PostGreSQL Routing Database

Document STPD99-0006

Version 1.03

Change History

Date	Version	Description	Change Note
11/06/07	1.0	Initial Release	AT
22/06/07	1.01	Add Route Corrected	AT
05/07/07	1.02	Updated to reflect the following changes to the database: 1. Added new table for announcement and reject cause specification for rejected calls 2. get_routing_function_v4() Appendix listing changed	AT
07/01/2008	1.03	Updated descriptions of 'ic_rules' and 'og_rules' tables	AT

Content

Table of Figures

1 INTRODUCTION

The SQL routing database is provided as an example to how a database should be used in conjunction with SVI SQL API (for more information please refer to the SVI SQL API Definition document – STPD83-0002SVISQLDatabaseAPI DefinitionV2.3).

1.1 DATABASE OVERVIEW

The database uses PostGreSQL v8.2.1 to store routing data and procedures to interface with the SVI SQL API.

The SVI uses calls into the SQL procedure stored in this database to make routing decisions. The database can make routing decisions based on Called Party Number and Incoming hunt group of a call. It can also be configured to perform a whitelist/blacklist lookup certain calls before looking up the routing table.

1.2 DATABASE BACKUP

For information on how to backup and restore the PostGreSQL database please refer to Chapter 23 (section 1) of the PostGreSQL 8.2.x User Guide:

http://www.postgresql.org/docs/8.2/static/index.html

2 DATABASE DESIGN

The database consists of 4 tables and functions written in Procedural SQL. The tables are discussed further in the next sub-section. The Functions have been listed in Appendix A for reference use.

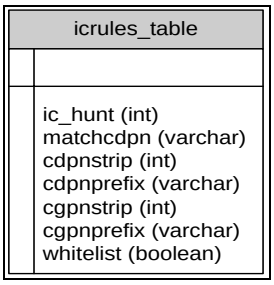
2.1 TABLES

The database consists of the 5 tables, which have been discussed in more detail below.

2.1.1 ICRULES_TABLE

All incoming routing requests should match with one of the entries in this table (matching based on 'ic_hunt' and 'matchcdpn') otherwise the database will reject the routing request.

Also, cdpn/cgpn stripping or prefixing specified in the icrules_table will be done before the routing and (if applicable) whitelist or routing table is looked up. These fields should be populated with a null string('') if no prefixing is required.

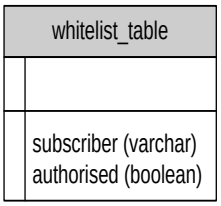


Field	Explanation
ic_hunt	Integer value indicating calls from which incoming hunt group (provisioned on the SVI) should be matched up against this entry in the table
matchcdpn	String for matching CDPN on incoming routing requests. If more than one match is found then the longest match is used.
cdpnstrip	Integer value, specifying how many (if any) cdpn digits should be stripped off in outgoing leg of the call. If no digits are to be stripped off it should be populated with 0.
cdpnprefix	Character value specifying what prefix (if any) to add to the cdpn digits in outgoing leg of the call. If no prefixing is required this field should be populated with ''.
cgpnstrip	Integer value, specifying how many (if any) cgpn digits should be stripped off in outgoing leg of the call. If no digits are to be stripped off it should be populated with 0.
cgpnprefix	Character value specifying what prefix (if any) to add to the cgpn digits in outgoing leg of the call. If no prefixing is required this field should be populated with ''.

announcement	Array of Integers indicating which set of announcement are valid for the call. The announcements here must be provisioned on the SVI using its internal config files.
whitelist	Boolean value indicating if the whitelist table should be looked up.

2.1.2 WHITELIST_TABLE

This table should contain details of all subscribers and will be referred to in case the icrules_table entry for a particular routing request has 'whitelist' set to 'true'.



Field	Explanation
Subscriber	CGPN of the subscriber. This should be an EXACT match. If any stripping/prefix is required before looking up the whitelist table then this can be done in the 'ic_rules' table.
Authorised	Boolean value indicating whether a subscriber is authorised (whitelist) or unauthorised (Blacklist)

2.1.3 ROUTING_TABLE

The routing table will be looked up for deciding where to route the call. For an incoming call the outgoing route will be based on a match with ic_hunt and cdpn. In case of more than one match the Longest Match will be selected.

Field	Explanation
ic_hunt	Array of Integers corresponsding to all the incoming hunt groups (provisioned on the SVI) to which this routing applies.
cdpn	String of CDPN used to determine the outgoing route. If there is more than one match then the longest match will be used.
og_hunt	Array of integers specifying all the applicable routes to the 'ic_hunt' + 'cdpn' pair in order of priority.

2.1.4 OGRULES_TABLE

This table contains all outgoing rules that should be applied before the SVI sends the Setup out for the 2nd leg of the call. A Routing request is matched to this table by comparing 'og_hunt' and 'matchcdpn' fields.

Field	Explanation
og_hunt	This should be populated with integer value of the outgoing hunt group.
matchcdpn	Character array.
cdpnstrip	Integer value, specifying how many (if any) cdpn digits should be stripped off in outgoing leg of the call. If no digits are to be stripped off it should be populated with 0.
cdpnprefix	Character value specifying what prefix (if any) to add to the cdpn digits in outgoing leg of the call. If no prefixing is required this field should be populated with ''.
cgpnstrip	Integer value, specifying how many (if any) cgpn digits should be stripped off in outgoing leg of the call. If no digits are to be stripped off it should be populated with 0.
cgpnprefix	Character value specifying what prefix (if any) to add to the cgpn digits in outgoing leg of the call. If no prefixing is required this field should be populated with ''.
cdpn_sc	Boolean value should be set to true to indicate CDPN Sending Complete
cdpn_noa	Integer value to set up CDPN Nature of Address in the outbound leg of the call.
cgpn_noa	Integer value to set up CGPN Nature of Address in the outbound leg of the call.
cgpn_pli	Integer value indicating CGPN Presentation Line Indicator of the outbound leg of the Call. Should be set to NULL to use default
cgpn_si	Integer value indicating CGPN Presentation Screening Indicator of the outbound leg of the Call. Should be set to NULL to use default
cgpn_br	Integer value indicating CGPN Bearer of the outbound leg of the Call. Should be set to NULL to use default
npdatabase	Integer Value specifying the Number Portability database for further lookups. This should be set to NULL if further lookups are not required.

mx_rttmpt	Integer Value that specifies maximum number of re-tries to be attempted before clearing the call back. Can be NULL
rttmpt_cds	Integer array of Clearing codes that on which the SVI should re-attempt calls. Can be NULL
og_protocol	Integer value specifying the protocol of the outgoing set-up. This field is mandatory and should always be populated.

2.1.5 ANN_AND_RJCAUSE_TABLE

This table is used for specifying an announcement or a Reject cause when the action returned by the SQL database is one of the two:

1. Unauthorised (actn Value 1)
2. No Routes Found (actn Value 5)

Field	Explanation
actn	Integer value: 1 for Unauthorised 5 for 'No Routes Found'
ann	Array of Integers specifying which announcement should be played out.
q850_cause	Integer Value specifying the clearing cause for all calls originating in SS7/H323
sip_cause	Integer Value specifying the clearing cause for incoming SIP calls

2.2 HOW THE DATABASE MAKES A ROUTING DECISION

Figure 1 shows how the SVI makes a decision each time it receives a routing request.

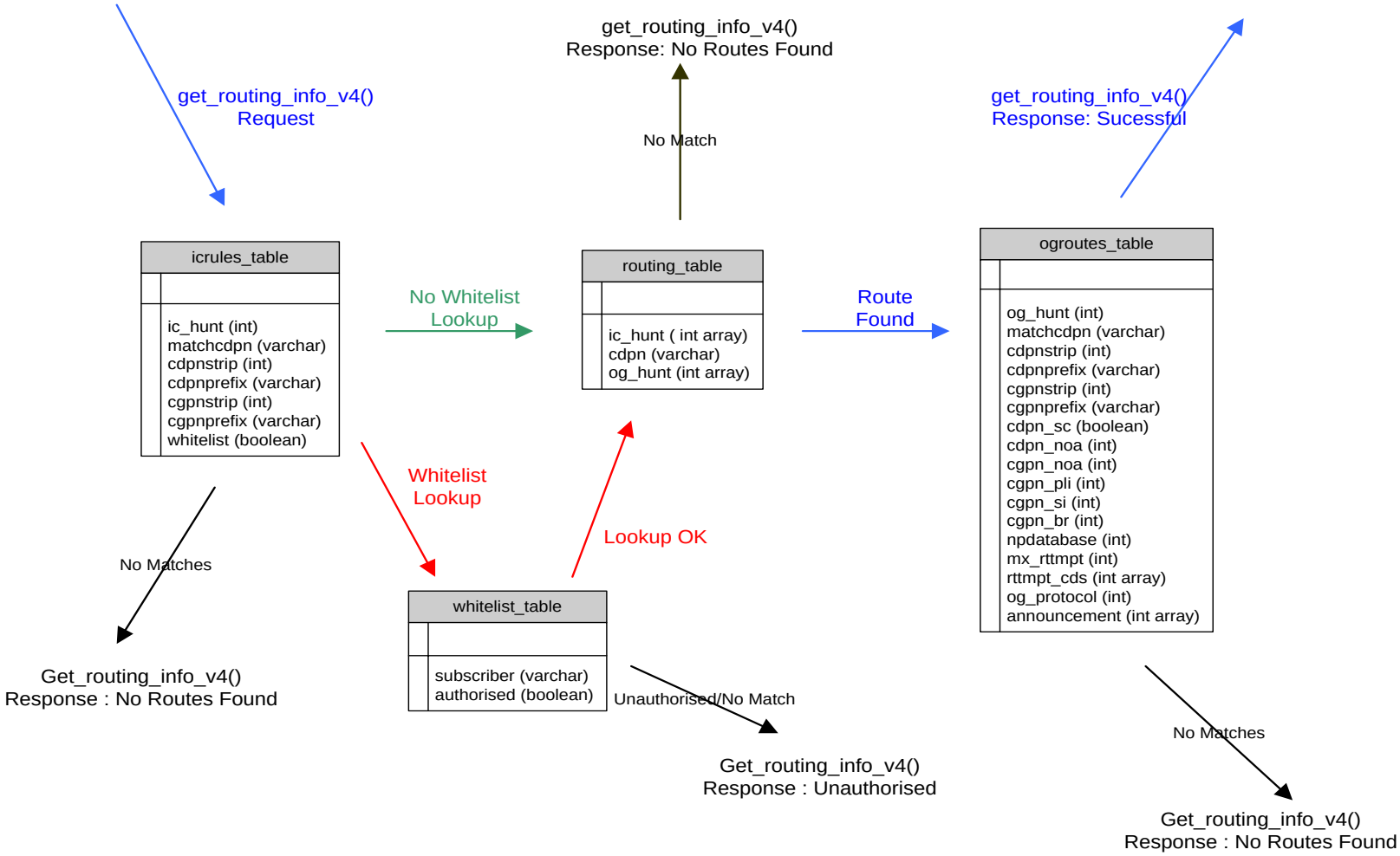


Figure 1: How Routing Decision is made

2.3 ADDING A ROUTE

To add a new route in the Database, information needs to be added to all the tables.

Icrules_table – This table allows user to match on the incoming hunt group and cdpn, also allowing for digit manipulation of both cdpn/cgpn before looking routing table as well whitelist table (if applicable).

The following statement will add a new entry into the icrules_table for all calls coming from hunt group 1 and cdpn starting from '951'.

INSERT INTO icrules_table VALUES (1, '951', 0, '', 0, '', '{0, 1}', false);

Please note that if digit manipulation is required at this stage elements 3, 4, 5 or 6 can be used to achieve that. If whitelist lookup is required then the last element should be set to true.

Routing_table – This table determines where to route the call.

The following statement sets the out going hunt group to 2

INSERT INTO routing_table VALUES ('{1}', '951', '{2}');

Please note the {} braces above for fields that are array.

Ogrules_table – This table specifies the outgoing rules that should be applied to the outbound leg of the setup. CDPN/CGPN stripping and prefixing can be done here, as well as setting CDPN NOA, CGPN NOA, CGPN SI (Screening Indicator), CGPN PLI (Presentation Line Indicator) and CGPN BR (Bearer). In the example below they have all been set to NULL which means that they will default to the values in the inbound leg of the call.

INSERT INTO ogrules_table VALUES (2, '951', 0, '', 0, '', true, NULL, NULL, NULL, NULL, NULL, NULL, 3, '{16}', 5);

For a 2nd SQL database lookup the nplookup field should be populated with the Instance of the 2nd SQL resource provisioned on the SVI.

2.4 Deleting a route

This can be done by removing the corresponding entry of the route from either of the three table mentioned above. For example, to remove the route added above any (or all) of the following statements can be used.

DELETE FROM icrules_table WHERE (ic_hunt = 1 AND matchcdpn = ‘951’);

DELETE FROM routing_table WHERE (ic_hunt = 1 AND matchcdpn = ‘951’);

DELETE FROM ogrules_table WHERE (og_hunt = 2 AND matchcdpn = ‘951’);

2.5 Setting up ann_and_rjcause_table

To configure the database to return announcements and a reject cause when response is unauthorised:

INSERT INTO ann_and_rjcause_table VALUES ('1', '{0}', '3', '401');

The above will set up the database to return announcement 0 and clearing cause 3 for SS7 and 401 for SIP in case the current routing request is ‘Unauthorised’.

2.6 Importing Data from CSV Files

To import data into any of the tables from a CSV file the following command can be used:

COPY <table name> FROM <filename> CSV

Prior to copying please make sure that the CSV file has same number of columns as the table in question.

2.7 Exporting Data to CSV File

To export data into a CSV file from any of the tables in the database the following command can be used:

COPY <table name> TO <filename> CSV

2.8 USE CASES

2.8.1 USE CASE FOR CALL REJECTED BY THE DATABASE

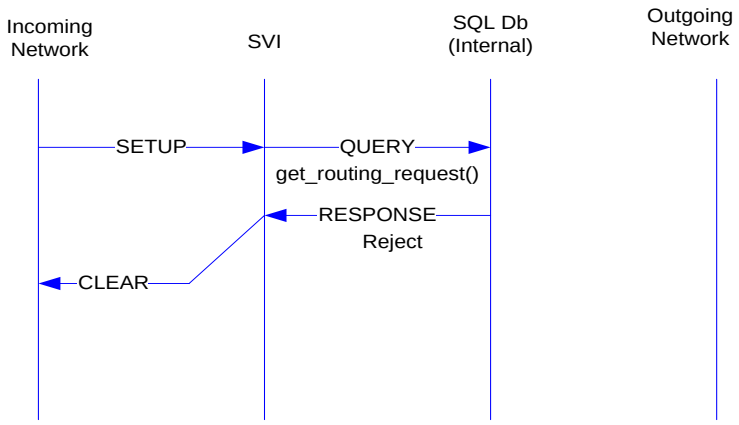


Figure 2: Use Case call rejected by the Database

2.8.2 USE CASE FOR CALL THAT GETS ROUTED ONWARDS

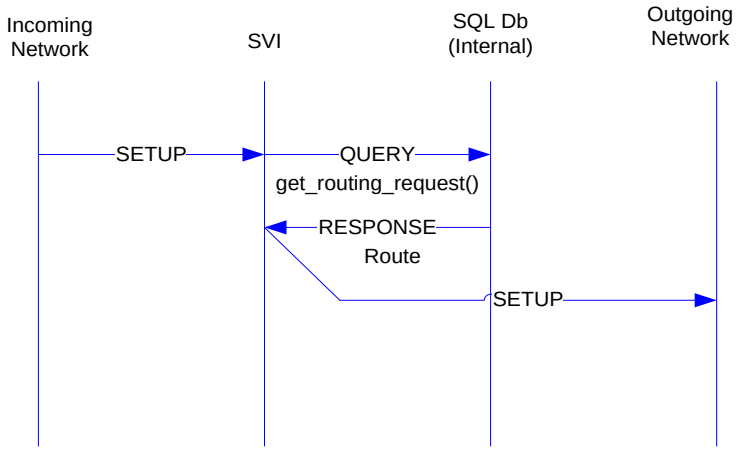


Figure 3: Use Case call gets routed onwards

2.8.3 USE CASE FOR CALL WITH 2 DATABASE LOOKUPS

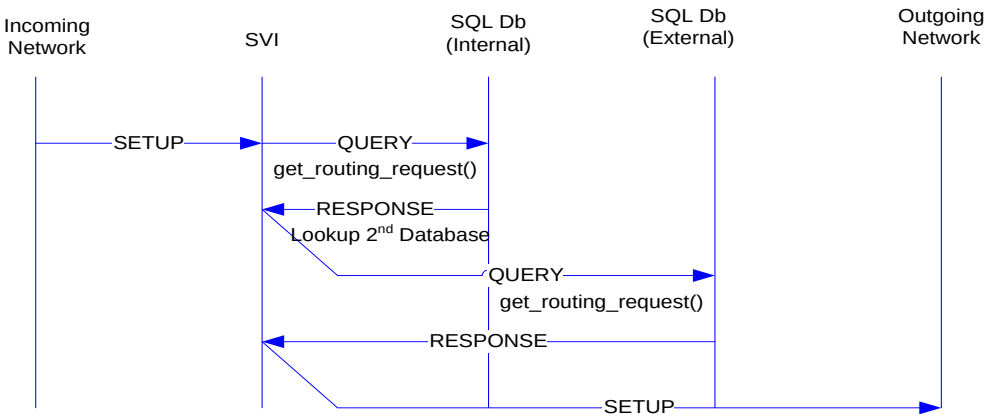


Figure 4: Use Case Two database lookups

3 APPENDIX A – POSTGRESQL DATABASE CODE

Code Listing for PostGreSQL routing Database

3.1 TABLE DEFINITIONS

```
CREATE TABLE routing_table (
    ic_hunt smallint[],
    cdpn character varying(40),
    og_hunt smallint[]
);

CREATE TABLE icrules_table (
    ic_hunt smallint,
    matchcdpn character varying(40),
    cdpnstrip smallint,
    cdpnprefix character varying(40),
    cgpnstrip smallint,
    cgpnprefix character varying(40),
    announcement smallint[],
    whitelist boolean
);

CREATE TABLE announcement_table (
    anntype character varying,
    ann smallint
);
```

```
CREATE TABLE ogrules_table (
    og_hunt smallint,
    matchcdpn character varying(40),
    matchcgpn character varying(40),
    cdpnstrip smallint,
    cdpnprefix character varying(40),
    cgpnstrip smallint,
    cgpnprefix character varying(40),
    cdpn_sc boolean,
    cdpn_noa smallint,
    cgpn_noa smallint,
    cgpn_pli smallint,
    cgpn_si smallint,
    cgpn_br smallint,
    npdatabase smallint,
    mx_rttmpt smallint,
    rttmpt_cds smallint[],
    og_protocol smallint
);

CREATE TABLE whitelist_table (
    subscriber character varying(40),
    authorised boolean
);

CREATE TABLE ann_and_rjcause_table (
    actn smallint,
    ann smallint[],
    q850_cause smallint,
    sip_cause smallint
```

);

3.2 TYPE DEFINITIONS

--Return type for get_routing_info_v4

```
CREATE TYPE routing_info_result_type AS (
        api_vn integer,
        ic_call_id character varying(512),
        err smallint,
        actn smallint,
        rjct_cd smallint,
        rjct_an smallint[],
        usr character varying(40),
        prty smallint,
        cdr_id integer,
        dst character varying(40),
        dst_ip inet,
        dst_prt smallint,
        cdpn_sc boolean,
        cdpn_dgts character varying(40),
        cdpn_noa smallint,
        cgpn_dgts character varying(40),
        cgpn_noa smallint,
        cgpn_clip smallint,
        cgpn_si smallint,
        og_prtcl smallint,
        og_br smallint,
        prg_tns smallint,
        sip_clip boolean,
```

```
        sip_uep boolean,
        sip_vn character varying(40),
        h323_fs boolean,
        h323_h245_tn boolean,
        h323_muc boolean,
        h323_mac boolean,
        h323_ka boolean,
        nat boolean,
        mx_rttmpt smallint,
        rttmpt_cds smallint[],
        prxy boolean,
        codec_ad boolean,
        codecs smallint[],
        sql_lookup smallint
);

--The types below used internally by database

CREATE TYPE icrules_type AS (
                                cdpnstrip smallint,
                                cdpnprefix varchar(40),
                                cgpnstrip smallint,
                                cgpnprefix varchar(40),
                                whitelist bool
                                );

CREATE TYPE new_digits_type AS (
                                cdpn varchar(40),
                                cgpn varchar(40)
                                );
```

```
CREATE TYPE ogrules_type AS (
                                        cdpnstrip smallint,
                                        cdpnprefix varchar(40),
                                        cgpnstrip smallint,
                                        cgpnprefix varchar(40),
                                        cdpn_noa smallint,
                                        cgpn_noa smallint,
                                        cdpn_sc bool,
                                        cgpn_pli smallint, --new
                                        cgpn_si smallint,  --new
                                        cgpn_br smallint,  --new
                                        npdatabase smallint,
                                        mx_rttmpt smallint,
                                        rttmpt_cds smallint[],
                                        og_protocol smallint,
                                        announcement smallint[]
                                        );
```

3.3 HELPER FUNCTIONS

```
--Helper functions for get_routing_info
********************************************************************************

CREATE LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION icrules_function(ic_huntrules smallint, cdpn varchar(40))
RETURNS icrules_type AS $$


DECLARE


icrules icrules_type;

matches record;

matchlength smallint;
```

```
BEGIN


IF ic_huntrules IS NOT NULL THEN

matchlength = 0;

        FOR matches IN SELECT * FROM icrules_table WHERE ic_hunt = ic_huntrules LOOP

                        IF ((LENGTH(matches.matchcdpn) > 0) AND
(SUBSTRING(cdpn,1,LENGTH(matches.matchcdpn)) = matches.matchcdpn) AND
(matchlength < LENGTH(matches.matchcdpn))) THEN

                                matchlength = LENGTH(matches.matchcdpn);

                                icrules.cdpnstrip       =       matches.cdpnstrip;

           icrules.cdpnprefix   =       matches.cdpnprefix;

           icrules.cgpnstrip    =       matches.cgpnstrip;

           icrules.cgpnprefix   =       matches.cgpnprefix;

           icrules.whitelist    =       matches.whitelist;

                        END IF;


        END LOOP;

        --if no match was found we search again, with a wider matching criteria

        --could be optimised!

        IF icrules IS NULL THEN

                FOR matches IN SELECT * FROM icrules_table WHERE ic_hunt = ic_huntrules
LOOP

                                IF (SUBSTRING(cdpn,1,LENGTH(matches.matchcdpn)) =
matches.matchcdpn) THEN

                                        icrules.cdpnstrip       =       matches.cdpnstrip;

                                        icrules.cdpnprefix      =
        matches.cdpnprefix;

                                        icrules.cgpnstrip       =       matches.cgpnstrip;

                                        icrules.cgpnprefix      =
        matches.cgpnprefix;

                                        icrules.whitelist       =       matches.whitelist;

                                END IF;
```

```
                END LOOP;
        END IF;
END IF;

RETURN icrules;

END;
$$ LANGUAGE plpgsql;

*****************************************************************************

CREATE OR REPLACE FUNCTION digit_manipulation_function(cdpn varchar(40), cgpn
varchar(40), rules icrules_type) RETURNS new_digits_type AS $$
DECLARE
digits new_digits_type;

BEGIN
digits.cdpn = cdpn;
digits.cgpn = cgpn;

IF rules IS NULL THEN
        RETURN digits;
END IF;
IF (rules.cdpnstrip > 0) THEN
        digits.cdpn = SUBSTRING(digits.cdpn, (rules.cdpnstrip + 1), LENGTH(digits.cdpn));
END IF;

IF (rules.cgpnstrip > 0) THEN
        digits.cgpn = SUBSTRING(digits.cgpn, (rules.cgpnstrip + 1), LENGTH(digits.cgpn));
END IF;
```

```
IF (LENGTH(rules.cdpnprefix) > 0 ) THEN
        digits.cdpn = rules.cdpnprefix || digits.cdpn;
END IF;

IF (LENGTH(rules.cgpnprefix) > 0 ) THEN
        digits.cgpn = rules.cgpnprefix || digits.cgpn;
END IF;

RETURN digits;

END;

$$ LANGUAGE plpgsql;
```

****

```
CREATE OR REPLACE FUNCTION whitelist_function (cgpn varchar(40)) RETURNS smallint AS
$$
DECLARE
lookup bool;
BEGIN
 --Return Values 0 not listed, 1 authorised, 2 unauthorised
SELECT authorised INTO lookup FROM whitelist_table WHERE subscriber = cgpn;

IF lookup IS NULL THEN RETURN 0;
ELSEIF lookup THEN RETURN 1;
ELSE RETURN 2;
```

```
END IF;


END;

$$ LANGUAGE plpgsql;



*********************************************************************************


CREATE OR REPLACE FUNCTION getogroute_function(ichunt smallint, iccdpn varchar(40))
RETURNS SETOF smallint AS $$

DECLARE

ogroutes smallint;

count smallint;

innercount smallint;

matches record;

matchlength smallint;


BEGIN

matchlength = 0;


FOR matches IN SELECT * FROM routing_table LOOP

        IF((SUBSTRING(iccdpn,1,LENGTH(matches.cdpn)) = matches.cdpn) AND
(matchlength < LENGTH(matches.cdpn))) THEN

                count = 1;

                WHILE  (matches.ic_hunt[count] IS NOT NULL) LOOP

                        IF matches.ic_hunt[count] = ichunt THEN

                                innercount = 1;

                                WHILE (matches.og_hunt[innercount] IS NOT NULL) LOOP

                                        ogroutes = matches.og_hunt[innercount];

                                        matchlength = LENGTH(matches.cdpn);
```

```
                                        return next ogroutes;
                                        innercount = innercount + 1;
                                    END LOOP;
                                END IF;
                                count= count + 1;
                            END LOOP;
                    END IF;
END LOOP;
RETURN;
END;
$$ LANGUAGE plpgsql;
```

****

```
CREATE OR REPLACE FUNCTION ogrules_function (hunt smallint, cdpn varchar(40)) RETURNS
ogrules_type AS $$
DECLARE
ogrules ogrules_type;
matches record;

BEGIN

IF (hunt IS NOT NULL) THEN

        FOR matches IN SELECT * FROM ogrules_table WHERE og_hunt = hunt LOOP
                IF ((LENGTH(matches.matchcdpn) > 0) AND
(SUBSTRING(cdpn,1,LENGTH(matches.matchcdpn)) = matches.matchcdpn)) THEN
                    ogrules.cdpnstrip           = matches.cdpnstrip;
                    ogrules.cdpnprefix          = matches.cdpnprefix;
```

```
                ogrules.cgpnstrip           = matches.cgpnstrip;
                ogrules.cgpnprefix          = matches.cgpnprefix;
                ogrules.cdpn_sc             = matches.cdpn_sc;
                ogrules.cdpn_noa            = matches.cdpn_noa;
                ogrules.cgpn_noa            = matches.cgpn_noa;
                ogrules.cgpn_pli            = matches.cgpn_pli;
                ogrules.cgpn_si             = matches.cgpn_si;
                ogrules.cgpn_br     = matches.cgpn_br;
                ogrules.npdatabase          = matches.npdatabase;
                ogrules.mx_rttmpt           = matches.mx_rttmpt;
                ogrules.rttmpt_cds          = matches.rttmpt_cds;
                ogrules.og_protocol         = matches.og_protocol;
                ogrules.announcement  = matches.announcement;
            END IF;

    END LOOP;
    --if no match was found we search again, with a wider matching criteria
    --could be optimised!

    IF ogrules IS NULL THEN

        FOR matches IN SELECT * FROM ogrules_table WHERE og_hunt = hunt LOOP
            IF (LENGTH(matches.matchcdpn) = 0 ) THEN
                ogrules.cdpnstrip           = matches.cdpnstrip;
                ogrules.cdpnprefix          = matches.cdpnprefix;

                ogrules.cgpnstrip           = matches.cgpnstrip;
                ogrules.cgpnprefix          = matches.cgpnprefix;
                ogrules.cdpn_sc             = matches.cdpn_sc;
                ogrules.cdpn_noa            = matches.cdpn_noa;
```

```
                    ogrules.cgpn_noa            = matches.cgpn_noa;
                    ogrules.cgpn_pli            = matches.cgpn_pli;
                    ogrules.cgpn_si             = matches.cgpn_si;
                    ogrules.cgpn_br     = matches.cgpn_br;
                    ogrules.npdatabase          = matches.npdatabase;
                    ogrules.mx_rttmpt           = matches.mx_rttmpt;
                    ogrules.rttmpt_cds          = matches.rttmpt_cds;
                    ogrules.og_protocol         = matches.og_protocol;
                    ogrules.announcement  = matches.announcement;

                END IF;

            END LOOP;

        END IF;

END IF;


RETURN ogrules;

END;
$$ LANGUAGE plpgsql;
```

3.4 GET_ROUTING_INFO_V4()

```
CREATE OR REPLACE FUNCTION get_routing_info_v4
  (api_vn INTEGER,
   ic_call_id VARCHAR(512),
   ic_br SMALLINT,
   cdpn_sc BOOL,
   cdpn_dgts VARCHAR(40),
   cdpn_noa SMALLINT,
   cgpn_dgts VARCHAR(40),
   cgpn_noa SMALLINT,
   cgpn_clip SMALLINT,
   cgpn_si SMALLINT,
   src VARCHAR(40),
   src_ip INET,
   src_prt INTEGER,
   ic_prtcl SMALLINT,
        usr   VARCHAR(40),
        alg    VARCHAR(40),
        realm   VARCHAR(40),
        nonce   VARCHAR(40),
        cnonce  VARCHAR(40),
        ncount  VARCHAR(40),
        qop     VARCHAR(40),
        method  VARCHAR(40),
        uri     VARCHAR(40),
        response VARCHAR(40)
   )
    RETURNS SETOF routing_info_result_type AS $$
```

```
-- DESCRIPTION
-- Attempts to find valid routes for the specified call details.
-- Returns list of routes if successful.



DECLARE
  routing_info routing_info_result_type;
  icdigits new_digits_type;
  ogdigits new_digits_type;
  icrules icrules_type;
  ogrules ogrules_type;
  whitelistresult smallint;
  ogroute smallint;
BEGIN

  -- fill in mandatory non-route specific response data
  -- assume api version 1.0
  routing_info.api_vn = 256;
  routing_info.ic_call_id = ic_call_id;
  routing_info.actn = 1; -- default is unauthorised

  -- check for invalid input data
  -- set error code accordingly
  -- return if error detected
  IF (api_vn < 0 OR ic_br < 1 OR cdpn_noa < 1)  -- OR src_prt < 0 OR ic_prtcl < 0 )
                                                                    THEN --OR cdpn_sc
IS false OR cdpn_dgts IS NOT NULL) THEN
     routing_info.err = 2; -- invalid input data
     RETURN  next routing_info;
          RETURN;
```

```
      -- finish
  ELSE
    routing_info.err = 1; -- no errors
  END IF;


--find icrules associated with this hg and perform whitelist (if required)
SELECT * INTO icrules FROM icrules_function(CAST(src AS smallint), cdpn_dgts);
IF icrules IS NOT NULL THEN
        SELECT * INTO icdigits FROM digit_manipulation_function(cdpn_dgts, cgpn_dgts,
icrules);
        IF (icrules.whitelist) THEN
                SELECT * INTO whitelistresult FROM whitelist_function(icdigits.cgpn);
                IF (whitelistresult = 0 OR whitelistresult = 2) THEN
                --unauthorised
                routing_info.actn = 1;
                SELECT ann INTO routing_info.rjct_an FROM ann_and_rjcause_table WHERE
actn = routing_info.actn;
                IF (ic_prtcl = 1 ) THEN
                        SELECT sip_cause INTO routing_info.rjct_cd FROM
ann_and_rjcause_table WHERE actn = routing_info.actn;
                ELSE
                        SELECT q850_cause INTO routing_info.rjct_cd FROM
ann_and_rjcause_table WHERE actn = routing_info.actn;
                END IF;
                RETURN next routing_info;
                RETURN;
                END IF;
        END IF;
END IF;
```

```
--if no rules were associated with this incoming hg

IF icrules IS NOT NULL THEN

        routing_info.cdpn_dgts = icdigits.cdpn;

        routing_info.cgpn_dgts = icdigits.cgpn;

ELSE

        routing_info.cdpn_dgts = cdpn_dgts;

        routing_info.cgpn_dgts = cgpn_dgts;

END IF;


--Repeat the following for each route found

FOR ogroute IN SELECT * FROM getogroute_function(CAST(src as smallint),
routing_info.cdpn_dgts) LOOP


IF icrules IS NOT NULL THEN

        routing_info.cdpn_dgts = icdigits.cdpn;

        routing_info.cgpn_dgts = icdigits.cgpn;

ELSE

        routing_info.cdpn_dgts = cdpn_dgts;

        routing_info.cgpn_dgts = cgpn_dgts;

END IF;

        routing_info.dst = CAST(ogroute as varchar);

        routing_info.actn = 4;

        SELECT * INTO ogrules FROM ogrules_function(ogroute, routing_info.cdpn_dgts);

        --IF ogrules IS NOT NULL THEN

                icrules.cdpnstrip = ogrules.cdpnstrip;

                icrules.cdpnprefix = ogrules.cdpnprefix;

                icrules.cgpnstrip = ogrules.cgpnstrip;

                icrules.cgpnprefix = ogrules.cgpnprefix;

                SELECT * INTO ogdigits FROM
digit_manipulation_function(routing_info.cdpn_dgts, routing_info.cgpn_dgts, icrules);
```

```
routing_info.cdpn_dgts = ogdigits.cdpn;
routing_info.cgpn_dgts = ogdigits.cgpn;

IF ogrules.cdpn_noa IS NULL THEN
        routing_info.cdpn_noa = cdpn_noa;
ELSE
        routing_info.cdpn_noa = ogrules.cdpn_noa;
END IF;

IF ogrules.cgpn_noa IS NULL THEN
        routing_info.cgpn_noa = cgpn_noa;
ELSE
        routing_info.cgpn_noa = ogrules.cgpn_noa;
END IF;

routing_info.og_br = ogrules.cgpn_br;
routing_info.cgpn_clip = ogrules.cgpn_pli;
routing_info.cgpn_si = ogrules.cgpn_si;

IF ogrules.mx_rttmpt IS NULL THEN
        routing_info.mx_rttmpt = 0;
ELSE
        routing_info.mx_rttmpt = ogrules.mx_rttmpt;
END IF;
IF ogrules.rttmpt_cds IS NOT NULL THEN
        routing_info.rttmpt_cds = ogrules.rttmpt_cds;
END IF;

IF ogrules.announcement IS NOT NULL THEN
        routing_info.rjct_an = ogrules.announcement;
```

```
                END IF;

                IF ogrules.npdatabase IS NOT NULL THEN
                        routing_info.sql_lookup = ogrules.npdatabase;
                        routing_info.actn = 5;
                END IF;

                IF ogrules.og_protocol IS NULL THEN
                        routing_info.og_prtcl = 5;
                ELSE
                        routing_info.og_prtcl = ogrules.og_protocol;
                END IF;

                --routing_info.rjct_an = ogrules.announcement;
        --END IF;

        RETURN next routing_info;
END LOOP;

--if no ogroute was found
IF ogroute IS NULL THEN
                routing_info.actn = 5;
                SELECT ann INTO routing_info.rjct_an FROM ann_and_rjcause_table WHERE
actn = routing_info.actn;
                IF (ic_prtcl = 1 ) THEN
                        SELECT sip_cause INTO routing_info.rjct_cd FROM
ann_and_rjcause_table WHERE actn = routing_info.actn;
                ELSE
                        SELECT q850_cause INTO routing_info.rjct_cd FROM
ann_and_rjcause_table WHERE actn = routing_info.actn;
                END IF;
```

```
                return next routing_info;
END IF;


RETURN;
END;
$$ LANGUAGE plpgsql;
```