versions, and updates the Rating tables accordingly (merging prior versions into the baseline version and deleting obsolete versions).

Performance Indicator Reservation Support

The Pricing Engine and Online Charging now manage Performance Indicator (PI) quantity values when processing reserve, debit, and release requests for Online Charging.

The decision whether to update PI values due to a reserve request is a pure business decision. The Pricing Engine supports any business decision, and can update all, specified, or no PI quantity attributes.

These request types are supported:

- *Reserve Request* – Two reserve options are supported:
  - *Full Amount Reserve Request* – Reservation of the full amount since the beginning of the session, that is, each reserve request arrives with the previous quota reserved amount and the current reserved amount for this session.
  - *Quota Reserve Request* – Reservation of only the next quota reserved amount that needs to be authorized.
- *Debit Request* – Two debit options are supported:
  - *Partial Debit Request* – A debit request that releases part of the reservation (only the debit amount) previously requested for the session.
  - *Full Debit Request* – A debit request that releases all reservations previously requested for the session.
- *Release Request* – Releases all reservations previously requested for the session, with no further processing.

Sourceless Product

This version of Rating is provided as a sourceless product.

Flexible Bill Cycle

In previous versions, each customer belonged to a bill cycle, and a cycle would be executed on one host. Now, cycles are not one entity, and can be executed on different hosts.

Now the rating partition (a usage host in which the events are rated and stored) is independent of the cycle, and customers belonging to a particular

cycle can be distributed across multiple partitions according to their Partition IDs. The partition definition table is maintained by Rating.

This distributed cycle deployment balances the event processing load and reduces the system's hardware requirements. It also simplifies the Change Cycle operation, since it no longer requires moving a customer between partitions.

However, cycle compatibility mode is provided, so the entire population of a bill cycle can be maintained on a single rating partition, if desired.

Usage Queries

Usage queries now operate on the population determined by the partition ID, rather than the cycle code as in previous Rating versions.

Usage queries now can be filtered by business arrangement (BA). This applies to usage queries at both the subscriber and the customer level.

Usage queries remain backwardly compatible with rated events created under previous versions of Rating.

Usage Extracts

The usage extracts now can select information for a BA from more than one cycle instance.

Offline Rater Envelope

Offline rater envelope instances are no longer limited to a single cycle, and can handle all cycles that are stored in a particular partition.

Flexible Bill Frequency

A customer's bill cycle frequency now can be defined as:

- Weekly, biweekly, or any other factor of a week
- Monthly, bimonthly, quarterly, or any other factor of a month up to a year

Moreover, the bill production frequency for a billing arrangement can be defined as a multiple of the related customer's cycle frequency, allowing the BA to be billed at a different frequency from the customer.

The following extension functions are provided in support of flexible bill frequency:

- *Get Cycle Frequency* – A new extension function that returns the frequency of the cycle associated with the current event. The return value can be Monthly or Weekly.
- *Get Cycle Multiplier* – A new extension function that returns the multiplier of the cycle associated with the current event.
- *Scale extension functions* – All the scale extension functions are extended with a multiplier parameter to adapt to multiple monthly or weekly frequencies. These extension functions multiply the scale values by the given multiplier. The extension functions support a default multiplier of 1 if no multiplier parameter is passed.

Allowance Proration

New functionality supports proration of allowances for subscriber moves and cycle changes.

Move Subscriber

When a subscriber is moved to a different customer, the subscriber's allowances are prorated as if the related service is cancelled on the date of the move. If the service is reconnected under the new customer, the new allowances are prorated according to the effective date.

The proration policy for each activity code is maintained in a generic reference table (GRT).

The Get Proration Factor extension function has been adapted to this functionality. It prorates the allowances when a subscriber is moved, even if they are defined as not prorated (depending on the policy in the GRT).

For implementations that are upgrading to version 6.0, the new functionality will apply only to new transactions that update customer offers.

Change Cycle

When a customer's billing cycle is changed, certain types of allowances (such as "use it or lose it" allowances) are prorated in the new cycle, even if they are defined as not prorated. (The old cycle terminates on its regular close date, so proration is not required.)

Proration calculations are based on a new application table, managed by the Rater Update Handler, that tracks customer change cycle request histories.

Unicode and Internationalization Support

Rating now accepts and supplies data in any language through its multibyte and Unicode support.

The Unicode standard is a character encoding standard used for the interchange, processing, and display of written text of different languages and technical disciplines. A multibyte character set is a character set encoded with a variable number of bytes per character. This representation is especially common in the Asia-Pacific market.

New Extension Functions

A number of extension functions have been added or enhanced. They are described in the *Rating Out-of-the-Box Extension Functions* document.

Vertical Versioning

A vertical versioning mechanism has been added to the Product Catalog for offers and packages.

Previously, all elements were managed on the same timeline (horizontal versioning). Now offers and packages can be managed on their own

individual time lines (vertical versioning), because the changes in their life cycles generally do not affect other elements. (Implementation Repository and Auxiliary Repository elements continue to be managed with horizontal versioning.)

The Pricing Engine supports vertical versioning of independent offers and pricing packages. This includes incremental loads, and loads of future or backdated offers and pricing packages.

Customer Offer Parameters

The new version of the Product Catalog introduces offer-level customer parameters (OLCPs). These OLCPs are filled with values during order capture, i.e., when the offer that contains items referring to those parameters is assigned to the customer.

Rating recognizes and uses OLCPs in the qualification and computation stages of rating, similarly to the way Customer Parameters are used.

Complex Attribute Support for All Core Types

Rating now supports Date and Boolean elementary types as core types on which to base complex attributes.

Performance Indicator Global Attributes

Rating also now supports the Product Catalog's global flag for PI attributes. This means that the Pricing Engine now can distinguish between "global" and "partitioned" PI attributes.

Global attributes are attributes that exist once in a partitioned PI, as opposed to partitioned attributes, for which a different instance exists in every partition. (This distinction has no meaning when a non-partitioned PI is declared.)

"Partitioned" has become the default value instead of "global."

Skip PITs with No PIs

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

On initialization, the Pricing Engine loads only the PIT handlers that are designated for loading in the new PIT Run Mode configuration variable.

Rated Event Fielded Database Representation

The Rated Event database now includes a standard fielded representation, as well as a binary large object (BLOB) representation.

The database representation is partitioned into two parts:

- *Static* – Core attributes used by the core Pricing Engine are static. These attributes are represented as database table fields. Such attributes are called “fielded” attributes. These attributes are part of the base event type.
- *Dynamic* – Attributes introduced by different Product Catalog implementations are dynamic. These are all represented by a single varchar column (the BLOB), which encodes the attribute values in a proprietary format. Attributes of every event type defined by the Product Catalog are stored in the dynamic partition.

Event BLOB Resolution

The new Rated Event database structure includes fielded data as well as a BLOB. The previous Oracle data layer has been replaced with a new data layer, based on the I/O Framework, which supports dynamic SQL. The Oracle data layer supports fielded persistence for any subset of the dynamic event attributes.

The design is based on these principles:

- Fielded representation is supported in addition to the previous binary representation. The Product Catalog specifies the representation at the event attribute level.
- All attributes defined for all event types are represented as fields in the Rated Event database, that is, a single physical record with the superset of all attributes defined for all event types.
- The hybrid mode allows for specifying, for each event type attribute, whether it is stored as a field or in the binary part.
- The Product Catalog specifies, for each event type attribute, whether it will be stored in the Rated Event table.
- A new dynamic data layer translates between the in-memory entity binary format and the fielded representation.
- Standard Oracle tools can be used for performing rated event extracts, queries, and other operations, without the need to invoke the Pricing Engine.
- The fielded columns can be used for performing queries and defining database indexes, which is not possible otherwise. However, creating indexes on the Rated Event table is not practical, due to its size.

Fielded Event Persistence

Attributes of rated events and rejected events can be stored in separate database columns. This new capability is in addition to the previous usage persistence mechanism of storing the dynamic usage attributes in binary format. The Product Catalog specifies the persistence type, whether fielded or binary, at the event type attribute level.

The following types of persistence for the Rated Event and Rejected Event tables are supported:

- Full binary – All dynamic attributes are stored in a single binary varchar column (same as the previous situation).
- Full fielded – All dynamic attributes are stored as database columns.
- Hybrid – Some of the dynamic attributes are stored as database columns, while others are stored in a singe binary varchar column.

Loops on Complex Attributes

Complex attributes are attributes with multiple values, where each value is associated with one or more dimension identifiers. The Product Catalog now supports indexed For loops, allowing a set of commands to be performed for each element or dimension in a complex attribute. Nested loops are supported.

The Pricing Engine fully supports Product Catalog loops on complex attributes.

Temporary Event Variables

The Product Catalog now maintains event variables, which are used to pass information from one PIT to another among all those executed for an event. The variable's life cycle is the same as that of the event being processed.

The Pricing Engine also supports the new event variable entity, which holds all temporary variables defined for an event. Dispatch case mapping can use the temporary event variable as well.

Period Set Resolution

In the Product Catalog, period sets now can be defined in quarter hours or minutes as well as hours.

Rating supports all period set resolutions.

Rerating Recovery

The Rerate process can be activated in a special recovery mode after an abnormal termination.

On a failure, the input file remains in "In Use" status, and no output entries are created in Audit & Control for the partial output files that were created. The last successfully processed customer is recorded in the recovery table. This enables the Rerate process in recovery mode to continue processing from the next customer.

The rerate map receives a unique key from Billing that includes the customer partition number. This key is used to verify the uniqueness of the entries in the recovery table for each map execution. (Billing provides the same key to the PI and event extracts on activation, to allow them to check when the rerate process is finished.)

The rerate map terminates when an error occurs, and must be rerun by Billing with the same map key.

Multithreaded Offline Rater

The offline rater envelope architecture has been changed to support multithreaded processing instead of multiprocess processing.

One or more rater daemons can serve rating partitions and multiple cycles.

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

Compressing Rated Event BLOB Column

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 types.

Any third-party dynamic library that implements a specific API for compressing and expanding binary data can be used to compress the BLOB column when storing data in the Oracle database. This feature reduces the use of database resources but can impair performance.

The Pricing Engine initializes the compressing library by querying a property database table, which holds the library name.

Rated Event and Rejected Event 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.

Event BLOB compression is optional, and occurs only in Full Binary or Hybrid modes, when the property table is set and the specific library exists.

3. TECHNICAL CHANGES

This chapter contains detailed technical information about the changes to Amdocs Rating 6.0.

General Technical Changes

No changes.

Deprecated APIs

No changes.

Third-Party Item Changes

No changes.

Amdocs Infrastructure Changes

No changes.

Configuration Control (CC) Changes

This section describes the Configuration Control changes in the current version.

CC Variables

No changes.

CC Configuration

No changes.

Customized Building Blocks

No changes.

Fixed Local Patches

No changes.

Configuration XML Changes

No changes.

Environment Changes

This section describes the environment changes in the current version.

Environment Variables

No changes.

Environment Structure

No changes.

WebLogic File Changes

This section describes the file changes to the WebLogic application server in the current version.

EJBs and Classes

No changes.

WebLogic Complementary Files

No changes.

Database Structure Changes

This section summarizes the changes to the data model in this version.

New Oracle Tables

The following tables describe new Oracle tables added in this version.

PM1_CONF_HIERARCHY

This table holds the hierarchy of Pricing Engine properties with other components.

PK	Name	M	Type (length)	Functionality	Valid Values	Domain Name
Y	PROCESS_NAME	Y	Char(32)	The name of the process that depends on other processes	N	pm1procname
N	PROCESS_GROUP	Y	Varchar2(256)	The name of the processes which the described process depends on	N	pm1procgrp

PM1_CONF_SECTION_PARAM

This table holds the global properties for the Pricing Engine. This is a configuration table that holds compression library's name.

PK	Name	M	Type (length)	Functionality	Valid Values	Domain Name
Y	CUSTOM_LEVEL	Y	Number(4)	High value override	N	pm1custlevl
Y	PARAM_CLASS	Y	Char(32)	The parameter class	N	pm1paramcls
Y	SECTION_NAME	Y	Varchar2(256)	Section name	N	pm1name
Y	PARAM_NAME	Y	Varchar2(256)	Parameter name	N	pm1name
N	PARAM_VALUE	N	Clob(1000)	Parameter value	N	pm1clob

PM1_CUSTOMER_OFFER_PARAMS

This table holds the customer offer parameter entities, which describe the customer parameters at the offer level.

PK	Name	M	Type (length)	Functionality	Valid Values	Domain Name
Y	SUBSCRIBER_ID	Y	Number(9)	Identifies the subscriber	N	pmsubagrno
Y	OFFER_INSTANCE	Y	Number(9)	The instance of the subscriber's offer. This value is retrieved from the offer in customer offers that passed qualification	N	pm1seqno
Y	EFFECTIVE_DATE	Y	Date(14)	Identifies the effective date of the parameters	N	datetime
N	EXPIRATION_DATE	N	Date(14)	Identifies the expiration date of the parameters	N	datetime
N	SYS_CREATION_DATE	N	Date(14)	The creation date of the record	N	datetime
N	SYS_UPDATE_DATE	N	Date(14)	The last update date of the record	N	datetime
N	APPLICATION_ID	N	Char(6)	The ID of the last application that updated the record	N	pmapplid
N	DYNAMIC_DATA	N	Varchar2(4000)	The list of parameters	N	pmtext4000

PM1_CYCLE_CHANGE

This table holds the change cycle requests that are made by the customers.

PK	Name	M	Type (length)	Functionality	Valid Values	Domain Name
Y	CUSTOMER_ID	Y	Number(9)	Identifies the customer that is associated with the request	N	pmcstomerid
Y	EFFECTIVE_DATE	Y	Date(8)	The date from which the new cycle becomes effective. (Old cycle close date + 1)	N	date
N	NEW_CYCLE_CODE	Y	Number(4)	The new cycle code	N	pmcyclecode
N	OLD_CYCLE_CODE	N	Number(4)	The changed cycle code	N	pmcyclecode
N	STATUS	N	Char(1)	For future use	N	pm1srsts
N	SYS_CREATION_DATE	Y	Date(14)	The creation date of the record	N	datetime
N	SYS_UPDATE_DATE	N	Date(14)	The last update date of the record	N	datetime
N	APPLICATION_ID	N	Char(6)	The ID of the last application that updated the record	N	pmapplid

PM1_ITEM_VERSION

This table contains item-specific information. Version information of pricing items is inherited from parent packages.

PK	Name	M	Type (length)	Functionality	Valid Values	Domain Name
Y	ID	Y	Number(9)	The ID of the item	N	pm1id
Y	PACKAGE_ID	Y	Number(9)	The ID of the package this item belongs to	N	pm1id
Y	MAJOR_VERSION_ID	Y	Number(3)	Major version of the item	N	pm1subvrsid
N	FIX_VERSION_ID	Y	Number(9)	Fix version of the item	N	pm1id
N	PRICING_ITEM_TYPE	Y	Varchar2(255)	The name of the PIT that describes the item	N	pm1prit
N	DYNAMIC_DATA	N	Blob(3200)	The XML file that describes the item parameters	N	pm1blob
N	NAME	Y	Varchar2(255)	The name of the item version	N	pm1name255

PM1_OFFER_VERSION

This table contains offer-specific information. Common version information is stored in the PM1_VV_ELEMENT table.

PK	Name	M	Type (length)	Functionality	Valid Values	Domain Name
Y	ID	Y	Number(9)	The ID of the offer	N	pm1id
Y	MAJOR_VERSION_ID	Y	Number(3)	Major version of the offer	N	pm1subvrsid
N	FIX_VERSION_ID	Y	Number(3)	Fix version of the offer	N	pm1subvrsid
N	SALE_EFFECTIVE_DATE	N	Date(14)	The date in which the offer starts to be active	N	datetime
N	SALE_EXPIRE_DATE	N	Date(14)	The date in which the offer stops to be active	N	datetime
N	OFFER_TYPE	Y	Char(1)	'P' – Price plan 'A' – Additional offer 'D' – Discount 'M' – Market Other value – ignore the offer	N	pm1gentype
N	OFFER_LEVEL	Y	Char(1)	The level of the offer: 'S' – for subscriber 'G' – for group	N	pm1gentype
N	BUSINESS_ENTITY	N	Number(9)	The business entity ID to which the offer is linked in the BOH	N	pmben
N	CURRENCY	N	Char(3)	The currency of the offer (for example, USD)	N	gn1currncy

PM1_PACKAGE_VERSION

This table contains package-specific information. Common version information is stored in the PM1_VV_ELEMENT table.

PK	Name	M	Type (length)	Functionality	Valid Values	Domain Name
Y	ID	Y	Number(9)	The ID of the pricing package	N	pm1id
Y	MAJOR_VERSION_ID	Y	Number(3)	Major version of the package	N	pm1subvrsid
N	FIX_VERSION_ID	Y	Number(3)	Fix version of the package	N	pm1subvrsid
N	PRIORITY	Y	Number(5)	Priority of the package	N	pmpriority
N	CRITERION	N	Varchar2(255)	Qualification Criteria name of the package	N	pm1name255

PM1_PC_VERSIONS

This table stores data regarding Product Catalog versions. It includes the Product Catalog version, effective and expiration dates, version type (relative or absolute), and version status (active or inactive).

PK	Name	M	Type (length)	Functionality	Valid Values	Domain Name
Y	VERSION	Y	Varchar2(9)	Product Catalog version	N	pm1version
N	EFFECTIVE_DATE	Y	Date(8)	Effective date of the Product Catalog version	N	date
N	EXPIRATION_DATE	N	Date(8)	Expiration date of the Product Catalog version	N	date
N	VERSION_TYPE	Y	Char(1)	Version type	N	pm1vertype
N	STATUS	N	Char(1)	Version status	Y (A,D,S)	pm1vrstatus

PM1_VV_ELEMENT

The Vertical Version Element table contains common parts of the vertically versioned element data. These include the ID, name, version major and fix numbers, expiration date, effective date, and based on relative version.

PK	Name	M	Type (length)	Functionality	Valid Values	Domain Name
Y	ID	Y	Number(9)	ID of the element	N	pm1id
Y	MAJOR_VERSION_ID	Y	Number(3)	Major version of the element	N	pm1subvrsid
N	FIX_VERSION_ID	Y	Number(3)	Fix version of the element	N	pm1subvrsid
N	EFFECTIVE_DATE	Y	Date(14)	Effective date of the element	N	datetime
N	EXPIRATION_DATE	N	Date(14)	Expiration date of the element	N	datetime
N	NAME	Y	Varchar2(255)	Name of the element version	N	pm1name255
N	BASE_ON_REL_VERS	Y	Varchar2(9)	IR version that the element is based on	N	pm1version
N	ELEMENT_TYPE	Y	Char(1)	The type of element: 'O' – Offer 'P' – Package 'I' - Item	N	pm1gentype
N	SEQUENCE_NUMBER	Y	Number(9)	Identifies the case in which an element of a future version that was already distributed has changed.	N	pmseqno

PM1_VV_RELATION

The table stores relations between Product Catalog elements.

PK	Name	M	Type (length)	Functionality	Valid Values	Domain Name
Y	PARENT_ID	Y	Number(9)	ID of the parent element	N	pm1id
Y	MAJOR_VERSION_ID	Y	Number(3)	Major version of the elements	N	pm1subvrsid
N	FIX_VERSION_ID	Y	Number(3)	Fix version of the elements	N	pm1subvrsid
Y	CHILD_ID	Y	Number(9)	ID of the child element	N	pm1id
N	CHILD_TYPE	Y	Char(1)	The type of the child: 'O' – Offer 'P' – Package 'I' - Item	N	pm1gentype
N	PARENT_TYPE	Y	Char(1)	The type of the parent: 'O' – Offer 'P' – Package 'I' - Item	N	pm1gentype
N	CHILD_ORDER	Y	Number(5)	The order in which the child elements have to appear	N	pmorder

Special Database Operations

No changes.

New Special Synonyms

No changes.

Database Area Objects

No changes.

Modified Oracle Tables

The following tables describe the modifications to Oracle tables in this version.

PERFORMANCE_INDICATOR

PK	Name	M	Type (length)	Functionality	Valid Values	Domain Name
N	RESERVATION_DATA	N	Blob(3200)	When using PI reservation, this field holds the reserved amount for each of the reserve-enabled PI attributes.	N	pm1blob

PM1_CYCLE_STATE

PK	Name	M	Type (length)	Functionality	Valid Values	Domain Name
Y	PARTITION_ID	Y	Number(4)	Partition ID of the cycle	N	gn1custpart

RATED_EVENT

PK	Name	M	Type (length)	Functionality	Valid Values	Domain Name
N	VERSION	N	Varchar2(9)	Version in which the event was created	N	pm1version
N	COMPRESS_ID	N	Number(9)	ID of the compression algorithm	N	pm1comprid
N	BA_NO	N	Number(12)	Billing arrangement number	N	pm1bano

REJECTED_EVENT

PK	Name	M	Type (length)	Functionality	Valid Values	Domain Name
N	VERSION	N	Varchar2(9)	Version of the event	N	pm1version
N	COMPRESS_ID	N	Number(9)	ID of the compression algorithm	N	pm1comprid
N	BA_NO	N	Number(12)	Billing arrangement number	N	pm1bano

Modified Objects

No changes.

New Entity Sizing

No changes.

Database Extract

No changes.

Domains

No changes.

Data Changes

This section describes the data changes in the current version.

Reference Data

No changes.

Application Data

No changes.

Security Data

No changes.

Operational Data

No changes.

TASK Connect Tables

The following table summarizes the changes to the Task Connect tables in this version:

Functionality	Task Name	Change Type	Connection code	Session arg	Domain	Description
Change connect code of Usage Queries server	OP1J2EESERVER	Change	Change USQ01PORT to USQ0000ENV	RPR1		Change connect code to USQ0000ENV. Add a new variable in addition to HOST, PORT. The new variable is PARTITION. It should be populated from task_name = RPR1PARTITION.
New connect code for Usage Queries server	OP1J2EESERVER	New	USQ0001ENV	RPR1		Create new connect code USQ0001ENV. Add a new variable in addition to HOST, PORT. The new variable is PARTITION. It should be populated from task_name = RPR1PARTITION.
New connect code for Usage Queries server	OP1J2EESERVER	New	USQ0002ENV	RPR1		Create new connect code USQ0002ENV. Add a new variable in addition to HOST, PORT. The new variable is PARTITION. It should be populated from task_name = RPR1PARTITION.
New connect code for Usage Queries server	OP1J2EESERVER	New	USQ0003ENV	RPR1		Create new connect code USQ0003ENV. Add a new variable in addition to HOST, PORT. The new variable is PARTITION. It should be populated from task_name = RPR1PARTITION.
New connect code for Usage Queries server	OP1J2EESERVER	New	USQ0004ENV	RPR1		Create new connect code USQ0004ENV. Add a new variable in addition to HOST, PORT. The new variable is PARTITION. It should be populated from task_name = RPR1PARTITION.

TimesTen Changes

This section describes the TimesTen changes in the current version.

New TimesTen Tables

The following tables describe new TimesTen tables added in this version.

PM1_CUSTOMER_OFFER_PARAMS

This table holds the customer offer parameter entities, which describe the customer parameters at the offer level.

PK	Name	M	Type (length)	Functionality	Valid Values	Domain Name
Y	SUBSCRIBER_ID	Y	Numeric(9)	Identifies the subscriber	N	
Y	OFFER_INSTANCE	Y	Numeric(9)	The instance of the subscriber's offer; this value is retrieved from the offer in customer offers that passed qualification	N	
Y	EFFECTIVE_DATE	Y	Timestamp	Effective date of the parameters	N	
N	EXPIRATION_DATE	N	Timestamp	Expiration date of the parameters	N	
N	SYS_CREATION_DATE	N	Timestamp	Creation date of the record	N	
N	SYS_UPDATE_DATE	N	Timestamp	Last update date of the record	N	
N	APPLICATION_ID	N	Char(6)	ID of the last application that updated the record	N	
N	DYNAMIC_DATA	N	Varchar(4000)	List of parameters	N	
N	TSTAMP	N	Binary(8)		N	

PM1_CYCLE_CHANGE

This table holds the change cycle requests that are made by the customers.

PK	Name	M	Type (length)	Functionality	Valid Values	Domain Name
Y	CUSTOMER_ID	Y	Numeric(9)	Identifies the customer that is associated with the request	N	
Y	EFFECTIVE_DATE	Y	Timestamp	Date from which the new cycle becomes effective (old cycle close date + 1)	N	
N	NEW_CYCLE_CODE	Y	Numeric(4)	New cycle code	N	
N	OLD_CYCLE_CODE	N	Numeric(4)	Changed cycle code (to support backward compatibility)	N	
N	STATUS	N	Char(1)	For future use	N	
N	SYS_CREATION_DATE	Y	Timestamp	Creation date of the record	N	
N	SYS_UPDATE_DATE	N	Timestamp	Last update date of the record	N	
N	APPLICATION_ID	N	Char(6)	ID of the last application that updated the record	N	
N	TSTAMP	N	Binary(8)		N	

Modified TimesTen Tables

The following tables describe the modifications to TimesTen tables in this version.

PERFORMANCE_INDICATOR

PK	Name	M	Type (length)	Functionality	Valid Values	Domain Name
N	RESERVATION_DATA	N	Varchar(3200)	When using PI reservation, this field holds the reserved amount for each of the reserve-enable PI attributes	N	

RATED_EVENT

PK	Name	M	Type (length)	Functionality	Valid Values	Domain Name
N	VERSION	N	Varchar(9)	The version in which the event was created	N	

REJECTED_EVENT

PK	Name	M	Type (length)	Functionality	Valid Values	Domain Name
N	VERSION	N	Varchar(9)	The version of the event	N	

Operational Job Definition Changes

This section summarizes all the Operational jobs that are new or changed in this version.

Operational Jobs

No changes.

Operational Parameter Tables

No changes.

Job Dependencies

No changes.

By-Request Jobs

No changes.