Oracle® Communications

Specification

User Data Repository Provisioning Database Application and Interface Specification for Release 14.0.0.0.0

F79367-01

April 2023

ORACLE®

CAUTION: Use only the Installation procedure included in the Install Kit.

Before installing any system, access My Oracle Support (https://support.oracle.com) and review any Technical Service Bulletins (TSBs) that relate to this procedure.

My Oracle Support (https://support.oracle.com) is your initial point of contact for all product support and training needs. A representative at Customer Access Support (CAS) can assist you with My Oracle Support registration.

Call the CAS main number at 1-800-223-1711 (toll-free in the US), or call the Oracle Support hotline for your local country from the list at http://www.oracle.com/us/support/contact/index.html.

See more information on My Oracle Support, see Appendix D.

Oracle Communications User Data Repository Provisioning Database Application and Interface (PDBI) Specification, Release 14.0.0.0.0

F79367-01

Table of Contents

List of Figures

List of Tables

Error Codes

Chapter 1. Introduction

1.1 Purpose and Scope

This document presents the Provisioning Database Application and Interface to be used by provisioning client applications to administer the Provisioning Database of the Oracle Communications User Data Repository (UDR) system. Through PDBI, an external provisioning system supplied and maintained by the network operator may add, change, or delete subscriber information in the Oracle Communications User Data Repository database.

The primary audience for this document includes customers, Oracle customer service, software development, and product verification organizations, and any other Oracle personnel who have a need to use the PDBI.

1.2 References

The following external document references capture the source material used to create this document.

[1] EAGLE Application Processor Provisioning Database Interface User's Guide, EPAP, Release 16

1.3 Glossary

This section lists terms and acronyms specific to this document.

Table 1: Glossary

Acronym/Term	Definition
ACID	Atomic, Consistent, Isolatable, Durable
BLOB	Binary Large Object
CFG	Configuration Data—data for components and system identification and configuration
CPS	Customer Provisioning System
DP	Database Processor
FRS	Feature Requirements Specification
FTP	File Transfer Protocol
GUI	Graphical User Interface
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity, or IMSI [im-zee]
IP	Internet Protocol
KPI	Key Performance Indicator
MEAL	Measurements, Events, Alarms, and Logs
MP	Message Processor

MSISDN	Mobile Subscriber ISDN Number
NA	Not Applicable
NE	Network Element
NPA	Numbering Plan Area (Area Code)
NPHO	Non Pool Host UDR
OAMP	Operations, Administration, Maintenance, and Provisioning
NOAMP	Network OAM and Provisioning
PCRF	Policy Charging and Rules Function
PS	Provisioning System
PSO	Pool Spanning UDRs
SDO	Subscriber Data Object
SEC	Subscriber Entity Configuration
SNMP	Simple Network Management Protocol
SOAM	System Operation, Administration, and Maintenance
SPR	Subscriber Profile Repository
TCP	Transmission Control Protocol
UDR	User Data Repository
UTC	Coordinated Universal Time
VIP	Virtual IP
XML	Extensible Markup Language
FABR	Full Address Based Resolution
MNP	Mobile Number Portability

Chapter 2. System Architecture

2.1 Overview

Oracle Communications User Data Repository (UDR) performs the function of an SPR, which is a database system that acts as a single logical repository that stores subscriber data. The subscriber data that traditionally has been stored into the HSS, HLR, AuC, or Application Servers is stored in UDR as specified in 3GPP UDC information model [3]. UDR facilitates the share and the provisioning of user related data throughout services of 3GPP system.

The data stored in UDR can be permanent and temporary data. Permanent data is subscription data and relates to the required information the system needs to perform the service. User identities (for example, MSISDN, IMSI, IMEI, NAI and AccountId), service data (for example, service profile) and authentication data are examples of the subscription data. This kind of user data has a lifetime as long as the user is permitted to use the service and may be modified by administration means. Temporary subscriber data is dynamic data which may be changed as a result of normal operation of the system or traffic conditions (for example, transparent data stored by Application Servers for service execution, user status, usage, and so on).

Oracle Communications User Data Repository is a database system providing the storage and management of subscriber policy control data for vSTP nodes. MNP subscriber data is created/retrieved/modified or deleted through the provisioning. The following subscriber data is stored in Oracle Communications User Data Repository:

- Subscriber
    - Profile
    - MnpDataGRN
    - MnpDataSPRN

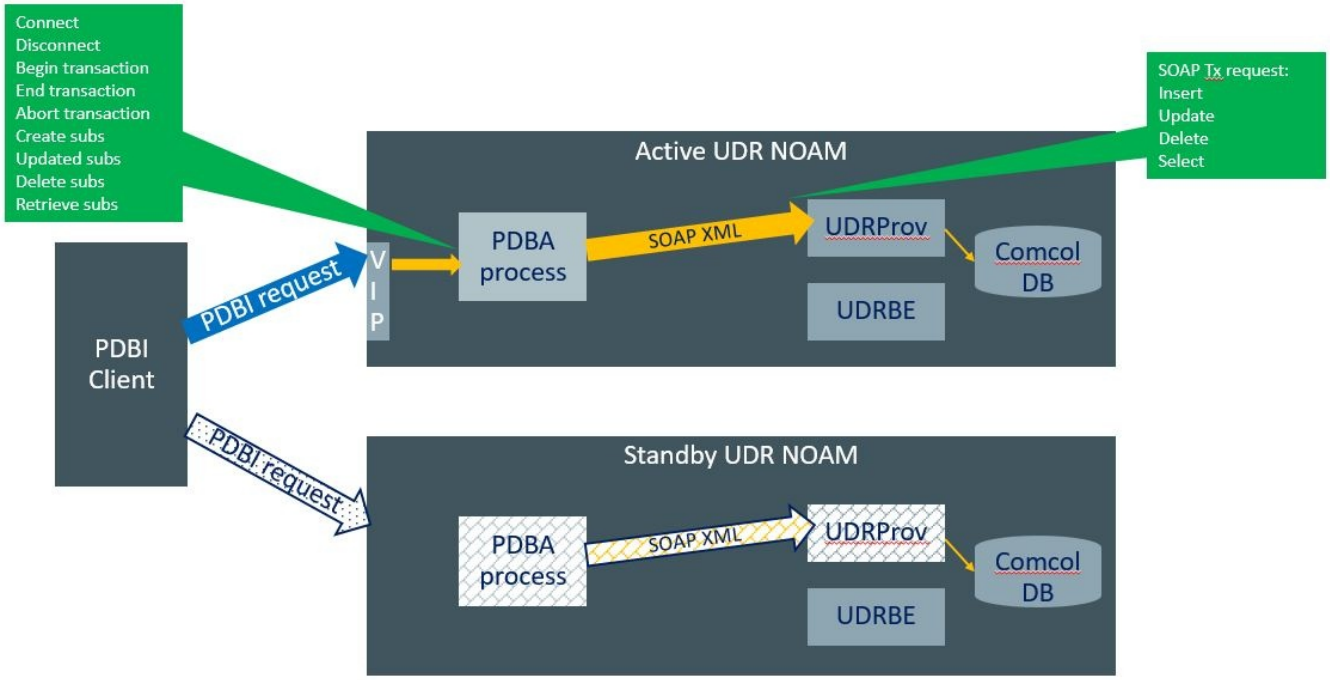
The Figure 1 below illustrates a high level the Oracle Communications User Data Repository Architecture.

Oracle Communications User Data Repository has the Network level OAMP server (NOAMP) in the architecture provides the provisioning, configuration and maintenance functions for all the network elements under it.

2.2 Provisioning Interface

The PDBI provides data manipulation commands for MNP based subscriber.

Figure 1: User Data Repository



2.2.1 Subscriber Commands

- MNP Subscriber profile create, retrieve, and delete
- MNPData create and delete
    1. MnpDataGRN
    2. MnpDataSPRN

2.3 Provisioning Database Application Server (PDBA)

The application in the provisioning process interfacing to PDBI provisioning clients runs on every active NOAMP server. The PDBA is responsible for:

- Accepting and authorizing PDBI provisioning client connections
- Processing and responding to PDBI requests received from provisioning clients
- Performing provisioning requests using the underlying SOAP on the NOAMP.

2.4 Provisioning Clients

The PDBA provides connections to the customer provisioning systems (CPS). These are independent information systems supplied and maintained by the network operator to be used for provisioning the UDR system. Through PDBA, the CPS may add, delete, change or retrieve information about any subscriber.

CPSs use PDBI to send requests to manipulate and query data in the Provisioning Database. Provisioning Clients establish TCP/IP connections to the PDBA running on the active NOAMP using the VIP for the primary NOAMP.

Provisioning clients need to re-establish connections with the PDBA using the VIP for the primary UDR after switchover from the active server for the primary to the standby UDR server. Provisioning clients also need to redirect connections to the VIP for the secondary after switchover from the primary UDR site to the disaster recovery UDR site.

Provisioning clients must run a timeout for the response to a request, if a response is not sent. If a response is not received, the client drops the connection and re-establishes it before trying again.

Provisioning clients are expected to re-send requests that resulted in a temporary error, or for which a response was not received.

2.5 Security

The following forms of security are provided for securing connections between the PDBI and provisioning clients in an unsecure/untrusted network:

- Client Server IP Address White List

2.5.1 Client Server IP Address White List

For securing connections between the PDBI and provisioning clients in an unsecure/untrusted network, a list of authorized IP addresses is provided.

The system configuration process maintains a white list of server IP addresses and/or IP address ranges from which clients are authorized to establish a TCP/IP connection from.

The PDBA verifies provisioning connections by utilizing the authorized IP address list. Any connect request coming from an IP address that is not on the list is denied (connection is immediately closed). All active connections established from an IP address, which is removed from the Authorized IP list, are immediately closed.

2.6 Multiple Connections

Multiple provisioning systems may be connected via PDBI simultaneously. All systems can open read transactions, but only one system at a time can open a write transaction. If more than one system requests a write transaction, contention for write access will be handled as follows:

1- The first system to submit a write request will be granted access, if it is authorized for write access.

2- If a second system submits a write request while the first transaction is still open, it either will be immediately rejected with WRITE_UNAVAIL return code, or will be queued for a specified time out period to wait on the first system's write transaction to complete.

3- The time out period mentioned above may be specified by the user in the write transaction request, and can be any value from 0 to 3600 seconds. If the value is not included or is set to 0, the second write request will be immediately rejected with WRITE_UNAVAIL return code.

4- If the time out value is set to any non-zero value, the second request will be held for that time period before being rejected. If the first user releases the write transaction before the second user's time out period has expired, the second user will then be granted write access.

5- If a third user submits a write request after the second user with a specified time out period, the third user's request will be queued behind the second user's request. Once the first user releases the transaction, the second user is granted access. After the second user releases the transaction, the third user is granted access and so forth. Of course, if any user's time out period expires, their request will be immediately rejected with WRITE_UNAVAIL return code.

6- If the third user sets a time out period longer than the second user, and the second user's time out period expires before the first user releases the transaction, the second user's request will be dropped

from the queue and the third user will move up in the queue. Thus, if the first user then releases the transaction before the third user's time out has expired; the third user will be granted access

NOTE: In order to achieve the maximum provisioning TPS rate that the UDR PDBI is certified for, multiple simultaneous provisioning connections are required.

2.7 Request Queue Management

If multiple clients simultaneously issues requests, each request is queued and processed in the order in which it was received on a per connection basis. The client is not required to wait for a response from one request before issuing another.

Incoming requests, whether multiple requests from a single client or requests from multiple clients, are not prioritized. Multiple requests on multiple connections from a single client are handled on a first-in, first-out basis. Generally, requests are answered in the order in which they are received, but this is not always guaranteed. A client sends a number of valid update requests, which are performed, and run in the order they are received. If the client were to then send an invalid request (such as if the XML could not be parsed) on a different connection, this is responded to immediately, potentially before the any, some, or all of the previous requests have been responded to.

NOTE: All requests from a client sent on a single connection are processed by UDR serially. Multiple requests can be sent without receiving a response, but each request is queued and not processed until the previous request has completed. A client can send multiple requests across multiple connections, and these may run in parallel (but requests on each connection are processed serially).

2.8 Database Transactions

Each create/update/delete request coming from PDBI triggers a unique database transaction, that is, a database transaction started by a request is committed before sending a response.

2.8.1 Transaction Mode

PDBA servers support the following database transactions modes:

- Normal Transaction Mode (default)
- Single Transaction Mode

The PDBI client specifies a transaction mode when establishing a connection with PDBA.

Normal Database Transaction Mode:

The default mode, called normal database transaction mode requires an explicit begin_txn request paired with end_txn or abort_txn request to complete the transaction.

In normal database transaction mode:

1- Many updates can be sent and committed to the database all at once when the transaction is completed. This results in a much faster rate of updates per second.

2- Transaction integrity is ensured by allowing updates to be aborted or rolled back if there is an unexpected failure before the transaction is completed. Updates are not committed to the database until the end_txnrequest is issued. If an unexpected failure occurs, or if the transaction is manually

aborted by the abort_txnrequest, the database will be maintained in the state it was in prior to the beginning of the transaction.

3- A write transaction can only be opened by one client connection at a time. Thereby, preventing multiple clients from updating the database at the same time. If a write transaction was not used, a client could do a retrieve of a subscription and get one answer, and then do the exact same retrieve and get a different answer if a different connection had modified or deleted the subscription being queried.

4- The maximum number of database manipulation requests that can be contained within a transaction is user configurable via OCUDR GUI. If the number of database manipulation requests exceeds the configurable limit, each subsequent request within the transaction will fail with TXN_TOO_BIG return code.

Single Database Transaction Mode

Single database transaction mode implicitly begins and ends a transaction for each individual update request.

In single database transaction mode:

1- database manipulation and query requests are sent without being enclosed by begin_txn and end_txnrequests.
2- the timeout parameter on all database manipulation requests is enabled to allow the client to specify how long to wait for the write transaction (if already in use by another connection) to complete before the request is rejected with WRITE_UNAVAL return code (exactly like the begin_txn request does). Database manipulation requests' timeout parameter is not enabled for specification in normal transaction mode.

2.8.2 ACID-Compliance

The PDBI supports Atomicity, Consistency, Isolation and Durability (ACID)-compliant database transactions, which guarantee transactions, are processed reliably.

Atomicity

Database manipulation requests are atomic. If one database manipulation request in a transaction fails, all of the pending changes can be rolled back by the client, leaving the database as it was before the transaction was initiated. However, the client also has the option to close the transaction, committing only the changes in that transaction, which were run successfully. If any database errors are encountered while committing the transaction, all updates are rolled back and the database is restored to its previous state.

Consistency

Data across all requests performed inside a transaction is consistent.

Isolation

Not all database changes made in a transaction by one client are viewable by any other clients until the changes are committed by closing the transaction. In other words, all database changes made in a transaction cannot be seen by operations outside of the transaction.

Durability

After a transaction has been committed and become durable, it persists and not be undone. Durability is achieved by completing the transaction with the persistent database system before acknowledging commitment. Provisioning clients only receive success responses for transactions that have been successfully committed and have become durable.

The system recovers committed transaction updates in spite of system software or hardware failures. If a failure (loss of power) occurs in the middle of a transaction, the database returns to a consistent state when it is restarted.

Data durability signifies the replication of the provisioned data to different parts of the system before a response is provided for a provisioning transaction. The following additive configurable levels of durability are supported:

- Durability to the disk on the active provisioning server (just 1)
- Durability to the local standby server memory (1 + 2)
- Durability to the active server memory at the Disaster Recovery site (1 + 2 + 3)

2.9 Connection Management

It is possible to enable/disable/limit the PDBI in a number of different ways.

2.9.1 Connections Allowed

The configuration variable `Allow PDBI Connections` controls whether PDBI connections are allowed to the configured port. If this variable is set to `NOT_ALLOWED`, then all existing connections are immediately dropped. Any attempts to connect are rejected.

When `Allow PDBI Connections` is set back to `ALLOWED`, the connections are accepted again.

2.9.2 Disable Provisioning

When the Oracle Communications User Data Repository GUI option to disable provisioning is selected, existing connections remain up and new connections are allowed. However, any provisioning request that is sent is rejected with a SERVICE_UNAVAILABLE error indicating the service is unavailable.

For an example of a provisioning request/response when provisioning is disabled, request will not be processed.

2.9.3 Idle Timeout

TCP connection between Provisioning client and PDBA is handled persistent fashion. It indicates the time to wait before closing the connection due to inactivity (requests are not received). It is 60 seconds and not configurable.

2.9.4 Maximum Simultaneous Connections

The maximum number of simultaneous PDBI client connections are 128. If an attempt is made to connect more than the number of PDBI connections allowed, the PDBI server rejects the connection.

2.9.5 TCP Port Number

The configuration variable "`tcpListeningPort`" defines the PDBI interface TCP listening port.

Default TCP port for PDBA is 5873 which is configurable in ProvOptions table.

2.10 Behavior during Low Free System Memory

If the amount of free system memory available to the database falls below a critical limit, then requests that create or update data may fail with the error MEMORY_FULL. Before this happens, memory threshold alarms are raised indicating the impending behavior if the critical level is reached.

The error returned by the PDBI when the critical level has been reached is:

```
INTERNAL_ERROR
```

2.11 Congestion Control

If UDR starts to encounter congestion (based on high CPU usage), then based on the congestion level, UDR rejects some requests (based on the reqname, see section 5.1.10).

- If the minor CPU usage threshold is crossed (CL1), then UDR rejects select requests
- If the major CPU usage threshold is crossed (CL2), then UDR rejects select, update, operation, and tx (transaction) requests
- If the critical CPU usage threshold is crossed (CL3), then UDR rejects all requests

The error returned by the PDBI when a request is rejected due to congestion is:

```
INTERNAL_ERROR
```

Chapter 3. PDBI Description

Oracle Communications User Data Repository supports a command line based provisioning interface for management of subscriber data. This interface supports querying, creation, modification and deletion of MNP based subscriber data. The PDBI Messages and PDBI Replies are transported over the TCP protocol.

Each PDBI Message/Reply contains an UDR format CSV request/response. The following CSV request types are supported:

- Update
- Insert
- Delete
- Select

A PDBI provisioning client application is responsible for:

- Establishing a TCP/IP connection with the PDBA server using the VIP for the primary UDR and the PDBA listening port (as specified in section 2.9).
- Creating and sending PDBI request messages (as specified in section 5.2.1) to the PDBA server.
- Receiving and processing PDBI response messages (as specified in section 5.2.2) received from the PDBA server.
- Detecting and handling connection errors. It is recommended that the TCP keep-alive interval for the client on the TCP/IP connection is set to detect and report a disconnection problem promptly.

Chapter 4. UDR Data Model

The UDR is a system used for the storage and management of subscriber policy control data. The UDR functions as a centralized repository of subscriber data for the PCRF.

The subscriber-related data includes:

- Profile and subscriber data

  Pre-provisioned information that describes the capabilities of each subscriber. This data is typically written by the OSS system (via a provisioning interface) and referenced by the PCRF (via the Sh interface).

The data architecture supports multiple Network Applications. This flexibility is achieved though implementation of a number of registers in a Subscriber Data Object (SDO) and storing the content as Binary Large Objects (BLOB). An SDO exists for each individual subscriber.

The Index contains information on the following:

- Subscription
  - A subscription exists for every individual subscriber
  - Maps a subscription to the user identities through which it can be accessed
  - Maps an individual subscription to the pool of which they are a member
- User Identities

  Use to map a specific user identity to a subscription

  - IMSI, MSISDN,IMEI, NAI and AccountId map to an individual subscription
  - SPRNID/GRNID maps to a SPRN/GRN entity

The Subscription Data Object (SDO):

- An SDO record contains a list of registers, holding a different type of entity data in each register
- An SDO record exists for:
  - Each individual subscriber

    Defined entities stored in the registers are:

    - Profile
    - MNP Data

Provisioning applications can create, retrieve, modify, and delete subscriber data. The indexing system allows access to the SDO via IMSI, MSISDN, IMEI, NAI or AccountId.

A field in an entity can be defined as mandatory, or optional. A mandatory field must exist, and cannot be deleted.

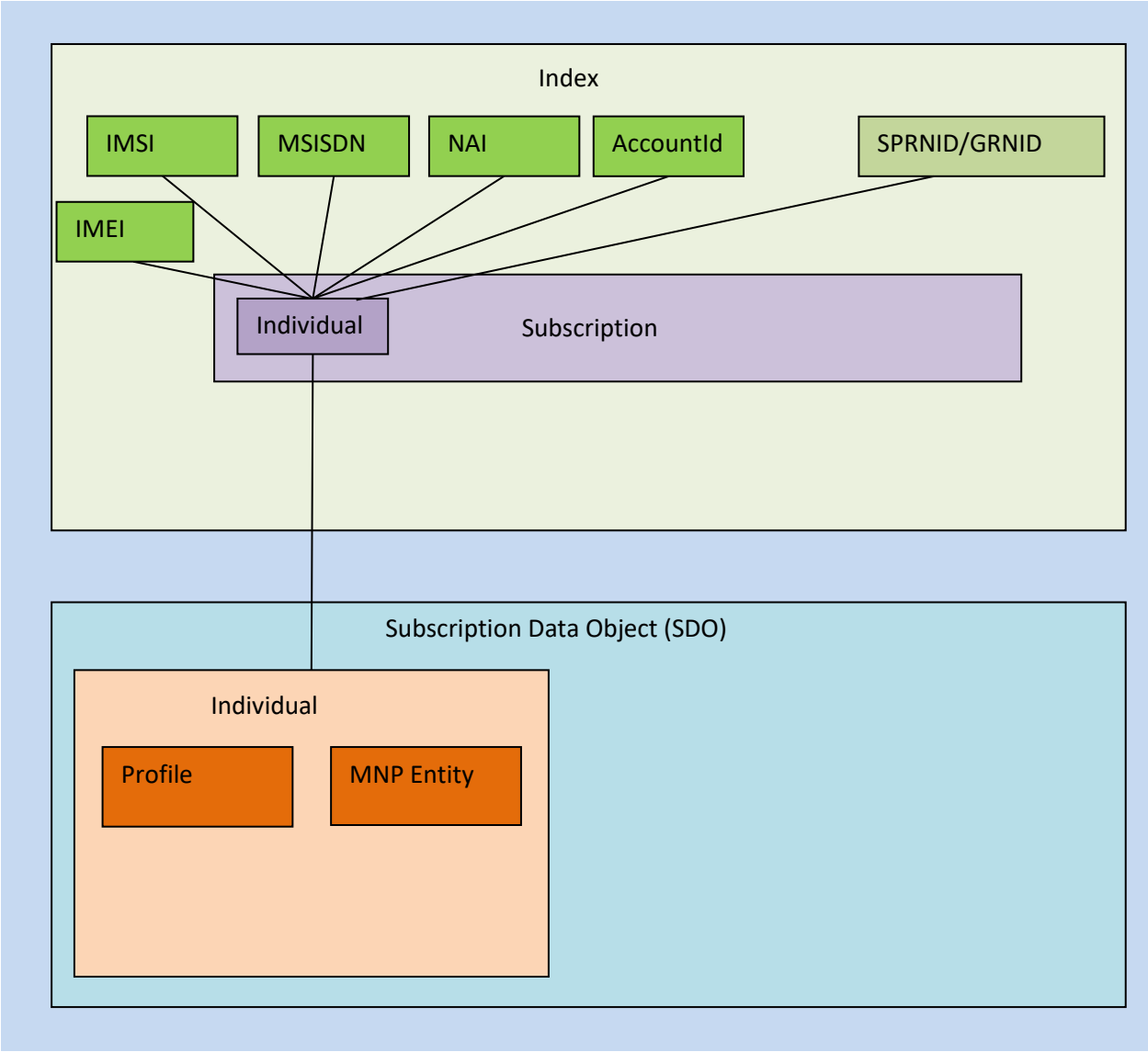
A field in an entity can have a default value. If an entity is created, and the field is not specified, it are created with the default value.

A field in an entity can be defined so that after it is created, it cannot be modified. Any attempt to update the field after creation fails.

A field in an entity can have a reset value. If a reset command is used on the entity, fields with a defined reset value are set to the defined value. This is only applicable to field values in a row for the MNP entity.

NOTE: This section describes the default UDR data model as defined in the Subscriber Entity Configuration (SEC). The data model can be customized via the UDR GUI.

Figure 2: Data Model



4.1 Subscriber Data

4.1.1 MNP Subscriber Profile

The Subscriber profile represents the identifying attributes associated with the user.

The subscriber profile supports the sequence of attributes in Table 4. Each record must have at least one of the key values: MSISDN, IMSI.

UDR only supports an MSISDN with 8 to 15 numeric digits. A preceding + (plus) symbol is not supported, and is rejected.

Table 2 MNP Subscriber Profile Entity Definition

Name(XML tag)	Type	Description
MSISDN	String	List of MSISDNs (8 to 15 numeric digits). A separate entry is included for each MSISDN associated with the profile for the subscriber.
IMSI	String	List of IMSIs (10 to 15 numeric digits). A separate entry is included for each IMSI associated with the profile for the subscriber.
PType (Portability Type)	Integer	Allowed values are 0 to 40.
CLDBL(Called Black List)	Integer	Allowed values are either 0 or 1
CLNBL(Calling Black List)	Integer	Allowed values are either 0 or 1
ASD	String	Any string that can be used to identify the ASD for the subscriber (1 to 10 characters). Allowed values are any ASCII printable character, values x20 to x7e.
GRNID	String	Any string that can be used to identify the GRNID for the subscriber (1 to 21 characters). Allowed values are any ASCII printable character, values x20 to x7e.
SPRNID	String	Any string that can be used to identify the SPRNID for the subscriber (1 to 21 characters). Allowed values are any ASCII printable character, values x20 to x7e.
Split	Integer	Allowed values are either 0 or 1 **Note: This field is only applicable for MNP range subscriber with split feature not for normal MNP subscriber**

4.1.2 MNP Data

MNP data entity are two type: MnpDataGRN and MnpDataSPRN. The MnpDataGRN and MnpDataSPRN entities are associated with a MNP based subscriber(s). These entities have field defined in table 15 and 16 below.

Table 3 MnpDataGRN Entity Definition

Name(XML tag)	Type	Description
EDigit	String	Entity Digits values are 1-15 digits.
RI	Integer	Allowed values are either 0 or 1 for routing indicator field.
PC	String	Point Code values are 0-32 characters.
PCDom	String	Point Code Domain values are 0-7 characters.
SSN	Integer	Allowed values are from 2-255 for SSN field.
SRFIMSI	String	SRFIMSI values are 5-15 digits.

DigAct	String	Digit Action values are 0-40 characters.

Table 4 MnpDataSPRN Entity Definition

Name(XML tag)	Type	Description
Type	String	Type values are 0-2 characters. E.g. RN
EDigit	String	Entity Digits values are 1-15 digits.
RI	Integer	Allowed values are either 0 or 1 for routing indicator field.
PC	String	Point Code values are 0-32 characters.
PCDom	String	Point Code Domain values are 0-7 characters.
SSN	Integer	Allowed values are from 2-255 for SSN field.
SRFIMSI	String	SRFIMSI values are 5-15 digits.
DigAct	String	Digit Action values are 0-40 characters.

Chapter 5. PDBI Message Definitions

5.1.1 Message Conventions

PDBI message specification syntax follows several conventions to convey what parameters are required or optional and how they and their values must be specified.

Table 5 Message Conventions

Symbol	Description
<>	Angle brackets are used to enclose parameter values, which are italicized. These values can be choices or names. For example, inParameter1 <1|2|3>, the numbers represent specific value choices. In Parameter2 <ServerName>, ServerName represents the actual value.
[]	Square brackets are used to enclose an optional parameter and its value, such as [, Parameter1 <1|2|3>]. A parameter and its value that are not enclosed in square brackets are mandatory.
|	When the pipe symbol is used in a parameter value list, such asParameter1 <1|2|3>, it indicates a choice between available values.
,	A comma is used to separate each parameter that is specified.

5.1.2 Message Instance Identifier (IID)

Each message has an Instance Identifier called the iid as its first parameter. The PDBI client to correlate request and response messages uses the iid parameter. The iid parameter is optional and if specified, is an integer between 1 and 4294967295 expressed as a decimal number in ASCII. If the user specifies the iid parameter in a request, the same iid parameter and value is returned by PDBA in the corresponding response. A unique iidvalue should be specified in each request message.

5.1.3 Request Messages

PDBA and PDBI support the request messages listed in Table. Unsupported requests are rejected with a PARSE_FAILED return code. All errors encountered while processing requests are recorded in PDBA's Command and Trace logs. PDBI clients are to construct request messages as specified in the sections referenced in Table 12

Table 6 PDBI Requests

PDBI Request	Description	Section
`connect`	Establish PDBI Connection Characteristics with PDBA	
`disconnect`	Terminate PDBI Connection with PDBA	
`begin_txn`	Begin Database Transaction	
`end_txn`	End Database Transaction	

abort_txn	Abort Database Transaction	
ent_sub	Enter Subscription Data	
upd_sub	Update Subscription Data	
dlt_sub	Delete Subscription Data	
rtrv_sub	Retrieve Subscription Data	

5.1.4 Response Messages

The rsp response message is sent by PDBA in response to a PDBI request. The syntax of the response messages is the same for all PDBI requests and is as follows:

```
rsp([iid <1..4294967295>,] rc #### [, data (... )])
```

Parameters:

iid (Optional) The Instance Identifier of the corresponding request if it was provided. Values: 1–4294967295

rc Return Code. Whether or not the client requested operation was successfully executed by PDBA. A mapping between the return code labels described in this section and the real integer value can be found in Appendix A – PDBI Response Message Return Codes. Values: 0 (success) non-zero (failure)

data (...) (Optional) Request-specific response data. Each response defined will declare what errors it returns (along with their meanings) and what the data section should look like for each error. If a response does not require a data section (meaning it is just a simple ack or nak), then data section will not be present

Common Response Messages

The response messages described in this section are common, in that they can be returned for all supported requests. For simplicity, these responses are not repeated in the response section of each supported request. Response Message Return Codes that are listed in the response section of each supported request are specific for that request. See Section 5.1 for an exhausted list of Response Message Return Codes.

PDBI Not Connected (NOT_CONNECTED)

Response The NOT_CONNECTED response indicates that the client has established a TCP/IP socket connection, however, it has not issued a connect() request. Once a PDBI client has established a TCP/IP socket connection with PDBA on its well known listening port (5873), it must issue the connect() request to complete the connection. This response does not contain a data section.

Parse Failed (PARSE_FAILED)

Response Since this is a string based API, all requests have the possible return code of PARSE_FAILED. With this special case response, it is possible that a data section will be present in the response to give more information about the parse failure. If the data section exists, there are two different optional parameters possible. The first

parameter is a reason text string stating explicitly what was wrong with the request. The second parameter is a location string that will contain the portion of the original request that contained the error with curly braces around the place where the error occurs. If no specific information was available, then the data section will not be present in the response.

```
data (reason "reason text", [location "XXXXXXX{}XXXXXXXXX"])
```

The possible reasons for parse failures are listed in the table below.

Table 7 Possible Reasons for Parse Failure

Reason	Description
Unknown request verb	The request verb did not match any of the known commands.
Space required	A white space character is missing after some element of the request.
Missing paren	An opening or closing parenthesis is missing.
Invalid value	An invalid value was provided for one of the specified parameters.
[parameter] parameter expected	A mandatory parameter is missing or a parameter is expected following a comma.
Unknown parameter	An unknown parameter was found.
Missing comma	A comma was missing after a parameter.
Value expected	A parameter label was found with no value following it.
Duplicate parameter	Multiple instances of a non-repeatable parameter was found

Bad Arguments (BAD_ARGS) Response

The BAD_ARGS response is similar to the PARSE_FAILED response in that it indicates a problem with the request received. The difference is that PARSE_FAILED indicates mostly syntactical problems with the request. BAD_ARGS is used more to indicate semantic problems with the request. It would be used to indicate missing mandatory parameters or illegal parameter combinations. The data section of a BAD_ARGS response will have a reason string that indicates what problem was encountered.

```
data (reason "<reason text>")
```

Invalid Parameter Value (INVALID_VALUE) Response

The INVALID_VALUE is used to indicate that one of the fields specified in the request had an invalid value. The data section of an INVALID_VALUE response will have a parameter string that indicating the offending field that was encountered and optional reason text.

```
data (param <field label> [<value - reason text>])
```

Internal Error (INTERNAL_ERROR) Response

The INTERNAL_ERROR is used to indicate that the PDBI server has encountered an internal error and was unable to complete the request. The data section of an INTERNAL_ERROR response will have a string that states “Please Contact Tekelec Technical Services.” More information regarding this error can be found in PDBA’s trace log.

```
data (“Please Contact Tekelec Technical Services.”)
```

Database Exception (DB_EXCEPTION) Response

The DB_EXCEPTION is used to indicate that PDBI server has encountered a database exception and as a result is unable to complete the request. The data section of a DB_EXCEPTION response will have a string that will provide information regarding the database exception encountered.” If the DB_EXCEPTION response is received while committing a transaction, the entire transaction is rolled back. Information regarding the database exception can be found in PDBA’s trace log. Tekelec Technical Services should be contacted.

```
data (reason “errid:Error Description”)
```

5.1.5 Connect

Once a PDBI client has established a TCP/IP socket connection with PDBA, it must issue the connect request. The connection will automatically be terminated if the connect request is not received within 5 seconds of establishing a TCP/IP socket connection with PDBA. The Connection Init Timeout PDBI Option (configurable in ProvOptions table) specifies the time lapse allowed between a TCP/IP socket connection being established and a connect request being sent to PDBA. NOTE: PDBA supports up to a maximum of 128 simultaneous remote PDBI connections.

Request

```
connect([iid XXXXX] [, version 1.0] [, rspsize <1..32>] [, endchar <null|newline>]
[, idletimeout <none|0..44640>] [, txnmode <normal|single>])
```

Parameters:

`version` (Optional) Version of PDBI API this client application is using. It is used to decide whether or not the client application will be able to successfully communicate with the PDBA. Values: 1.0

`rspsize` (Optional) The maximum size (in K bytes) of responses that the PDBA will send back. This is useful for ensuring that retrieve requests that result in a large number of instances being returned come back in manageable sized responses to the client. Specifying rspsize overrides the Maximum Response Size PDBI Option configurable via EXHR GUI [5]. Values: 1–32 (default = 4)

endchar (Optional) Character that PDBA is to use to terminate responses. Values: null Terminate responses with a NULL (or \0) character (default). newline Terminate responses with a newline (or \n) character.

idletimeout (Optional) Number of minutes the connection can be idle before it is automatically terminated by PDBA. Values: 0Connection is never terminated for idleness (default) 1–44640Connection is terminated after specified idle minutes.

txnmode (Optional) Database Transaction mode. Values: normal All updates must be sent inside a transaction explicitly begun and ended by begin_txn and end_txn/abort_txn requests respectively. (default) single transactions are implicitly begun and ended for individual update requests.

Response

The connect response indicates whether or not PDBA is capable and willing to accept a new connection.

Error Codes 1: Connect

Return Code	Description	Data Parameter
SUCCESS	Connection accepted.	The assigned connection id and active status are returned. data (connectId <0...4294967295>, side <active|standby>)
UNKNOWN_VERSION	The specified version of PDBI API is unknown/unsupported.	None
ALREADY_CONNECTED	A connect request was sent on a socket that already had an established connection.	None
CONNECTION_DENIED	Interface is disabled. Connections are not being accepted.	data (reason "Interface Disabled")
	All available connections are in use by other clients. No more connections are being accepted.	data (reason "Too many connections")
	Connections are not permitted to standby PDBA.	data (reason "Connection to Standby PDBA not permitted")

5.1.6 Disconnect

Request

The disconnect request terminates the PDBI connection with PDBA. PDBA cleans up all data stored for the connection and then terminates the connection. If the client has a transaction open, it will automatically be aborted and any updates in the transaction will be backed out. Note: PDBA behavior is the same if the client neglects to send this request and just closes the socket or if the client abnormally terminates and the operating system closes the socket.

```
disconnect([iid XXXXX])
```

Response

The `disconnect` response indicates whether any errors were encountered during the termination of the connection. All disconnect requests results in the connection being terminated. Non-SUCCESS return codes simply indicate the error PDBA encountered prior to terminating the connection.

Error Codes 2: Disconnect

Return Code	Description	Data Parameter
`SUCCESS`	Connection terminated w/o any errors.	None
`ACTIVE_TXN`	An active transaction was aborted prior to terminating the connection.	None

5.1.7 Begin Transaction

Request

The `begin_txn` request begins a database transaction. All data manipulation and query requests (enter, update, delete, and retrieve) must be sent within the context of a transaction when connected in normal transaction mode. A client connection can only have one transaction open at a time. A database transaction can be opened for either read (read-only) or write (read/write) access.

Read database transactions:

- only accept database status and query requests. Database manipulation requests (enter, update, and delete) are rejected with `NO_WRITE_PERMISSION` return code.
- may be opened by multiple client applications at the same time
- query responses are sent back to the client immediately. There is no need to end a read transaction until the client is done sending query requests.

Write database transactions:

- opened only on connections to PDBA on the active server of the Primary NOAMP cluster. Attempts to open a write transaction when connected to PDBA on other NOAMPs will fail with a `STANDBY_SIDE` return code. PDBI clients should always use the primary NOAMP cluster's XMI VIP when connecting to PDBA. Primary NOAMP cluster's XMI VIP always connects the client to PDBA on the Active NOAMP.
- opened only by one client at a time. If a write transaction is already opened by another client, the `begin_txn` request is rejected immediately with `WRITE_UNAVAIL` or is queued up for the time specified by the `timeout` parameter. If the `timeout` parameter is specified with a non-zero value and that period of time elapses before the write transaction is opened, the `begin_txn` request is rejected with `WRITE_UNAVAIL`.
- accepts database manipulation and query requests (enter, update, delete, and retrieve)
- data manipulation requests are evaluated for validity and applied to a local database view which is a virtual representation of the main database plus local modifications made within the active transaction

- local database view changes are not committed to the main database until the transaction is ended with a `end_txn` request.
- query responses are sent back to the client immediately. However, the data returned will reflect any updates that have been made by the transaction, but not yet committed. If the transaction is aborted, then the data retrieved from the query may no longer be valid.
- can be aborted and rolled back with an `abort_txn` request anytime before the transaction is ended with an `end_txn` request.

```
begin_txn([iid XXXXX,] type <read|write> [, timeout <0..3600>])
```

Parameters:

`type` Type of transaction to open.

Values: read (read-only), query requests only

write (read/write), data manipulation and query requests

`timeout` (Optional) The amount of time (in seconds) to wait to open a write transaction if another connection already has one open. Clients waiting to open a write transaction will be processed in the order that their begin_txn requests were received. Values: 0 (return immediately if not available) to 3600 seconds (default is 0)

Response

Error Codes 3: Begin Transaction

Return Code	Description	Data Parameter
SUCCESS	Transaction successfully opened.	None
PROV_PROHIBITED	Provisioning has been manually disabled.	data (reason "Provisioning manually disabled")
NO_WRITE_PERMISSION	The PDBI client making the connection does not have write access permissions.	None
WRITE_UNAVAIL	Another client already has a write transaction open. This will only be returned to clients who do have write access permissions.	The IP address information of the client that already has the write transaction. data (id <0...4294967295>, ip <ip addr>, port <0...65635>

ACTIVE_TXN	A read or write transaction is already open on this connection.	None

5.1.8 End Transaction

Request

end_txn request ends an active database transaction.

If the currently opened transaction is a read transaction, ending the transaction really does not do anything except inform PDBA that you are done making queries. There are no database commits, etc.

If the currently opened transaction is a write transaction that had one or more successful updates, then ending the transaction will cause all the database changes to be committed. The new database level will be returned in the data section of the response. It is very important to understand that all previous updates, even though they received a successful return code, are not committed to the database until the end_txn request is received. If none of the updates were successful, a NO_UPDATES code is returned. If one or more data manipulation request were successful, a SUCCESS return code is returned.

```
end_txn([iid XXXXX])
```

Response

The end_txn response returns the results of ending a database transaction.

Error Codes 4: End Transaction

Return Code	Description	Data Parameter
SUCCESS	Database transaction was committed successfully. If the transaction type was write, then all database changes were committed successfully.	If the transaction type was write, then the new database level is returned. Otherwise, no data parameter is returned.
PROV_PROHIBITED	Provisioning has been manually disabled.	data (reason "Provisioning manually disabled")
NO_ACTIVE_TXN	There was no currently active transaction for this connection.	None
NO_UPDATES	The write transaction had no successful updates. No database change will occur and no new database level will be given.	None

5.1.9 Abort Transaction

Request

The `abort_txn` request aborts a database transaction. If the transaction to be aborted is a write transaction, all updates are rolled back prior to closing the transaction. If the transaction to be aborted is a read transaction, the transaction is simply closed.

Note: Sending an `abort_txn` request while receiving responses from previous query requests will not cause the query responses to stop.

```
abort_txn([iid XXXXX])
```

Response

The `abort_txn` response returns the results of aborting a database transaction.

Error Codes 5: Abort Transaction

Return Code	Description	Data Parameter
`SUCCESS`	Transaction aborted successfully	None
`NO_ACTIVE_TXN`	There was no currently active transaction for this connection.	None

5.1.10 Enter Subscription Data

The `ent_sub` request enters subscription data.

Request

The `ent_sub` request can be specified using various combinations of parameters to enter the following subscription data and relationships:

- Enter IMSI w/o any associated DNs
- Enter IMSI associated with 1 up to 8 DNs
- Enter Standalone DNs
- Enter MNP Range Subscriber with BDN and EDN for Split feature

Enter IMSI without any associated DNs:

This specification of the `ent_sub` request creates an IMSI record without any associated DNs.

Here DN is interchangeable to MSISDN.

```
ent_sub([iid XXXXX,] imsi XXXXX, pt XXXX, asd XXXXXX, cgbl yes|no, cdbl yes|no,
sp|rn XXXXX. grn XXXXX [, timeout <0..3600>])
```

Enter IMSI associated with 1 to 8 DNs:

This specification of the `ent_sub` request will attempt to create a subscription with an IMSI with up to 8 DN associations.

```
ent_sub([iid XXXXX,] imsi XXXXX, dn DN1, ..., dn DN8, pt XXXX, asd XXXXXX, cgbl
yes|no, cdbl yes|no, sp|rn XXXXX. grn XXXXX [, timeout <0..3600>])
```

Enter Standalone DNs:

This specification of the ent_sub request will attempt to create up to eight single DNs without associating any of them with an IMSI.

```
ent_sub([iid XXXXX,] dn DN1, ..., dn DN8, pt XXXX, asd XXXXXX, cgbl yes|no, cdbl
yes|no, sp|rn XXXXX, grn XXXXX [, timeout <0..3600>])
```

Enter MNP Range Subscriber with BDN and EDN for split feature:

This specification of the ent_sub request will attempt to create MNP range subscriber with bdn, edn and rsplit flag alongwith other fields.

```
ent_sub([iid XXXXX,] bdn XXXXX, edn XXXXX, pt XXXX, asd XXXXXX, cgbl yes|no, cdbl
yes|no, sp|rn XXXXX, grn XXXXX, rsplit yes|no [, timeout <0..3600>])
```

Note: rsplit field is only applicable for split feature.

Parameters:

imsi A single IMSI

Values: a string with 10 to 15 characters where each character must be a decimal digit from 0 to 9.

dn Max 8 DN[MSISDN]

Values: a string with 8 to 15 characters where each character must be a decimal digit from 0 to 9.

bdn Begin DN[MSISDN]

Values: a string with 8 to 15 characters where each character must be a decimal digit from 0 to 9.

edn End DN[MSISDN]

Values: a string with 8 to 15 characters where each character must be a decimal digit from 0 to 9.

pt Portability Type

Values:

asd Additional subscription details

Values:

cgbl Calling Black List

Values: yes or no

cdbl Called Black List

Values: yes or no

sp It is a type of SPRNID

Values: EE60

(1 to 21 characters). Allowed values are any ASCII printable character, values x20 to x7e.

The SP and RN fields can not have values at the same time.

rn It is a type of SPRNID

(1 to 21 characters). Allowed values are any ASCII printable character, values x20 to x7e.

grn It is referred as GRNID

Values: Any string that can be used to identify the GRNID for the subscriber (1 to 21 characters). Allowed values are any ASCII printable character, values x20 to x7e.

rsplit It is referred as Split flag that can be used to identify if range can be splited or not.

Values: yes or no

Response

The `ent_sub` response returns the results of entering subscription information.

Error Codes 6: Enter Subscription Data

Return Code	Description	Data Parameter
`SUCCESS`	Subscription information entered successfully.	None
`PROV_PROHIBITED`	Provisioning has been manually disabled.	`data (reason "Provisioning manually disabled")`
`WRITE_IN_READ_TXN`	The create request was sent on a read only transaction.	None
`IMSI_DN_LIMIT`	The addition of the DNs specified in the request would cause the IMSI to have more than eight DNs.	`data (reason "More than 8 DNs")`
`TXN_TOO_BIG`	This request would cause the current transaction to be larger than the limit.	None
`NO_ACTIVE_TXN`	There was no currently active transaction for this connection.	None
`NO_WRITE_PERMISSION`	The PDBI client making the request does not have write access permissions.	None
`WRITE_UNAVAIL`	Another client already has a write transaction open.	The IP address information of the client that already has the write transaction. `data (id <0...4294967295>, ip <ip addr>, port <0...65635>)`

5.1.11 Update Subscription Data

The `upd_sub` request modifies existing subscription data.

Request

The `upd_sub` request can be specified using various combinations of parameters to update the following existing subscription data and relationships:

- Modify single field for a specific key IMSI or DN[MSISDN] or BDN and EDN
- Modify multiple fields for a specific key IMSI or DN[MSISDN] or BDN and EDN

Modify single field for a specific key IMSI or DN or BDN and EDN:

This specification of the `upd_sub` request modifies the single field for a specific IMSI or DN.

```
upd_sub([iid XXXXX,] imsi XXXXX, sp|rn XXXXX [, timeout <0..3600>])
```

```
upd_sub([iid XXXXX,] dn XXXXX, sp|rn XXXXX [, timeout <0..3600>])
```

```
upd_sub([iid XXXXX,] bdn XXXXX, edn XXXXX, pt XXXXX [, timeout <0..3600>])
```

Modify multiple fields for a specific key IMSI or DN or BDN and EDN:

This specification of the `upd_sub` request modifies multiple single field for a specific IMSI or DN.

```
upd_sub([iid XXXXX,] imsi XXXXX, pt XXXXX, sp|rn XXXXX, grn XXXXX [, timeout
<0..3600>])
```

```
upd_sub([iid XXXXX,] dn XXXXX, asd XXXXX, cgbl yes|no, grn XXXXX [, timeout
<0..3600>])
```

```
upd_sub([iid XXXXX,] bdn XXXXX, edn XXXXX, asd XXXXX, cgbl yes|no, grn XXXXX,
rsplit yes|no [, timeout <0..3600>])
```

Note: BDN and EDN for Split feature will have extra field rsplit.

Parameters:

imsi A single IMSI

Values: a string with 10 to 15 characters where each character must be a decimal digit from 0 to 9.

dn Max 8 DN[MSISDN]

Values: a string with 8 to 15 characters where each character must be a decimal digit from 0 to 9.

bdn Begin DN[MSISDN]

Values: a string with 8 to 15 characters where each character must be a decimal digit from 0 to 9.

edn End DN[MSISDN]

Values: a string with 8 to 15 characters where each character must be a decimal digit from 0 to 9.

pt Portability Type

Values:

asd Additional subscription details

Values:

cgbl Calling Black List

Values: yes or no

cdbl Called Black List

Values: yes or no

sp It is a type of SPRNID

Values: EE60

(1 to 21 characters). Allowed values are any ASCII printable character, values x20 to x7e.

The SP and RN fields can not have values at the same time.

rn It is a type of SPRNID

(1 to 21 characters). Allowed values are any ASCII printable character, values x20 to x7e.

grn It is referred as GRNID

Values: Any string that can be used to identify the GRNID for the subscriber (1 to 21 characters).

Allowed values are any ASCII printable character, values x20 to x7e.

rsplit It is referred as Split flag that can be used to identify if range can be splited or not.

Values: yes or no

Response

The `upd_sub` response returns the results of updating subscription data.

Error Codes 7: Update Subscription Data

Return Code	Description	Data Parameter
`SUCCESS`	Subscription information entered successfully.	None
`PROV_PROHIBITED`	Provisioning has been manually disabled.	`data (reason "Provisioning manually disabled")`
`WRITE_IN_READ_TXN`	The create request was sent on a read only transaction.	None
`NOT_FOUND`	The requested IMSI or DN was not found.	None

IMSI_DN_LIMIT	The addition of the DNs specified in the request would cause the IMSI to have more than eight DNs.	data (reason "More than 8 DNs")
TXN_TOO_BIG	This request would cause the current transaction to be larger than the limit.	None
NO_ACTIVE_TXN	There was no currently active transaction for this connection.	None
NO_WRITE_PERMISSION	The PDBI client making the request does not have write access permissions.	None
WRITE_UNAVAIL	Another client already has a write transaction open.	The IP address information of the client that already has the write transaction. data (id <0...4294967295>, ip <ip addr>, port <0...65635>)
NO_UPDATES	The write transaction had no successful updates. No database change will occur and no new database level will be given.	None

5.1.12 Retrieve Subscription Data

The rtrv_sub request retrieves information about existing subscription data.

Request

The rtrv_sub request can be specified using various combinations of parameters to accomplish the following:

- Retrieve Subscription Data for a Specific IMSI
- Retrieve Subscription Data for Specific DN
- Retrieve Subscription Data for bdn and edn for split feature

Retrieve Subscription Data for a Specific IMSI

This specification of rtrv_sub request retrieves subscription data for a specific IMSI

```
rtrv_sub([iid XXXXX,] imsi XXXXX)
```

Retrieve Subscription Data for a Specific DN

This specification of rtrv_sub request retrieves the subscription information for a specific DN

```
rtrv_sub([iid XXXXX,] dn XXXXX)
```

Retrieve Subscription Data for BDN and EDN for split feature

This specification of `rtrv_sub` request retrieves the subscription information for BDN and EDN

```
rtrv_sub([iid XXXXX,] bdn XXXXX, edn XXXXX)
```

Parameters:

imsi The specific IMSI to retrieve.

Value: a string with 10 to 15 characters where each character must be a decimal digit from 0 to 9

dn The specific DN to retrieve (specified in international format).

Values: a string with 8 to 15 characters where each character must be a decimal digit from 0 to 9.

bdn The specific BDN to retrieve (specified in international format).

Values: a string with 8 to 15 characters where each character must be a decimal digit from 0 to 9.

edn The specific EDN to retrieve (specified in international format).

Values: a string with 8 to 15 characters where each character must be a decimal digit from 0 to 9.

Response

The `rtrv_sub` response returns the results of the retrieve subscription data operation.

The syntax (shown in subsequent sections) of the response data section for a successful `rtrv_sub` request depends on the type of data requested for retrieval.

Error Codes 8: Retrieve Subscription Data

Return Code	Description	Data Parameter
SUCCESS	The request succeeded and this is the last (or only) response	See subsequent sections for the contents of the response data section.
NOT_FOUND	The requested DN or IMSI was not found.	None
NO_ACTIVE_TXN	There was no currently active transaction for this connection.	None
INTERRUPTED	Request was interrupted due to loss of connection or process restart.	data (reason "received XXXX signal")

IMSI Query Response Data

`rtrv_sub` requests with the `imsi` parameter specified will generate response containing the specified subscription data.

```
data (imsis (imsi (id XXXX [, dns ( dn1 [, dn2] ... [, dn8])], pt XXXXX, asd
XXXXX, cgbl yes|no, cdbl yes|no, sp|rn XXXXX, grn XXXXX)))
```

DN Query Response Data

`rtrv_sub` requests with the `dn` parameters specified will generate one or more responses containing the specified subscription data.

```
data (dns (dn (id XXXXX [, imsi XXXXX] , pt XXXXX, asd XXXXX, cgbl yes|no, cdbl
yes|no, sp|rn XXXXX, grn XXXXX)))
```

BDN and EDN Response Data

`rtrv_sub` requests with the `bdn` and `edn` parameters specified will generate one or more responses containing the specified subscription data.

```
data (dnblock (dnblock (bdn XXXXX ,edn XXXXX, pt XXXXX, asd XXXXX, cgbl yes|no,
cdbl yes|no, sp|rn XXXXX, grn XXXXX, rsplit yes|no)))
```

5.1.13 Delete Subscription Data

The `dlt_sub` request is used for deleting subscription data for a specific key like DN or IMSI, or bdn and edn for split

Request

The `dlt_sub` request can be specified using key[IMSI or DN or bdn and edn together] parameter.

For specific key [IMSI or DN]

```
dlt_sub([iid XXXXX,] imsi|dn XXXXX [, timeout <0..3600>])
```

For bdn and edn keys with split feature

```
dlt_sub([iid XXXXX,] bdn XXXXX, edn XXXXX [, timeout <0..3600>])
```

Parameters:

imsi A single IMSI

Values: a string with 10 to 15 characters where each character must be a decimal digit from 0 to 9.

dn Max 8 DN[MSISDN]

Values: a string with 8 to 15 characters where each character must be a decimal digit from 0 to 9.

bdn Max 8 BDN[MSISDN]

Values: a string with 8 to 15 characters where each character must be a decimal digit from 0 to 9.

edn Max 8 EDN[MSISDN]

Values: a string with 8 to 15 characters where each character must be a decimal digit from 0 to 9.

timeout (Optional) The amount of time (in seconds) to wait to open a write transaction if another connection already has one open. Write transaction waiting to be open, will be opened in the

order that the requests were received. This option is only allowed if the client used the txnmodesingle option on the connect() request. Values: 0 (return immediately if not available) to 3600 seconds (default is 0)

Response

The `dlt_sub` response returns the results of deleting subscription data.

Error Codes 9: Delete Subscription Data

Return Code	Description	Data Parameter
`SUCCESS`	Subscription information entered successfully.	None
`PROV_PROHIBITED`	Provisioning has been manually disabled.	`data (reason "Provisioning manually disabled")`
`WRITE_IN_READ_TXN`	The create request was sent on a read only transaction.	None
`NOT_FOUND`	The requested DN or IMSI was not found.	None
`TXN_TOO_BIG`	This request would cause the current transaction to be larger than the limit.	None
`NO_ACTIVE_TXN`	There was no currently active transaction for this connection.	None
`NO_WRITE_PERMISSION`	The PDBI client making the request does not have write access permissions.	None
`WRITE_UNAVAIL`	Another client already has a write transaction open.	The IP address information of the client that already has the write transaction. `data (id <0...4294967295>, ip <ip addr>, port <0...65635>)`

5.1.14 Unsupported operations of PDBI in UDR

There are some operations of PDBI, which are not supported in UDR.

SR NO.	Operation of PDBI	Description
1	`ent_entity`	To create a network entity like MnpSPRN or MnpGRN entity for MNP
2	`rtrv_entity`	To retrieve entity information for MnpSPRN or MnpGRN entities
3	`status`	To query the status information from PDBA

4	rtrv_dbrpt	To retrieves the Database Report of servers meeting the specified criteria
5	rtrv_dblist	To retrieve the database status for all servers matching the specified criteria
6	export	To export the provisioning data

Chapter 6. MNP Subscriber Provisioning

NOTE: For command responses, the error code values described are listed in Appendix B.

6.1 MNP Subscriber Profile Commands

Table 8 Summary of Subscriber Profile Commands

Command	Description	Keys	Command Syntax
Create Profile	Create a subscriber or subscriber profile	DN[MSISDN], IMSI	`ent_sub([iid XXXXX,] imsi XXXXX, pt XXXX, asd XXXXXX, cgbl yes|no, cdbl yes|no, sp|rn XXXXX. grn XXXXX [, timeout <0..3600>])`
Get Profile	Get subscriber profile data		`rtrv_sub([iid XXXXX,] imsi XXXXX, sp XXXXX [, timeout <0..3600>])`
Update Profile	Update a subscriber or subscriber profile		`upd_sub([iid XXXXX,] imsi XXXXX, sp XXXXX [, timeout <0..3600>])`
Delete Profile	Delete all subscriber profile data and all opaque data associated with the subscriber		`dlt_sub([iid XXXXX,] imsi|dn XXXXX [, timeout <0..3600>])`
Create Profile with Split feature	Create a subscriber or subscriber profile with split feature	BDN and EDN [MSISDN]	`ent_sub([iid XXXXX,] bdn XXXXX,edn XXXXX, pt XXXX, asd XXXXXX, cgbl yes|no, cdbl yes|no, sp XXXXX. grn XXXXX, rsplit yes|no [, timeout <0..3600>])` Note: Split feature should be enabled.

6.1.1 Create Profile

Description

This operation creates a MNP subscriber profile using the field-value pairs that are specified in the request content.

NOTES

- The subscriber profile data provided is fully validated against the definition in the SEC. If the validation check fails, then the request is rejected.

Prerequisites

A subscriber with any of the keys supplied in the profile must not exist.

A MNP subscriber with any of the entity(s);MnpGRN or and MnpSPRN supplied in the profile must be provisioned first.

For MNP range subscriber with split, split feature should be enabled first.

Create Profile Examples

Request 1

A subscriber is created, with a MSISDN, and IMSI keys along with other fields.

```
# ent_sub(iid 1, imsi 184569547984229, dn 33123654862, pt 1, asd 1, cgbl yes, cdbl
yes, sp 1111177777, grn 10000)
```

Response 1

The request is successful, and the subscriber was created.

```
# rsp(iid 1, rc 0, data (Success))
```

Request 2

A subscriber is created, with an IMSI key. Another subscriber exists with same key of IMSI.

```
# ent_sub(iid 1, imsi 184569547984229,  pt 1, asd 1, cgbl yes, cdbl yes, sp
1111177777, grn 10000)
```

Response 2

The request fails. The error value indicates a subscriber exists with the given IMSI, and the affected rows are 0.

```
# rsp(iid 1, rc 1015)
```

Request 3

A subscriber is created, with a bdn, and edn keys along with other fields.

```
# ent_sub(iid 1, bdn 111111111, edn 111111119, pt 1, asd 1, cgbl yes, cdbl yes, sp
1111177777, grn 10000, rsplit yes)
```

Response 3

The request is successful, and the subscriber was created.

```
# rsp(iid 1, rc 0, data (Success))
```

Request 4

A subscriber is created, with a dn key and SPRNID (1111177777) is present with type **rn** .

```
# ent_sub (iid 2, dn 56920632097, pt 1, sp 1111177777, timeout 10)
```

Response 4

The request is unsuccessful, and the subscriber was not created as correct RNID type is not present.

```
# rsp(iid 3, rc 1012)
```

Request 5

A subscriber is created, with a dn key and SPRNID (1111177777) is present with type **rn** .

```
# ent_sub (iid 2, dn 56920632097, pt 1, rn 1111177777, timeout 10)
```

Response 5

The request is successful, and the subscriber was created.

```
# rsp(iid 2, rc 0, data (Success))
```

Request 6

A subscriber is created, with a bdn, and edn keys along with other fields also SPRNID (1111177777) is present with type **rn.**

```
# ent_sub(iid 3, bdn 111111111, edn 111111119, pt 1, asd 1, cgbl yes, cdbl yes, sp
1111177777, grn 10000)
```

Response 6

The request is successful, and the subscriber was created.

```
# rsp(iid 3, rc 1012)
```

Request 7

A subscriber is created, with a multiple dn, SPRNID (1111177777) is present with type **rn.**

```
# ent_sub (iid 2, dn 56920633098, dn 56920633099, dn 56920633100, pt 1, rn
1111177777, timeout 10)
```

Response 7

The request is successful, and the subscriber was created.

```
# rsp(iid 2, rc 0, data (Success))
```

6.1.2 Get Profile

Description

This operation retrieves all field-value pairs created for a subscriber that is identified by the Keys specified in keyNameX and keyValueX.

The keyNameX and keyValueX values are required in the request in order to identify the subscriber. The response content includes only valid field-value pairs, which have been previously provisioned or created by default.

Prerequisites

A subscriber with the Keys of the keyNameX/keyValueX values supplied must exist.

Get Profile Examples

Request 1

A request is made to get profile data for a subscriber An IMSI key is supplied.

```
# rtrv_sub(iid 1, imsi 184569547984229)
```

Response 1

The request is successful, and the subscriber profile data is returned.

```
# rsp(iid 1, rc 0, data (imsis (imsi (id 184569547984229, dns (33123654862,
33123654863, 33123654864), pt 1, asd 1, cgbl yes, cdbl yes, sp 1111177777, grn
10000))))
```

Request 2

A request is made to get profile data for a subscriber. A MSISDN key is supplied but it is not present.

```
# rtrv_sub(iid 1, dn 33123654865)
```

Response 2

The request is failed, and no subscriber profile data is returned.

```
# rsp(iid 1, rc 1013)
```

Request 3

A request is made to get profile data for a MNP range subscriber with bdn and edn

```
# rtrv_sub(iid 1, bdn 11111111, edn 1111111119)
```

Response 3

The request is successful, and the subscriber profile data is returned.

```
# rsp(iid 3, rc 0, data (dnblocks (dnblock ( bdn 1111111111, edn 1111111119, pt 4,
asd asd, cgbl yes, cdbl no, sp 1111177777, grn 10000, rsplit yes))))
```

Request 4

A request is made to get profile data for a subscriber. A MSISDN key is supplied for SPRNID of type **rn**.

```
# rtrv_sub(iid 4, dn 56920632097)
```

Response 4

The request is successful, and the subscriber profile data is returned.

```
# rsp(iid 4, rc 0, data (dns (dn (id 56920632097, pt 1, rn 1111177777))))
```

Request 5

A request is made to get profile data for a subscriber. A MSISDN key is supplied for subscriber with SPRNID of type **sp**.

```
# rtrv_sub(iid 4, dn 56920632097)
```

Response 5

The request is successful, and the subscriber profile data is returned.

```
# rsp(iid 5, rc 0, data (dns (dn (id 56920632098, pt 1, sp 6666677777))))
```

6.1.3 Update Profile

Description

This operation updates all profile data for the subscriber that is identified by the keys specified in keyNameX and keyValueX.

Prerequisites

A subscriber with the Keys of the keyNameX/keyValueX values supplied must exist.

For MNP range subscriber with split, split feature should be enabled first.

Update Profile Examples

Request 1

The subscriber with the given MSISDN is updated with single field. The subscriber exists.

```
# upd_sub(iid 1, dn 33123654862, sp 10000)
```

Response 1

The request is successful and field “sp” value is updated to 10000 for imsi 33123454862.

rsp(iid 1, rc 0, data (Success))

Request 2

The subscriber with the given MSISDN is updated with multiple fields. The subscriber exists.

```
# upd_sub(iid 1, imsi 33123654862, cgbl yes, cdbl no, sp 10000)
```

Response 2

The request is successful and field “sp” value is updated to 10000, cgbl to yes and cdbl to no for imsi 33123454862.

```
# rsp(iid 1, rc 0, data (Success))
```

Request 3

The subscriber with the given MSISDN is updated with single field. The subscriber does not exist.

```
# upd_sub(iid 1, imsi 33123654864, grn 20000)
```

Response 3

The request fails.

```
# rsp(iid 1, rc 1013)
```

Request 4

The MNP range subscriber with the given BDN and EDN is updated with single field. The subscriber exists.

```
# upd_sub(iid 1, bdn 11111111, edn 1111111119, pt 2)
```

Response 4

The request is successful and field “pt” value is updated to 2 for MNP range subscriber 1111111111-11111119.

rsp(iid 1, rc 0, data (Success))

Request 5

The subscriber with the given MSISDN is updated with single field. The subscriber exists.

```
# upd_sub(iid 5, dn 56920632098, rn 6666677777)
```

Response 5

The request is unsuccessful as the type for the SPRNID 6666677777 is sp .

rsp(iid 5, rc 1012)

Request 6

The subscriber with the given MSISDN is updated with single field. The subscriber exists.

```
# upd_sub(iid 6, dn 56920632098, sp 6666677777)
```

Response 6

The request is successful and the field is updated.

rsp(iid 6, rc 0, data (Success))

6.1.4 Delete Profile

Description

This operation deletes all profile data for the subscriber that is identified by the Keys specified in keyNameX and keyValueX.

Prerequisites

A subscriber with the Keys of the keyNameX/keyValueX values supplied must exist.

For MNP range subscriber with split, split feature should be enabled first.

Delete Profile Examples

Request 1

The subscriber with the given MSISDN is deleted. The subscriber exists.

```
# dlt_sub(iid 1, dn 33123654862)
```

Response 1

The request is success, and Success is returned in response data.

```
# rsp(iid 1, rc 0, data (Success))
```

Request 2

The subscriber with the given MSISDN is deleted. The subscriber does not exist.

```
# dlt_sub(iid 1, dn 33123654865)
```

Response 2

The request fails. The error value indicates a subscriber with the given MSISDN does not exist, and the affected rows are 0.

```
# rsp(iid 1, rc 1013)
```

Request 3

The MNP range subscriber with the given bdn and edn is deleted. The MNP range subscriber exists.

```
# dlt_sub(iid 1, bdn 1111111111, edn 1111111119)
```

Response 3

The request is success, and Success is returned in response data.

```
# rsp(iid 1, rc 0, data (Success))
```

Chapter 7. Example Sessions with normal and single transaction

7.1 Normal Transaction Session

This example shows a `normal transaction` connection with the create and update profile requests.

Table 9 Example-1 showing normal transaction connection

Message		Description
PDBI Client->UDR	`connect(iid 1, version 1.0)`	A PDBI connection has been established.
UDR->PDBI Client	`rsp(iid 1, rc 0, data (connectId 1, side active))`	
PDBI Client -> UDR	`begin_txn(iid 2, type write)`	A write transaction has been opened.
UDR-> PDBI Client	`rsp(iid 2, rc 0)`	
PDBI Client-> UDR	`ent_sub(iid 3, imsi 958209032900, dn 1122334455, pt 1, asd 1, cgbl yes, cdbl yes, sp 20000, grn 10000)`	A create request for subscriber is initiated.
PDBI Client-> UDR	`upd_sub(iid 4, dn 33123654862, sp 10000)`	A update request for subscriber is initiated.
PDBI Client-> UDR	`end_txn(iid 5)`	The write transaction has been ended. The requests have been executed as a SOAP transaction.
UDR->PDBI Client	`rsp(iid 5, rc 0, data(Success))`	
PDBI Client->UDR	`disconnect(iid 6)`	The client is done and has disconnected.
UDR-> PDBI Client	`rsp(iid 6, rc 0)`	

7.2 Single Transaction Session

This example shows a connection using the `txnmode single` connect option with the creation of a few different kinds of subscriptions.

Table 10 Example-2 showing connection using txnmode single connect option

Message		Description
PDBI Client->UDR	`connect(iid 1, version 1.0, txnmode single)`	A PDBI connection has been established.
UDR->PDBI Client	`rsp(iid 1, rc 0, data (connectId 1, side active))`	
PDBI Client-> UDR	`ent_sub(iid 2, imsi 958209032900, dn 1122334455, pt 1, asd 1, cgbl yes, cdbl yes, sp 20000, grn 10000)`	Subscriber is created.
UDR->PDBI Client	`rsp(iid 2, rc 0, data (Success))`	
PDBI Client-> UDR	`upd_sub(iid 3, dn 33123654862, sp 10000)`	Subscriber is updated.
UDR->PDBI Client	`rsp(iid 3, rc 0, data (Success))`	
PDBI Client->UDR	`disconnect(iid 4)`	The client is done and has disconnected.
UDR-> PDBI Client	`rsp(iid 4, rc 0)`	

Appendix A – PDBI Response Message Return Codes

PDBI response message return codes are returned by PDBA in the rc parameter of the PDBI rsp message (see section 5.1.4). The rc parameter of an PDBI response message indicates the success or failure of an PDBI request. The complete set of PDBI response message return codes and their associated values are defined in the following table.

Table 11 PDBI response message return codes and their associated values

Return Code	Value	Description
SUCCESS	0000	No Error
INTERNAL_ERROR	1001	An unexpected error was encountered and as a result, the request was unable to be completed. Tekelec Customer Care should be called to investigate.
NOT_CONNECTED	1002	The client has established a TCP/IP socket connection with the PDBI server, but has not yet issued a connect() request.
ALREADY_CONNECTED	1003	The client has already sent a connect() request over the established TCP/IP socket connection with the PDBI server.
PARSE_FAILED	1004	Failed parsing incoming request message.
WRITE_UNAVAIL	1005	Another client has already opened a transaction for write access.
NO_WRITE_PERMISSION	1006	The PDBI client does not have write access permissions.
Reserved	1007	Reserved
STANDBY_SIDE	1008	A write transaction was attempted on the standby server.
NO_ACTIVE_TXN	1009	A read or write transaction is not currently open for this connection.
ACTIVE_TXN	1010	A read or write transaction is already open on this connection or an open transaction was aborted prior to terminating the connection.
WRITE_IN_READ_TXN	1011	A write request was sent in a read only transaction.
INVALID_VALUE	1012	One of the fields in the request has an invalid value.
NOT_FOUND	1013	Item cannot be found in the database.
CONFLICT_FOUND	1014	The item already exists in the database or the E.164 address specified in the request is reserved for an HLR.
ITEMS_EXISTS	1015	The item already exists.
PARTIAL_SUCCESS	1016	The request has succeeded, but this is one of several responses.
NO_UPDATES	1017	The write transaction did not have any successful updates.
INTERRUPTED	1018	The operation was interrupted due to cancelation or failure.

BAD_ARGS	1019	The request has semantic problems.
CONNECTION_DENIED	1020	Client connection is denied due to the reason (e.g. Interface disabled, Too many connections) specified in the response data.
NE_NOT_FOUND	1021	The specified SP does not exist.
CONTAINS_SUBS	1022	The Network Entity to be deleted still contains subscription data.
UNKNOWN_VERSION	1023	The specified version of PDBI API is not supported.
UNIMPLEMENTED	1025	Incoming message is valid, but not implemented.
IMSI_DN_LIMIT	1027	Maximum number of DNs for the specified IMSI already exists.
TXN_TOO_BIG	1029	Maximum size of transaction has exceeded the allowed limit.
DB_EXCEPTION	1031	An unexpected database exception occurred. Tekelec Customer Care should be called to investigate.
PERMISSION_DENIED	1046	Permission denied (e.g. The PDBI command cannot be issued by a remote client)
PROV_PROHIBITED	1051	System provisioning has been manually disabled through the Application GUI. The associated request should be resent after provisioning has been manually re-enabled.

Appendix B – UDR GUI configuration for PDBI

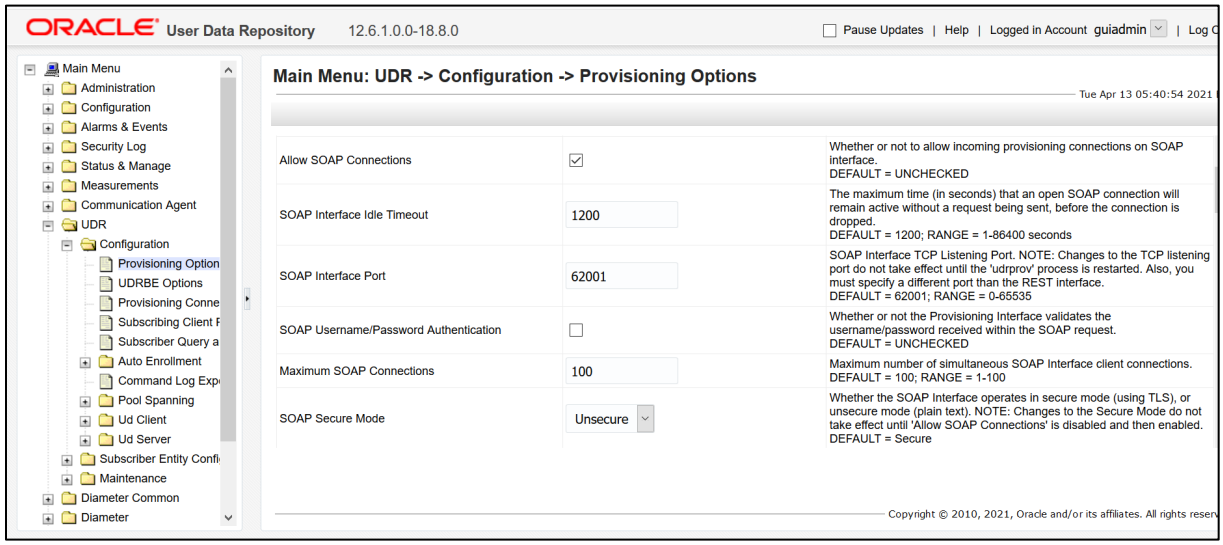
Login to UDR active NOAMP GUI and follow the below steps to configure the parameters for PDBI.

Step-1 Configure the ProvOptions screen under *Main Menu: UDR -> Configuration -> Provisioning Options*

"Allow SOAP Connections" field should be enabled.

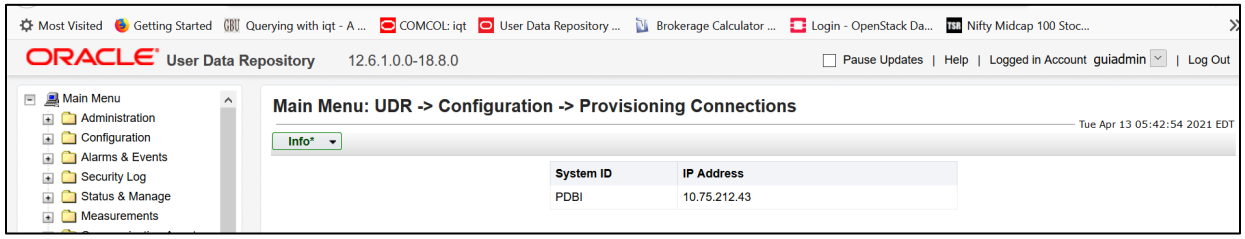
"SOAP Secure Mode" field should be "unsecure".

"Allow PDBI Connections" field should be enabled.



Step-2 Configure the whitelist in ProvConnections screen under *Main Menu: UDR -> Configuration -> Provisioning Connections.*

Here, whitelist means an IP from where PDBI request will be entertained by UDR.



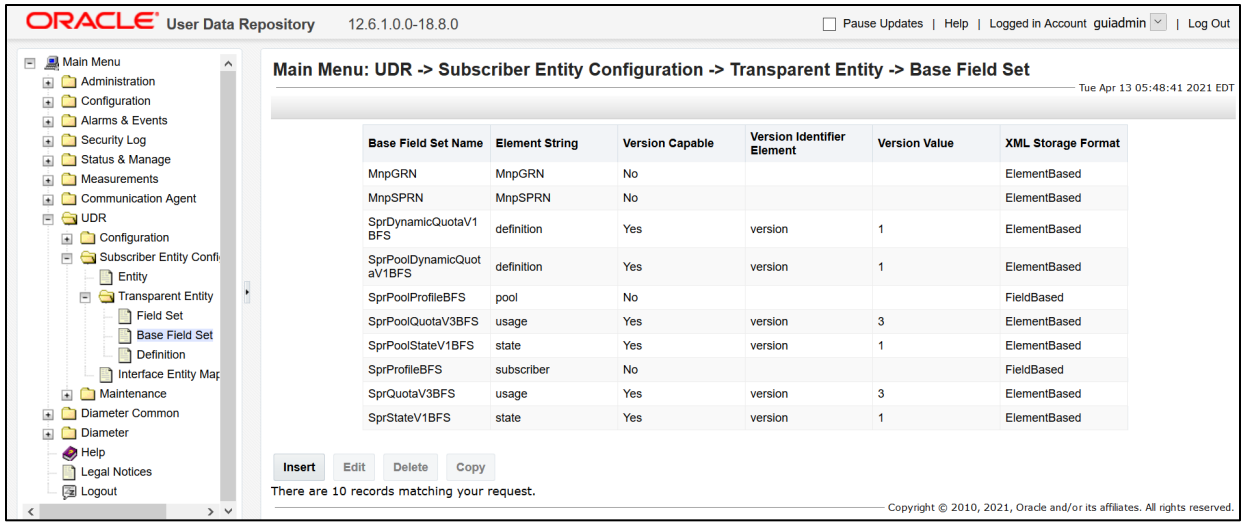
Step-3 Configure the MNP based SEC, execute the loader at active NOAMP server.

Path of MNP loader at NOAMP - /usr/TKLC/udr/prod/maint/loaders/upgrade/enablevMNPSec

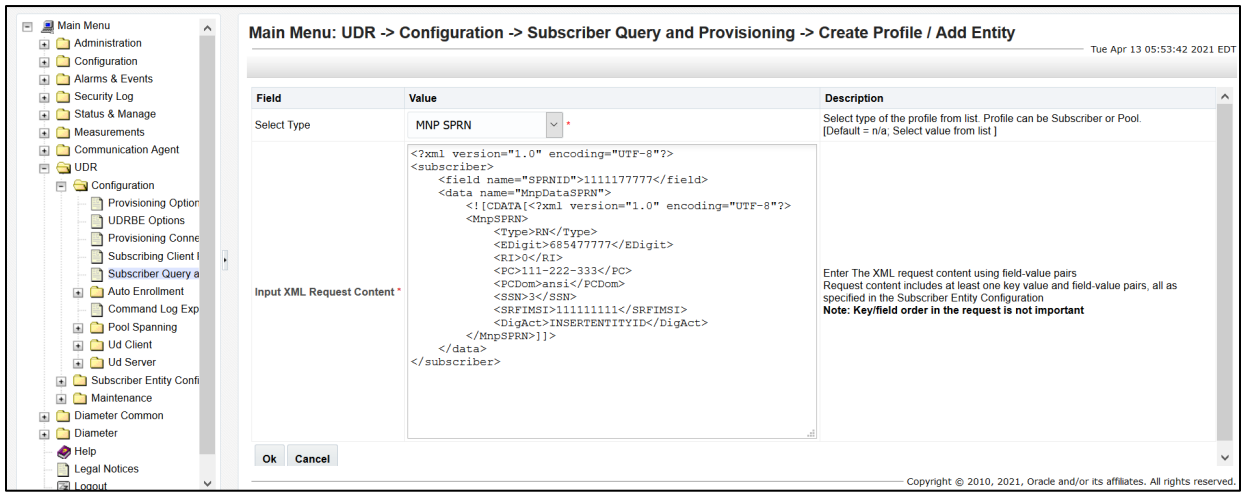
[root@PDBI-UDR-NOAMP admusr]# /usr/TKLC/udr/prod/maint/loaders/upgrade/enablevMNPSec

[root@PDBI-UDR-NOAMP admusr]#

Verify that "Subscriber Entity Configuration screen" should have MnpGRN and MnpSPRN basefield configured as per below snapshot:



Step-4 Please create GRN and SPRN entity from active NOAMP GUI as we do not support entity creation request from PDBI, at *Main Menu: UDR->Configuration-> Subscriber Query and Provisioning-> Create Profile/Add Enmity*



Note: This step-5 is only applicable when PDBI request needs to be executed for Split feature

Step-5 Please enable below loaders on active NOAMP

Path of MNP loader at NOAMP - /usr/TKLC/udr/prod/maint/loaders/upgrade/enableMNPwithSplit

- /usr/TKLC/udr/prod/maint/loaders/upgrade/enableSplitFeature

[root@PDBI-UDR-NOAMP admusr]# /usr/TKLC/udr/prod/maint/loaders/upgrade/enableMNPwithSplit

[root@PDBI-UDR-NOAMP admusr]# /usr/TKLC/udr/prod/maint/loaders/upgrade/enableSplitFeature

Appendix C – My Oracle Support

My Oracle Support (*https://support.oracle.com*) is your initial point of contact for all product support and training needs. A representative at Customer Access Support (CAS) can assist you with My Oracle Support registration.

Call the CAS main number at 1-800-223-1711 (toll-free in the US), or call the Oracle Support hotline for your local country from the list at *http://www.oracle.com/us/support/contact/index.html*. When calling, select the options in sequence on the Support telephone menu:

1. Select 2 for New Service Request
2. Select **3** for Hardware, Networking and Solaris Operating System Support
3. Select one of the following options:
    - For Technical issues such as creating a Service Request (SR), Select **1**
    - For Non-technical issues such as registration or assistance with My Oracle Support, Select **2**

You are connected to a live agent who can assist you with My Oracle Support registration and opening a support ticket.

My Oracle Support is available 24 hours a day, 7 days a week, and 365 days a year.

Appendix D – Locate Product Documentation on the Oracle Help Center Site

Oracle Communications customer documentation is available on the web at the Oracle Help Center (OHC) site, *http://docs.oracle.com*. You do not have to register to access these documents. Viewing these files requires Adobe Acrobat Reader, which can be downloaded at *http://www.adobe.com*

1. Access the Oracle Help Center site at *http://docs.oracle.com*
2. Click **Industries**.
3. Under the Oracle Communications subheading, click the **Oracle Communications documentation** link.
4. The Communications Documentation page displays. Most products covered by these documentation sets are under the headings Network Session Delivery and Control Infrastructure or Platforms.
5. Click your Product and then the Release Number.
6. A list of the entire documentation set for the selected product and release displays.
7. To download a file to your location, right-click the **PDF** link, select **Save target as** (or similar command based on your browser), and save to a local folder.

Appendix E – Feature Flag “enableSplitFeature” behavior:

1. Feature flag is enabled:
    - Split field not sent: By default Split field with value ‘1’ will be added to subscriber profile.
    - Split field value is ‘0’: Subscriber profile with Split field value ‘0’(non splitable range) will be inserted.
    - Split field value is ‘1’: Subscriber profile with Split field value ‘1’ will be inserted.

2. Feature flag is disabled:
    - Split field not sent : Subscriber profile will not have split field.
    - Split Field with value 1: Create/Update requests are rejected with invalid XML.
    - Split Field with value 0: Create/Update requests are rejected with invalid XML.