Documentation

Carnivora

A powerfull backend for web-service management

September 1, 2015

Contents

1 Module “_postgresql_user”

`PostgreSQL users and their priviledges`

1.1 Roles

1.1.1 Role “edentata”

Account for edentata web frontend

- Login: true

1.1.2 Role “machine_example”

Account for machine example

- Login: true

2 Module “backend”

```
Carnivora Backend

The backend module provides everything required for the backend API.
The backend API delivers content required for building configs etc.
to clients, called machines.
```

2.1 Tables

2.1.1 Table “backend”.“auth”

```
Grants rights to backend API clients based on SQL roles.
```

- Primary key:
  - role

Columns

role

```
Grantee for right to access the backend date for the defined machine.
A role is basically a user or a user group on the SQL server.
```

- Type: commons.t_key

machine

```
Machine for which the rights are granted.
```

- Type: dns.t_domain
- References: backend.machine.name
  - On delete: CASCADE

2.1.2 Table “backend”.“machine”

```
Physical or virtual machines that hosts services.
```

- Primary key:
  - name

Columns

name

```
Machine name
```

- Type: dns.t_domain

2.2 Functions

2.2.1 Function “backend”.“_active”

```
Is not 'del'
```

- Parameters:
  - **backend_status** *backend.t_status*
- Returns: boolean

2.2.2 Function “backend”.“_conditional_notify”

```
Notifies if first argument is true. Throws inaccessible otherwise.
```

- Parameters:
  - **p_condition** *boolean*
  - **p_service** *commons.t_key*
  - **p_subservice** *commons.t_key*
  - **p_domain** *dns.t_domain*
- Returns: void

2.2.3 Function “backend”.“_conditional_notify_service_entity_name”

```
Notifies if first argument is true. Throws inaccessible otherwise.
```

- Parameters:
  - **p_condition** *boolean*
  - **p_service_entity_name** *dns.t_domain*
  - **p_service** *commons.t_key*
  - **p_subservice** *commons.t_key*
- Returns: void

2.2.4 Function “backend”.“_deleted”

```
Is 'del'
```

- Parameters:
  - **backend_status** *backend.t_status*
- Returns: boolean

2.2.5 Function “backend”.“_get_login”

```
Shows informations for the current backend login.
Throws an error if the current user is not a grantee
for a machine.
```

- Parameters: *non*
- Returns: TABLE

2.2.6 Function "backend"."_machine_priviledged"

```
Checks if a currently connected machine is priviledged to obtain data for
a certain service for a certain domain name.

WARNING: The parameter p_domain must be a domain, which means an entry in
the column dns.service.domain. It must not be confused with a service_entity_name.
```

- Parameters:
  - **p_service** *commons.t_key*
  - **p_domain** *dns.t_domain*
  - **p_include_inactive** *boolean*
- Variables defined for body:
  - **v_machine** *dns.t_domain*
- Returns: boolean
- Execute privilege:
  - backend

2.2.7 Function "backend"."_machine_priviledged_service"

```
Checks if a currently connected machine is priviledged to obtain data for
a certain service for a certain servicee name.

WARNING: The parameter p_server_name must be a service name. It must not be
confused with a domain.
```

- Parameters:
  - **p_service** *commons.t_key*
  - **p_service_entity_name** *dns.t_domain*
  - **p_include_inactive** *boolean*
- Variables defined for body:
  - **v_machine** *dns.t_domain*
- Returns: boolean
- Execute privilege:
  - backend

2.2.8 Function "backend"."_notify"

```
Informs all machines about changes.

To listen to signals use LISTEN "carnivora/machine.name.example".
The payload has the form 'mail.domain.example/email/list'.
```

- Parameters:
  - **p_machine** *dns.t_domain*
  - **p_service_entity_name** *dns.t_domain*
  - **p_service** *commons.t_key*
  - **p_subservice** *commons.t_key*
- Returns: void

2.2.9 Function "backend"."_notify_domain"

```
Informs all machines about changes.

WARNING: The parameter p_domain must be a domain, which means an entry in
the column dns.service.domain. It must not be confused with a service_entity_name.
```

- Parameters:
  - **p_service** *commons.t_key*
  - **p_subservice** *commons.t_key*
  - **p_domain** *dns.t_domain*
- Returns: void

2.2.10 Function "backend"."_notify_service_entity_name"

```
Informs all machines about changes.

WARNING: The parameter p_service_entity_name must be a servcie name. It must not be
confused with a domain.
```

- Parameters:
  - **p_service_entity_name** *dns.t_domain*
  - **p_service** *commons.t_key*
  - **p_subservice** *commons.t_key*
- Returns: void

2.3 Domains

2.3.1 Domain "backend"."t_status"

```
Backend status
```

2.4 Roles

2.4.1 Role "backend"

vms

- Login:

3 Module “commons”

```
Carnivora Commons

Usefull templates, functions and domains.
```

3.1 Functions

3.1.1 Function “commons”.“_hash_password”

```
SHA512 hash of the password with 16 charcters random salt.
The returned format is the traditional 'crypt(3)' format.
```

- Parameters:
  - **p_password** *commons.t_password_plaintext*
- Language: plpython3u
- Returns: commons.t_password

3.1.2 Function “commons”.“_idn”

```
Converts a unicode domain name to IDN (ASCII)
```

- Parameters:
  - **p_domain** *varchar*
- Language: plpython3u
- Returns: varchar
- Execute privilege:
  - userlogin
  - backend

3.1.3 Function “commons”.“_passwords_equal”

```
Compares a plaintext password with an arbitrary 'crypt(3)' hashed password.

Uses
```

- Parameters:
  - **p_password_plaintext** *commons.t_password_plaintext*
  - **p_password_hash** *commons.t_password*
- Language: plpython3u
- Returns: boolean

3.1.4 Function “commons”.“_raise_inaccessible_or_missing”

```
Raised whenever a operation on an object failes because it is not owned by
the user or it is not found.
```

- Parameters:
  - **p_raise** *boolean* Controls if the exception is raised
- Returns: void

3.1.5 Function "commons"."_reverse_array"

`Copied from`

- Parameters:
  - **p_array** *anyarray*
- Returns: anyarray
- Execute privilege:
  - userlogin
  - backend

3.1.6 Function "commons"."_uuid"

`Returns a random uuid`

- Parameters: *non*
- Returns: uuid

3.2 Domains

3.2.1 Domain "commons"."t_port"

`Port`

3.2.2 Domain "commons"."t_password"

`unix hash thingy - todo: propper checking of format`

3.2.3 Domain "commons"."t_password_plaintext"

`Password in plaintext`

3.2.4 Domain "commons"."t_key"

`Key`

4 Module “dns”

DNS

4.1 Tables

4.1.1 Table “dns”.“custom”

Direct name server entries.

- Primary key:
  - id

Columns

type

Type (?) like MX, A, AAAA, ...

- Type: dns.t_type

rdata

fancy rdata storage

- Type: dns.t_rdata

ttl

Time to live, NULL indicates default value

- Type: dns.t_ttl
- Can be *NULL*

backend_status

Status of database entry in backend. NULL: nothing pending, 'ins': entry not present on backend client, 'upd': update pending on backend client, 'del': deletion peding on backend client.

- Type: backend.t_status
- Can be *NULL*
- Default value: `ins'

registered

Registered domain of which domain is a sub domain

- Type: dns.t_domain
- References: dns.registered.domain

domain

`domain of entry`

- Type: dns.t_domain

id

```
uuid serial number to identify database elements uniquely
The default value is generated using uuid_generate_v4().
```

- Type: uuid
- Default value: uuid_generate_v4()

4.1.2 Table "dns"."registered"

`Domains registered under a public suffix.`

- Primary key:
  - domain
- Foreign keys:
  1. **Reference service entity**
     - Columns:
       a) service_entity_name →
       b) service →
     - Referenced columns:
       a) system.service_entity.service_entity_name
       b) system.service_entity.service
  2. **Reference subservice entity**
     - Columns:
       a) service_entity_name →
       b) service →
       c) subservice →
     - Referenced columns:
       a) system.subservice_entity.service_entity_name
       b) system.subservice_entity.service
       c) system.subservice_entity.subservice

Columns

owner

`for ownage`

- Type: user.t_user
- References: user.user.owner

backend_status

```
Status of database entry in backend. NULL: nothing pending,
'ins': entry not present on backend client, 'upd': update
pending on backend client, 'del': deletion peding on
backend client.
```

- Type: backend.t_status
- Can be *NULL*
- Default value: 'ins'

service_entity_name

`Service entity name`

- Type: dns.t_domain

service

`Service (e.g. email, jabber)`

- Type: commons.t_key

subservice

`Subservice (e.g. account, alias)`

- Type: commons.t_key

domain

`Domain`

- Type: dns.t_domain

public_suffix

`Public Suffix`

- Type: varchar

4.1.3 Table "dns"."service"

`Name server entries based on system.service (i.e. system.service_dns)`

- Primary key:
  - domain
  - service
- Foreign keys:
  1. **Reference service entity**
     - Columns:
       a) service_entity_name →
       b) service →
     - Referenced columns:
       a) system.service_entity.service_entity_name
       b) system.service_entity.service

Columns

service_entity_name

`Service entity name`

- Type: dns.t_domain

service

`Service (e.g. email, jabber)`

- Type: commons.t_key

backend_status

```
Status of database entry in backend. NULL: nothing pending,
'ins': entry not present on backend client, 'upd': update
pending on backend client, 'del': deletion peding on
backend client.
```

- Type: backend.t_status
- Can be *NULL*
- Default value: \`ins'

registered

`Registered domain of which domain is a sub domain`

- Type: dns.t_domain
- References: dns.registered.domain

domain

`domain for which the entries should be created`

- Type: dns.t_domain

4.2 Functions

4.2.1 Function "dns"."_domain_order"

`ORDER`

- Parameters:
  - **p_domain** *dns.t_domain*
- Returns: varchar()
- Execute privilege:
  - userlogin
  - backend

4.2.2 Function "dns"."del_custom"

`Delete Custom`

- Parameters:
  - **p_id** *uuid*
- Variables defined for body:
  - **v_nameserver** *dns.t_domain*
  - **v_managed** *commons.t_key*
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

4.2.3 Function "dns"."del_registered"

`Delete registered domain`

- Parameters:
  - **p_domain** *dns.t_domain*
- Variables defined for body:
  - **v_nameserver** *dns.t_domain*
  - **v_managed** *commons.t_key*
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

4.2.4 Function "dns"."del_service"

`deletes all service entries of a specific domain`

- Parameters:
  - **p_domain** *dns.t_domain*
  - **p_service** *commons.t_key*
- Variables defined for body:
  - **v_nameserver** *dns.t_domain*
  - **v_managed** *commons.t_key*
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

4.2.5 Function "dns"."fwd_registered_status"

`Update status`

- Parameters:
  - **p_domain** *dns.t_domain*
  - **p_backend_status** *backend.t_status*
  - **p_include_inactive** *boolean*
- Variables defined for body:
  - **v_machine** *dns.t_domain*
- Returns: void
- Execute privilege:
  - backend

4.2.6 Function "dns"."ins_custom"

`Ins Custom`

- Parameters:
  - **p_registered** *dns.t_domain*
  - **p_domain** *dns.t_domain*
  - **p_type** *dns.t_type*
  - **p_rdata** *dns.t_rdata*
  - **p_ttl** *integer*
- Variables defined for body:
  - **v_nameserver** *dns.t_domain*
  - **v_managed** *commons.t_key*
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

4.2.7 Function "dns"."ins_registered"

`registeres new domain`

- Parameters:
  - **p_domain** *dns.t_domain*
  - **p_subservice** *commons.t_key*
  - **p_service_entity_name** *dns.t_domain*
  - **p_public_suffix** *varchar*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

4.2.8 Function "dns"."ins_service"

`Creates service dns entry`

- Parameters:
  - **p_registered** *dns.t_domain*
  - **p_domain** *dns.t_domain*
  - **p_service_entity_name** *dns.t_domain*
  - **p_service** *commons.t_key*
- Variables defined for body:
  - **v_nameserver** *dns.t_domain*
  - **v_managed** *commons.t_key*
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*

- Returns: void
- Execute privilege:
  - userlogin

4.2.9 Function "dns"."sel_activatable_service"

```
Activatable services
```

- Parameters: *non*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: TABLE
- Execute privilege:
  - userlogin

4.2.10 Function "dns"."sel_available_service"

```
List all domains that have a service entry in dns with their service.
This is particularly usefull since these domains are ready for use with
this service. Usually this means that accounts etc. can be created for this
domain.
```

- Parameters: *non*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: TABLE
- Execute privilege:
  - userlogin

4.2.11 Function "dns"."sel_custom"

```
sel custom
```

- Parameters: *non*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: TABLE
- Execute privilege:
  - userlogin

4.2.12 Function "dns"."sel_nameserver"

```
Select available nameservers
```

- Parameters: *non*
- Variables defined for body:
  - **v_owner** *user.t_user*

  - **v_login** *user.t_user*

- Returns: TABLE
- Execute privilege:
  - userlogin

4.2.13 Function "dns"."sel_registered"

```
List registered domains
```

- Parameters: *non*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: TABLE
- Execute privilege:
  - userlogin

4.2.14 Function "dns"."sel_service"

```
Select service based dns entries
```

- Parameters: *non*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: TABLE
- Execute privilege:
  - userlogin

4.2.15 Function "dns"."sel_usable_domain"

```
Usable domains
```

- Parameters:
  - **p_service** *commons.t_key*
  - **p_subservice** *commons.t_key*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: TABLE
- Execute privilege:
  - userlogin

4.2.16 Function "dns"."srv_record"

```
Servers both record types combined: Raw entries and the ones assembled
from records templates for services (system.service_entity_dns).
```

- Parameters:
  - **p_include_inactive** *boolean*
- Variables defined for body:
  - **v_machine** *dns.t_domain*
- Returns: TABLE
- Execute privilege:
  - backend

4.2.17 Function "dns"."upd_custom"

```
Ins Custom
```

- Parameters:
  - **p_id** *uuid*
  - **p_rdata** *dns.t_rdata*
  - **p_ttl** *integer*
- Variables defined for body:
  - **v_nameserver** *dns.t_domain*
  - **v_managed** *commons.t_key*
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

4.3 Domains

4.3.1 Domain "dns"."t_domain"

```
Domain name unicode (not IDN)
```

4.3.2 Domain "dns"."t_type"

```
Resource record type
```

4.3.3 Domain "dns"."t_rdata"

```
record entry
```

4.3.4 Domain "dns"."t_ttl"

```
time to live
```

5 Module “domain_reseller”

```
Features for Domains Registered via a Reseller

Stores additional details for dns.registered domains. Also supports storing
contact informations (handles).

This module sends the following signals:
 - domain_reseller/handle
 - domain_registered/managed
 - domain_registered/unmanaged
```

5.1 Tables

5.1.1 Table “domain_reseller”.“handle”

```
Handles (Domain Contacts)

Domain contacts that can be used as owner, admin-c, tech-c or zone-c.
```

- Primary key:
  - alias
- Foreign keys:
  1. **Reference service entity**
     - Columns:
       a) service_entity_name →
       b) service →
     - Referenced columns:
       a) system.service_entity.service_entity_name
       b) system.service_entity.service
  2. **Reference subservice entity**
     - Columns:
       a) service_entity_name →
       b) service →
       c) subservice →
     - Referenced columns:
       a) system.subservice_entity.service_entity_name
       b) system.subservice_entity.service
       c) system.subservice_entity.subservice

Columns

service_entity_name

```
Service entity name
```

- Type: dns.t_domain

service

`Service (e.g. email, jabber)`

- Type: commons.t_key

subservice

`Subservice (e.g. account, alias)`

- Type: commons.t_key

owner

`for ownage`

- Type: user.t_user
- References: user.user.owner

backend_status

```
Status of database entry in backend. NULL: nothing pending,
'ins': entry not present on backend client, 'upd': update
pending on backend client, 'del': deletion peding on
backend client.
```

- Type: backend.t_status
- Can be *NULL*
- Default value: `ins'

alias

`Free choosable alias`

- Type: varchar

id

`Internal id at reseller`

- Type: varchar
- Can be *NULL*

fname

`First name`

- Type: varchar

lname

`Last name`

- Type: varchar

address

`Address`

- Type: varchar

pcode

`Postcode`

- Type: varchar

city

`City`

- Type: varchar

country

`Country`

- Type: varchar

state

`State`

- Type: varchar

email

`Email`

- Type: email.t_address

phone

`Phone`

- Type: varchar

organization

`Organization`

- Type: varchar
- Can be *NULL*

fax

`Fax`

- Type: varchar
- Can be *NULL*

mobile_phone

`Mobile phone`

- Type: varchar
- Can be *NULL*

5.1.2 Table "domain_reseller"."registered"

`Addtional informations to those stored in dns.registered`

- Primary key:
  - domain

Columns

domain

`Domain`

- Type: dns.t_domain
- References: dns.registered.domain
  - On delete: CASCADE

registrant

`Registrant (Owner)`

- Type: varchar
- References: domain_reseller.handle.alias

admin_c

`Admin-C`

- Type: varchar
- References: domain_reseller.handle.alias

tech_c

`Tech-C`

- Type: varchar
- Can be *NULL*
- References: domain_reseller.handle.alias

zone_c

`Zone-C`

- Type: varchar
- Can be *NULL*
- References: domain_reseller.handle.alias

payable

`Payable`

- Type: timestamp
- Can be *NULL*

period

`Renewal period (years)`

- Type: integer
- Can be *NULL*

registrar_status

`Registrar status`

- Type: varchar
- Can be *NULL*

registry_status

`Registry status`

- Type: varchar
- Can be *NULL*

last_status

`Last update status`

- Type: varchar
- Can be *NULL*

5.2 Functions

5.2.1 Function "domain_reseller"."del_handle"

`Deletes handle`

- Parameters:
  - **p_alias** *varchar*
- Variables defined for body:
  - **v_service_entity_name** *dns.t_domain*
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

5.2.2 Function "domain_reseller"."fwd_handle_id"

`Insert handle id`

- Parameters:
  - **p_alias** *varchar*
  - **p_id** *varchar*
  - **p_include_inactive** *boolean*
- Variables defined for body:
  - **v_machine** *dns.t_domain*
- Returns: void
- Execute privilege:
  - backend

5.2.3 Function "domain_reseller"."fwd_registered_status"

`Update status`

- Parameters:
  - **p_domain** *dns.t_domain*
  - **p_payable** *timestamp*
  - **p_period** *integer*
  - **p_registrar_status** *varchar*
  - **p_registry_status** *varchar*
  - **p_last_status** *varchar*
  - **p_include_inactive** *boolean*
- Variables defined for body:
  - **v_machine** *dns.t_domain*
- Returns: void
- Execute privilege:
  - backend

5.2.4 Function "domain_reseller"."ins_handle"

`Inserts handle`

- Parameters:
  - **p_alias** *varchar*
  - **p_service_entity_name** *dns.t_domain*
  - **p_fname** *varchar*
  - **p_lname** *varchar*
  - **p_address** *varchar*
  - **p_pcode** *varchar*
  - **p_city** *varchar*
  - **p_country** *varchar*
  - **p_state** *varchar*
  - **p_email** *email.t_address*
  - **p_phone** *varchar*
  - **p_organization** *varchar*
  - **p_fax** *varchar*
  - **p_mobile_phone** *varchar*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

5.2.5 Function "domain_reseller"."ins_registered"

`Inserts details for registered domain`

- Parameters:
  - **p_domain** *dns.t_domain*

  - **p_registrant** *varchar*
  - **p_admin_c** *varchar*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

5.2.6 Function "domain_reseller"."sel_handle"

`Selects handles`

- Parameters:
  - **p_hide_foreign** *bool*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: SETOF domain_reseller."handle"
- Execute privilege:
  - userlogin

5.2.7 Function "domain_reseller"."sel_registered"

`Selects details for registered domains`

- Parameters: *non*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: TABLE
- Execute privilege:
  - userlogin

5.2.8 Function "domain_reseller"."sel_reseller"

`Selects available resellers`

- Parameters: *non*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: TABLE
- Execute privilege:
  - userlogin

5.2.9 Function "domain_reseller"."srv_handle"

```
Serves handles
```

- Parameters:
  - **p_include_inactive** *boolean*
- Variables defined for body:
  - **v_machine** *dns.t_domain*
- Returns: SETOF domain_reseller."handle"
- Execute privilege:
  - backend

5.2.10 Function "domain_reseller"."srv_registered"

```
Serves details for registered domains
```

- Parameters:
  - **p_include_inactive** *boolean*
- Variables defined for body:
  - **v_machine** *dns.t_domain*
- Returns: TABLE
- Execute privilege:
  - backend

5.2.11 Function "domain_reseller"."upd_handle"

```
Updates handle
```

- Parameters:
  - **p_alias** *varchar*
  - **p_address** *varchar*
  - **p_pcode** *varchar*
  - **p_city** *varchar*
  - **p_country** *varchar*
  - **p_state** *varchar*
  - **p_email** *email.t_address*
  - **p_phone** *varchar*
  - **p_organization** *varchar*
  - **p_fax** *varchar*
  - **p_mobile_phone** *varchar*
- Variables defined for body:
  - **v_service_entity_name** *dns.t_domain*
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

5.2.12 Function "domain_reseller"."upd_registered"

```
Updates details for registered domain
```

- Parameters:
  - **p_domain** *dns.t_domain*
  - **p_admin_c** *varchar*
- Variables defined for body:
  - **v_nameserver** *dns.t_domain*
  - **v_managed** *commons.t_key*
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

6 Module “email”

```
Email and Mailing lists

This module sends the following signals:
 - email/alias
 - email/list
 - email/mailbox
 - email/redirection
```

6.1 Tables

6.1.1 Table “email”.“address”

```
Collection of all known addresses
```

- Primary key:
  - localpart
  - domain
- Foreign keys:
  1. **reference dns (service)**
     - Columns:
       a) domain →
       b) service →
       c) service_entity_name →
     - Referenced columns:
       a) dns.service.domain
       b) dns.service.service
       c) dns.service.service_entity_name
  2. **Reference subservice entity**
     - Columns:
       a) service_entity_name →
       b) service →
       c) subservice →
     - Referenced columns:
       a) system.subservice_entity.service_entity_name
       b) system.subservice_entity.service
       c) system.subservice_entity.subservice

Columns

domain

```
Domain name
```

- Type: dns.t_domain

service

`Service`

- Type: commons.t_key

service_entity_name

`ent. name`

- Type: dns.t_domain

subservice

`Subservice (e.g. account, alias)`

- Type: commons.t_key

localpart

`Local part`

- Type: email.t_localpart

6.1.2 Table "email"."alias"

`Aliases for e-mail mailboxes, owner is determined by mailbox.owner`

- Primary key:
  - localpart
  - domain
- Foreign keys:
  1. **reference dns (service)**
     - Columns:
       a) domain →
       b) service →
       c) service_entity_name →
     - Referenced columns:
       a) dns.service.domain
       b) dns.service.service
       c) dns.service.service_entity_name
  2. **Reference subservice entity**
     - Columns:
       a) service_entity_name →
       b) service →
       c) subservice →
     - Referenced columns:
       a) system.subservice_entity.service_entity_name
       b) system.subservice_entity.service
       c) system.subservice_entity.subservice
  3. **reference to a mailbox**
     - Columns:
       a) mailbox_localpart →
       b) mailbox_domain →
     - Referenced columns:
       a) email.mailbox.localpart
       b) email.mailbox.domain

Columns

domain

`Domain name`

- Type: dns.t_domain

service

`Service`

- Type: commons.t_key

service_entity_name

`ent. name`

- Type: dns.t_domain

subservice

`Subservice (e.g. account, alias)`

- Type: commons.t_key

backend_status

```
Status of database entry in backend. NULL: nothing pending,
'ins': entry not present on backend client, 'upd': update
pending on backend client, 'del': deletion peding on
backend client.
```

- Type: backend.t_status
- Can be *NULL*
- Default value: `ins'

localpart

`Local part`

- Type: email.t_localpart

mailbox_localpart

`Mailbox to which the mails will be delivered`

- Type: email.t_localpart

mailbox_domain

`Mailbox to which the mails will be delivered`

- Type: dns.t_domain

6.1.3 Table "email"."list"

`Mailing lists`

- Primary key:
  - localpart
  - domain
- Foreign keys:
  1. **reference dns (service)**
     - Columns:
       a) domain →
       b) service →
       c) service_entity_name →
     - Referenced columns:
       a) dns.service.domain
       b) dns.service.service
       c) dns.service.service_entity_name
  2. **Reference subservice entity**
     - Columns:
       a) service_entity_name →
       b) service →
       c) subservice →
     - Referenced columns:
       a) system.subservice_entity.service_entity_name
       b) system.subservice_entity.service
       c) system.subservice_entity.subservice

Columns

domain

`Domain name`

- Type: dns.t_domain

service

`Service`

- Type: commons.t_key

service_entity_name

`ent. name`

- Type: dns.t_domain

subservice

`Subservice (e.g. account, alias)`

- Type: commons.t_key

owner

`for ownage`

- Type: user.t_user
- References: user.user.owner

backend_status

```
Status of database entry in backend. NULL: nothing pending,
'ins': entry not present on backend client, 'upd': update
pending on backend client, 'del': deletion peding on
backend client.
```

- Type: backend.t_status
- Can be *NULL*
- Default value: `ins'

option

`Free options in JSON format`

- Type: jsonb
- Default value: `{}'

localpart

`Local part of the email list address`

- Type: email.t_localpart

admin

`Email address of the list admin`

- Type: email.t_address

options

`Arbitrary options`

- Type: jsonb
- Can be *NULL*

6.1.4 Table "email"."list_subscriber"

`list subscribers`

- Primary key:
  - address
  - list_localpart
  - list_domain
- Foreign keys:
  1. **reference to a list**
     - Columns:
       a) list_localpart →
       b) list_domain →
     - Referenced columns:
       a) email.list.localpart
       b) email.list.domain

Columns

backend_status

```
Status of database entry in backend. NULL: nothing pending,
'ins': entry not present on backend client, 'upd': update
pending on backend client, 'del': deletion peding on
backend client.
```

- Type: backend.t_status
- Can be *NULL*
- Default value: `ins'

option

```
Free options in JSON format
```

- Type: jsonb
- Default value: `{}'

address

```
Subscribers address
```

- Type: email.t_address

list_localpart

```
List
```

- Type: email.t_localpart

list_domain

```
List
```

- Type: dns.t_domain

6.1.5 Table "email"."mailbox"

```
E-mail mailboxs correspond to something a mail user can login into. Basically
a mailbox represents a mailbox. A mailbox is bound to a specific address.
Further addresses can be linked to mailboxs via aliases.
```

- Primary key:
  - localpart
  - domain
- Foreign keys:
  1. **reference dns (service)**
     - Columns:
       a) domain →
       b) service →
       c) service_entity_name →
     - Referenced columns:
       a) dns.service.domain
       b) dns.service.service

    c) dns.service.service_entity_name

2. **Reference subservice entity**
   - Columns:
     a) service_entity_name →
     b) service →
     c) subservice →
   - Referenced columns:
     a) system.subservice_entity.service_entity_name
     b) system.subservice_entity.service
     c) system.subservice_entity.subservice

Columns

domain

`Domain name`

- Type: dns.t_domain

service

`Service`

- Type: commons.t_key

service_entity_name

`ent. name`

- Type: dns.t_domain

subservice

`Subservice (e.g. account, alias)`

- Type: commons.t_key

owner

`for ownage`

- Type: user.t_user
- References: user.user.owner

backend_status

```
Status of database entry in backend. NULL: nothing pending,
'ins': entry not present on backend client, 'upd': update
pending on backend client, 'del': deletion peding on
backend client.
```

- Type: backend.t_status
- Can be *NULL*
- Default value: `ins'

option

`Free options in JSON format`

- Type: jsonb
- Default value: `{}'

localpart

`Local part`

- Type: email.t_localpart

uid

`Unix user identifier`

- Type: SERIAL

password

`Unix shadow crypt format`

- Type: commons.t_password

quota

`Quota for mailbox in MiB`

- Type: int
- Can be *NULL*

6.1.6 Table "email"."redirection"

`Redirections`

- Primary key:
  - localpart
  - domain
- Foreign keys:
  1. **reference dns (service)**
     - Columns:
       a) domain →
       b) service →
       c) service_entity_name →
     - Referenced columns:
       a) dns.service.domain
       b) dns.service.service
       c) dns.service.service_entity_name
  2. **Reference subservice entity**
     - Columns:
       a) service_entity_name →
       b) service →
       c) subservice →
     - Referenced columns:
       a) system.subservice_entity.service_entity_name
       b) system.subservice_entity.service
       c) system.subservice_entity.subservice

Columns

domain

`Domain name`

- Type: dns.t_domain

service

`Service`

- Type: commons.t_key

service_entity_name

`ent. name`

- Type: dns.t_domain

subservice

`Subservice (e.g. account, alias)`

- Type: commons.t_key

owner

`for ownage`

- Type: user.t_user
- References: user.user.owner

backend_status

```
Status of database entry in backend. NULL: nothing pending,
'ins': entry not present on backend client, 'upd': update
pending on backend client, 'del': deletion peding on
backend client.
```

- Type: backend.t_status
- Can be *NULL*
- Default value: `ins'

localpart

`Local part`

- Type: email.t_localpart

destination

`External address to which the mails will be delivered`

- Type: email.t_address

6.2 Functions

6.2.1 Function "email"."_address"

`List all addresses`

- Parameters: *non*
- Returns: TABLE

6.2.2 Function "email"."_address_valid"

`x`

- Parameters:
  - **p_localpart** *email.t_localpart*
  - **p_domain** *dns.t_domain*
- Returns: void

6.2.3 Function "email"."del_alias"

`Delete Alias`

- Parameters:
  - **p_localpart** *email.t_localpart*
  - **p_domain** *dns.t_domain*
  - **p_mailbox_localpart** *email.t_localpart*
  - **p_mailbox_domain** *dns.t_domain*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

6.2.4 Function "email"."del_list"

`Delete mailing list`

- Parameters:
  - **p_domain** *dns.t_domain*
  - **p_localpart** *email.t_localpart*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

6.2.5 Function "email"."del_list_subscriber"

`del`

- Parameters:
  - **p_list_localpart** *email.t_localpart*
  - **p_list_domain** *dns.t_domain*
  - **p_address** *email.t_address*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

6.2.6 Function "email"."del_mailbox"

`Delete mailbox`

- Parameters:
  - **p_localpart** *email.t_localpart*
  - **p_domain** *dns.t_domain*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

6.2.7 Function "email"."del_redirection"

`Delete redirection`

- Parameters:
  - **p_localpart** *email.t_localpart*
  - **p_domain** *dns.t_domain*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

6.2.8 Function "email"."ins_alias"

`Create e-mail aliases`

- Parameters:
  - **p_localpart** *email.t_localpart*
  - **p_domain** *dns.t_domain*
  - **p_mailbox_localpart** *email.t_localpart*
  - **p_mailbox_domain** *dns.t_domain*
- Variables defined for body:
  - **v_subservice** *commons.t_key* (default: *'alias'*)
  - **v_num_total** *int*
  - **v_num_domain** *int*
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

6.2.9 Function "email"."ins_list"

`Creates a mailing list`

- Parameters:
  - **p_localpart** *email.t_localpart*
  - **p_domain** *dns.t_domain*
  - **p_admin** *email.t_address*
- Variables defined for body:
  - **v_subservice** *commons.t_key* (default: *'list'*)
  - **v_num_total** *int*
  - **v_num_domain** *int*
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

6.2.10 Function "email"."ins_list_subscriber"

`Adds a subscriber to a mailing list`

- Parameters:
  - **p_address** *email.t_address*
  - **p_list_localpart** *email.t_localpart*
  - **p_list_domain** *dns.t_domain*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

6.2.11 Function "email"."ins_mailbox"

`Creates an email box`

- Parameters:
  - **p_localpart** *email.t_localpart*
  - **p_domain** *dns.t_domain*
  - **p_password** *commons.t_password_plaintext*
- Variables defined for body:
  - **v_subservice** *commons.t_key* (default: *'mailbox'*)
  - **v_num_total** *int*
  - **v_num_domain** *int*
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

6.2.12 Function "email"."ins_redirection"

`Creates a redirection`

- Parameters:
  - **p_localpart** *email.t_localpart*
  - **p_domain** *dns.t_domain*
  - **p_destination** *email.t_address*
- Variables defined for body:
  - **v_subservice** *commons.t_key* (default: *'redirection'*)
  - **v_num_total** *int*
  - **v_num_domain** *int*
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

6.2.13 Function "email"."sel_alias"

`Select aliases`

- Parameters: *non*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: TABLE
- Execute privilege:
  - userlogin

6.2.14 Function "email"."sel_list"

```
List all lists
```

- Parameters: *non*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: TABLE
- Execute privilege:
  - userlogin

6.2.15 Function "email"."sel_list_subscriber"

```
a
```

- Parameters: *non*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: TABLE
- Execute privilege:
  - userlogin

6.2.16 Function "email"."sel_mailbox"

```
List all mailboxes
```

- Parameters: *non*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: TABLE
- Execute privilege:
  - userlogin

6.2.17 Function "email"."sel_redirection"

```
Lists all redirections
```

- Parameters: *non*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: TABLE
- Execute privilege:
  - userlogin

6.2.18 Function "email"."srv_alias"

`Lists all email aliases`

- Parameters:
  - **p_include_inactive** *boolean*
- Variables defined for body:
  - **v_machine** *dns.t_domain*
- Returns: TABLE
- Execute privilege:
  - backend

6.2.19 Function "email"."srv_list"

`Lists all mailinglists`

- Parameters:
  - **p_include_inactive** *boolean*
- Variables defined for body:
  - **v_machine** *dns.t_domain*
- Returns: TABLE
- Execute privilege:
  - backend

6.2.20 Function "email"."srv_list_subscriber"

`Lists all mailinglist subscribers`

- Parameters:
  - **p_include_inactive** *boolean*
- Variables defined for body:
  - **v_machine** *dns.t_domain*
- Returns: TABLE
- Execute privilege:
  - backend

6.2.21 Function "email"."srv_mailbox"

`Lists all mailboxes`

- Parameters:
  - **p_include_inactive** *boolean*
- Variables defined for body:
  - **v_machine** *dns.t_domain*
- Returns: TABLE
- Execute privilege:
  - backend

6.2.22 Function "email"."srv_redirection"

`Lists all mailinglists`

- Parameters:
  - **p_include_inactive** *boolean*
- Variables defined for body:
  - **v_machine** *dns.t_domain*
- Returns: TABLE
- Execute privilege:
  - backend

6.2.23 Function "email"."upd_list"

`Change list admin`

- Parameters:
  - **p_localpart** *email.t_localpart*
  - **p_domain** *dns.t_domain*
  - **p_admin** *email.t_address*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

6.2.24 Function "email"."upd_mailbox"

`Change mailbox password`

- Parameters:
  - **p_localpart** *email.t_localpart*
  - **p_domain** *dns.t_domain*
  - **p_password** *commons.t_password_plaintext*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

6.3 Domains

6.3.1 Domain "email"."t_localpart"

`Local part of an email address, the thing in front of the @`

6.3.2 Domain "email"."t_address"

`Email address, TODO validity checks`

7 Module “jabber”

```
Jabber (XMPP)

This module sends the following signals:
 - jabber/account
```

7.1 Tables

7.1.1 Table “jabber”.“account”

```
Jabber accounts
```

- Primary key:
  - node
  - domain
- Foreign keys:
  1. **reference dns (service)**
     - Columns:
       a) domain →
       b) service →
       c) service_entity_name →
     - Referenced columns:
       a) dns.service.domain
       b) dns.service.service
       c) dns.service.service_entity_name
  2. **Reference subservice entity**
     - Columns:
       a) service_entity_name →
       b) service →
       c) subservice →
     - Referenced columns:
       a) system.subservice_entity.service_entity_name
       b) system.subservice_entity.service
       c) system.subservice_entity.subservice

Columns

domain

```
Domain name
```

- Type: dns.t_domain

service

```
Service
```

- Type: commons.t_key

service_entity_name

```
ent. name
```

- Type: dns.t_domain

subservice

```
Subservice (e.g. account, alias)
```

- Type: commons.t_key

owner

```
for ownage
```

- Type: user.t_user
- References: user.user.owner

backend_status

```
Status of database entry in backend. NULL: nothing pending,
'ins': entry not present on backend client, 'upd': update
pending on backend client, 'del': deletion peding on
backend client.
```

- Type: backend.t_status
- Can be *NULL*
- Default value: `ins'

node

```
part in front of the @ in account name
```

- Type: email.t_localpart

password

```
Unix shadow crypt format
```

- Type: commons.t_password

7.2 Functions

7.2.1 Function "jabber"."del_account"

```
Delete jabber account
```

- Parameters:
  - **p_node** *email.t_localpart*
  - **p_domain** *dns.t_domain*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

7.2.2 Function "jabber"."ins_account"

```
Insert jabber account
```

- Parameters:
  - **p_node** *email.t_localpart*
  - **p_domain** *dns.t_domain*
  - **p_password** *commons.t_password_plaintext*
- Variables defined for body:
  - **v_num_total** *integer*
  - **v_num_domain** *integer*
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

7.2.3 Function "jabber"."sel_account"

```
Select jabber accounts
```

- Parameters: *non*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: TABLE
- Execute privilege:
  - userlogin

7.2.4 Function "jabber"."srv_account"

```
Lists all jabber accounts
```

- Parameters:
  - **p_include_inactive** *boolean*
- Variables defined for body:
  - **v_machine** *dns.t_domain*
- Returns: TABLE
- Execute privilege:
  - backend

7.2.5 Function "jabber"."upd_account"

```
Change jabber account password
```

- Parameters:
  - **p_node** *email.t_localpart*
  - **p_domain** *dns.t_domain*
  - **p_password** *commons.t_password_plaintext*

- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

8 Module “server_access”

```
Server Access

Explicit passwd entries for shell acounts and sftp.

This module sends the following signals:
 - server_access/sftp
 - server_access/ssh
```

8.1 Tables

8.1.1 Table “server_access”.“user”

`unix user`

- Primary key:
  - user
- Foreign keys:
  1. **Reference service entity**
     - Columns:
       a) service_entity_name →
       b) service →
     - Referenced columns:
       a) system.service_entity.service_entity_name
       b) system.service_entity.service
  2. **Reference subservice entity**
     - Columns:
       a) service_entity_name →
       b) service →
       c) subservice →
     - Referenced columns:
       a) system.subservice_entity.service_entity_name
       b) system.subservice_entity.service
       c) system.subservice_entity.subservice

Columns

service_entity_name

`Service entity name`

- Type: dns.t_domain

service

`Service (e.g. email, jabber)`

- Type: commons.t_key

subservice

`Subservice (e.g. account, alias)`

- Type: commons.t_key

backend_status

```
Status of database entry in backend. NULL: nothing pending,
’ins’: entry not present on backend client, ’upd’: update
pending on backend client, ’del’: deletion peding on
backend client.
```

- Type: backend.t_status
- Can be *NULL*
- Default value: `ins'

owner

`for ownage`

- Type: user.t_user
- References: user.user.owner

uid

`Unix user identifier`

- Type: SERIAL

user

`User`

- Type: server_access.t_user

password

`Unix shadow crypt format`

- Type: commons.t_password
- Can be *NULL*

8.2 Functions

8.2.1 Function “server_access”.“del_user”

`delete`

- Parameters:
  - **p_user** *server_access.t_user*
  - **p_service_entity_name** *dns.t_domain*
- Variables defined for body:
  - **v_subservice** *commons.t_key*
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

8.2.2 Function "server_access"."ins_user"

`ins user`

- Parameters:
  - **p_user** *server_access.t_user*
  - **p_service_entity_name** *dns.t_domain*
  - **p_subservice** *commons.t_key*
  - **p_password** *commons.t_password_plaintext*
- Variables defined for body:
  - **v_password** *commons.t_password*
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

8.2.3 Function "server_access"."sel_user"

`sel user`

- Parameters: *non*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: TABLE
- Execute privilege:
  - userlogin

8.2.4 Function "server_access"."srv_user"

`backend server_access.user`

- Parameters:
  - **p_include_inactive** *boolean*
- Variables defined for body:
  - **v_machine** *dns.t_domain*
- Returns: TABLE
- Execute privilege:
  - backend

8.2.5 Function "server_access"."upd_user"

`passwd user`

- Parameters:
  - **p_user** *server_access.t_user*
  - **p_service_entity_name** *dns.t_domain*
  - **p_password** *commons.t_password_plaintext*

- Variables defined for body:
  - **v_password** *commons.t_password* (default: *NULL*)
  - **v_subservice** *commons.t_key*
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

8.3 Domains

8.3.1 Domain "server_access"."t_user"

```
Unix user. This type only allows a subset of those names allowed by POSIX.
```

9 Module “system”

```
Carnivora System

Manages services, service entities and contingents.
```

9.1 Tables

9.1.1 Table “system”.“inherit_contingent”

```
x
```

- Primary key:
  - owner
  - priority

Columns

owner

```
for ownage
```

- Type: user.t_user
- References: user.user.owner

donor

```
Donor
```

- Type: user.t_user

priority

```
Priority, higher values take precedence
```

- Type: int

9.1.2 Table “system”.“service”

```
Services

Just a list of services that exist. Modules do register their services here.
Use system._setup_register_service(, ) to insert into this
table.
```

- Primary key:
  - service

Columns

service

```
Service name
```

- Type: commons.t_key

module

```
Module name, just to keep track who uses this name
```

- Type: commons.t_key

9.1.3 Table “system”.“service_entity”

```
Service Entity

Names under which services are made available. For example (mail.example.org, email)
could be a mail-server system referred to as mail.example.org by carnivora.
Such a system can consist of multiple physical or virtual machines. The corresponding
machines are listed in system.service_entity_machine. A core feature of services is the
definition of 'templates' for dns records which have to be present for every domain
that uses this service. Such 'templates' can be defined in system.service_dns.
Domain names can be enabled for services in dns.service. Service enabled domains
are automatically equipped with the required dns entries accorting to the existing
'templates'.

The service_entity_name might be exposed to users as the address of this service. For
example as SMTP or SSH server etc. The exact interpretation of the service_entity_name
depends on the module and the frontend.
```

- Primary key:
  - service_entity_name
  - service

Columns

service_entity_name

```
Host name
```

- Type: dns.t_domain

service

```
email, ssh, ...
```

- Type: commons.t_key
- References: system.service.service

9.1.4 Table “system”.“service_entity_dns”

```
Service Entity DNS

Resource records that have to be present to use a service. The records
in this table can be understood as 'templates'. The table does not
contain a name (domain) for the records. Rather for every domain that
uses this service, all appropriate records are created for this domain
based on this table. The assignment from domain to services can
be found in dns.service.
```

- Primary key:
  - id
- Foreign keys:

1. **Reference service entity**
   - Columns:
     a) service_entity_name →
     b) service →
   - Referenced columns:
     a) system.service_entity.service_entity_name
     b) system.service_entity.service

Columns

service_entity_name

```
Service entity name
```

- Type: dns.t_domain

service

```
Service (e.g. email, jabber)
```

- Type: commons.t_key

type

```
Type (?) like MX, A, AAAA, ...
```

- Type: dns.t_type

rdata

```
fancy rdata storage
```

- Type: dns.t_rdata

ttl

```
Time to live, NULL indicates default value
```

- Type: dns.t_ttl
- Can be *NULL*

id

```
uuid serial number to identify database elements uniquely
The default value is generated using uuid_generate_v4().
```

- Type: uuid
- Default value: uuid_generate_v4()

domain_prefix

```
Domain prefix
```

- Type: varchar
- Can be *NULL*

9.1.5 Table "system"."service_entity_machine"

```
Service Entity Machine

List of machines that provice a certain service. This information is
used to provide these machines access to the data they need to provide
the service. See also the module 'backend'.
```

- Primary key:
  - machine_name
  - service_entity_name
  - service
- Foreign keys:
  1. **Reference service entity**
     - Columns:
       a) service_entity_name →
       b) service →
     - Referenced columns:
       a) system.service_entity.service_entity_name
       b) system.service_entity.service

Columns

service_entity_name

```
Service entity name
```

- Type: dns.t_domain

service

```
Service (e.g. email, jabber)
```

- Type: commons.t_key

machine_name

```
Assigns machine
```

- Type: dns.t_domain
- References: backend.machine.name

9.1.6 Table "system"."subservice"

```
Subservices
```

- Primary key:
  - service
  - subservice

Columns

service

```
Service
```

- Type: commons.t_key
- References: system.service.service

subservice

```
Subservice (concretization the service)
```

- Type: commons.t_key

9.1.7 Table "system"."subservice_entity"

```
Subservice Entity

Names under which subservices are made available.

See also: Table system.service_entity
```

- Primary key:
  - service_entity_name
  - service
  - subservice
- Foreign keys:
  1. **service ent**
     - Columns:
       a) service_entity_name →
       b) service →
     - Referenced columns:
       a) system.service_entity.service_entity_name
       b) system.service_entity.service
  2. **subservice**
     - Columns:
       a) service →
       b) subservice →
     - Referenced columns:
       a) system.subservice.service
       b) system.subservice.subservice

Columns

service_entity_name

```
Service entity name
```

- Type: dns.t_domain

service

```
Service name
```

- Type: commons.t_key

subservice

```
account, alias, ...
```

- Type: commons.t_key

9.1.8 Table "system"."subservice_entity_contingent"

`Subservice entity contingent`

- Primary key:
  - service
  - subservice
  - service_entity_name
  - owner
- Foreign keys:
  1. **Reference service entity**
     - Columns:
       a) service_entity_name →
       b) service →
     - Referenced columns:
       a) system.service_entity.service_entity_name
       b) system.service_entity.service
  2. **Reference subservice entity**
     - Columns:
       a) service_entity_name →
       b) service →
       c) subservice →
     - Referenced columns:
       a) system.subservice_entity.service_entity_name
       b) system.subservice_entity.service
       c) system.subservice_entity.subservice

Columns

service_entity_name

`Service entity name`

- Type: dns.t_domain

service

`Service (e.g. email, jabber)`

- Type: commons.t_key

subservice

`Subservice (e.g. account, alias)`

- Type: commons.t_key

owner

`for ownage`

- Type: user.t_user
- References: user.user.owner

domain_contingent

`Limit per domain`

- Type: integer

total_contingent

`Limit on the total`

- Type: integer

9.1.9 Table “system”.“subservice_entity_domain_contingent”

`Subservice entity per domain contingent`

- Primary key:
  - service
  - subservice
  - service_entity_name
  - domain
  - owner
- Foreign keys:
  1. **Reference service entity**
     - Columns:
       a) service_entity_name →
       b) service →
     - Referenced columns:
       a) system.service_entity.service_entity_name
       b) system.service_entity.service
  2. **Reference subservice entity**
     - Columns:
       a) service_entity_name →
       b) service →
       c) subservice →
     - Referenced columns:
       a) system.subservice_entity.service_entity_name
       b) system.subservice_entity.service
       c) system.subservice_entity.subservice

Columns

service_entity_name

`Service entity name`

- Type: dns.t_domain

service

`Service (e.g. email, jabber)`

- Type: commons.t_key

subservice

```
Subservice (e.g. account, alias)
```

- Type: commons.t_key

owner

```
for ownage
```

- Type: user.t_user
- References: user.user.owner

domain

```
Specific domain for which the access is granted
```

- Type: dns.t_domain

domain_contingent

```
Limit per domain
```

- Type: integer

9.2 Functions

9.2.1 Function "system"."_contingent_ensure"

```
Throws exceptions if the contingent is exceeded
```

- Parameters:
  - **p_owner** *user.t_user*
  - **p_service** *commons.t_key*
  - **p_subservice** *commons.t_key*
  - **p_domain** *dns.t_domain*
  - **p_current_quantity_total** *integer*
  - **p_current_quantity_domain** *integer*
- Variables defined for body:
  - **v_remaining** *integer*
  - **v_total_contingent** *integer*
  - **v_domain_contingent** *integer*
  - **v_domain_contingent_default** *integer*
  - **v_domain_contingent_specific** *integer*
  - **v_service_entity_name** *dns.t_domain*
  - **v_domain_owner** *user.t_user*
- Returns: void

9.2.2 Function "system"."_contingent_total"

```
Contingent
```

- Parameters:
  - **p_owner** *user.t_user*
  - **p_service** *commons.t_key*
  - **p_service_entity_name** *dns.t_domain*
- Variables defined for body:
  - **v_user** *integer*
  - **v_default** *integer*
- Returns: integer

9.2.3 Function "system"."_effective_contingent"

```
contingent
```

- Parameters: *non*
- Returns: TABLE

9.2.4 Function "system"."_effective_contingent_domain"

```
contingent
```

- Parameters: *non*
- Returns: TABLE

9.2.5 Function "system"."_inherit_contingent_donor"

```
Returns all contingent donors for a given user with their priority.
```

- Parameters:
  - **p_owner** *user.t_user*
- Returns: TABLE

9.2.6 Function "system"."_setup_register_service"

```
Allows modules to register their services during setup.
Returns the total number of service names registered
for this module.
```

- Parameters:
  - **p_module** *commons.t_key*
  - **p_service** *commons.t_key*
- Returns: integer

9.2.7 Function "system"."_setup_register_subservice"

```
Allows modules to register their services during setup.
Returns the total number of service names registered
for this module.
```

- Parameters:
  - **p_service** *commons.t_key*
  - **p_subservice** *commons.t_key*
- Returns: integer

9.2.8 Function "system"."sel_inherit_contingent"

```
Select inherit contingent
```

- Parameters: *non*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: TABLE
- Execute privilege:
  - userlogin

9.2.9 Function "system"."sel_usable_host"

```
Usable hosts
```

- Parameters:
  - **p_service** *commons.t_key*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: TABLE
- Execute privilege:
  - userlogin

10 Module “user”

```
Carnivora Users: Users own things objects in the DB,
and they can login into frontends (edentata)
```

10.1 Tables

10.1.1 Table “user”.“deputy”

```
Deputies for users
```

- Primary key:
  - deputy
  - represented

Columns

deputy

```
Deputy
```

- Type: user.t_user
- References: user.user.owner
  - On delete: CASCADE

represented

```
User for which the deputy can act
```

- Type: user.t_user
- References: user.user.owner
  - On delete: CASCADE

10.1.2 Table “user”.“session”

```
User login sessions
```

- Primary key:
  - id

Columns

id

```
Session id
```

- Type: varchar
- Default value: "user"._session_id()

owner

`for ownage`

- Type: user.t_user
- References: user.user.owner

act_as

`Act as`

- Type: user.t_user

started

`Session started at this time`

- Type: timestamp
- Default value: CURRENT_TIMESTAMP

10.1.3 Table “user”.“user”

`User`

- Primary key:
  - owner

Columns

owner

`User name`

- Type: user.t_user

password

`Unix shadow crypt format`

- Type: commons.t_password
- Can be *NULL*

login

`Login enabled`

- Type: bool

contact_email

`Optional contact email address`

- Type: email.t_address
- Can be *NULL*

10.2 Functions

10.2.1 Function "user"."_get_login"

```
Shows informations for the current user login.
Throws an exception if no login is associated to the
current database connection.
```

- Parameters: *non*
- Returns: TABLE

10.2.2 Function "user"."_session_id"

```
Gives an id for the database connection that is unique over all database connections.
It is used to identify user logins.

Not sure if this stays unique with distributed infrastructure!
```

- Parameters: *non*
- Returns: varchar

10.2.3 Function "user"."ins_deputy"

```
Act as deputy
```

- Parameters:
  - **p_act_as** *user.t_user*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

10.2.4 Function "user"."ins_login"

```
Try to bind database connection to new user session.
Returns valid if sueccessfull, invalid otherwise.
```

- Parameters:
  - **p_owner** *commons.t_key*
  - **p_password** *commons.t_password_plaintext*
- Returns: boolean
- Execute privilege:
  - userlogin

10.2.5 Function "user"."sel_deputy"

`sel deputy`

- Parameters: *non*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: TABLE
- Execute privilege:
  - userlogin

10.2.6 Function "user"."upd_user"

`change user passwd`

- Parameters:
  - **p_password** *commons.t_password_plaintext*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

10.3 Domains

10.3.1 Domain "user"."t_user"

`Username`

10.4 Roles

10.4.1 Role "userlogin"

Do user actions via this group

- Login:

10.4.2 Role "system"

Highly priviledged user

- Login:

11 Module “web”

```
Websites

This module sends the following signals:
 - web/alias
 - web/site
```

11.1 Tables

11.1.1 Table “web”.“alias”

```
Aliases
```

- Primary key:
  - domain
  - site_port
- Foreign keys:
  1. **reference dns (service)**
     - Columns:
       a) domain →
       b) service →
       c) service_entity_name →
     - Referenced columns:
       a) dns.service.domain
       b) dns.service.service
       c) dns.service.service_entity_name
  2. **Reference subservice entity**
     - Columns:
       a) service_entity_name →
       b) service →
       c) subservice →
     - Referenced columns:
       a) system.subservice_entity.service_entity_name
       b) system.subservice_entity.service
       c) system.subservice_entity.subservice
  3. **site**
     - Columns:
       a) site →
       b) service_entity_name →
       c) site_port →
     - Referenced columns:
       a) web.site.domain
       b) web.site.service_entity_name
       c) web.site.port

4. **dns**
   - Columns:
     a) domain →
     b) service →
     c) service_entity_name →
   - Referenced columns:
     a) dns.service.domain
     b) dns.service.service
     c) dns.service.service_entity_name

Columns

domain

`Domain name`

- Type: dns.t_domain

service

`Service`

- Type: commons.t_key

service_entity_name

`ent. name`

- Type: dns.t_domain

subservice

`Subservice (e.g. account, alias)`

- Type: commons.t_key

backend_status

```
Status of database entry in backend. NULL: nothing pending,
'ins': entry not present on backend client, 'upd': update
pending on backend client, 'del': deletion peding on
backend client.
```

- Type: backend.t_status
- Can be *NULL*
- Default value: `ins'

site

`Site`

- Type: dns.t_domain

site_port

`port`

- Type: commons.t_port
- Default value: 80

11.1.2 Table "web"."https"

`stores https information`

- Primary key:
  - identifier
  - domain
  - port
- Foreign keys:
  1. **site**
     - Columns:
       a) domain →
       b) port →
     - Referenced columns:
       a) web.site.domain
       b) web.site.port

Columns

backend_status

```
Status of database entry in backend. NULL: nothing pending,
'ins': entry not present on backend client, 'upd': update
pending on backend client, 'del': deletion peding on
backend client.
```

- Type: backend.t_status
- Can be *NULL*
- Default value: 'ins'

identifier

`PK`

- Type: commons.t_key

domain

`Domain`

- Type: dns.t_domain

port

`Port`

- Type: commons.t_port

x509_request

`Certificate request`

- Type: web.t_cert
- Can be *NULL*

x509_certificate

```
Certificate
```

- Type: web.t_cert
- Can be *NULL*

authority_key_identifier

```
Identifier of the certificate that has signed this cert.
The Authority Key Identifier allows to build the chain of trust.
See .
Hopefully there exists an entry in web.intermediate_cert
or a root certificate with an equal subjectKeyIdentifier.

Is NULL whenever x509_certificate is NULL.
```

- Type: varchar
- Can be *NULL*

11.1.3 Table "web"."intermediate_cert"

```
Intermediate certificates
```

- Primary key:
  - subject_key_identifier

Columns

subject_key_identifier

```
Identifies this certificate
```

- Type: varchar

authority_key_identifier

```
Subject key identifier of the cert that has signed this cert.
NULL is not allowed, since self signed cert do not belong into intermediate
certs.
```

- Type: varchar

x509_certificate

```
Intermediate certificate
```

- Type: web.t_cert

11.1.4 Table "web"."intermediate_chain"

```
xxx
```

- Primary key:
  - domain
  - port
  - identifier
  - subject_key_identifier

- Foreign keys:
    1. **https cert**
        - Columns:
            a) domain →
            b) port →
            c) identifier →
        - Referenced columns:
            a) web.https.domain
            b) web.https.port
            c) web.https.identifier

Columns

domain

`Domain`

- Type: dns.t_domain

port

`Port`

- Type: commons.t_port

identifier

`Identifier`

- Type: commons.t_key

order

`Ordering from leaf to root`

- Type: integer

subject_key_identifier

`SubjectKeyIdentifier`

- Type: varchar
- References: web.intermediate_cert.subject_key_identifier

11.1.5 Table "web"."site"

`Website`

- Primary key:
    - domain
    - port
- Foreign keys:
    1. **reference dns (service)**
        - Columns:
            a) domain →

    b) service →
    c) service_entity_name →
  - Referenced columns:
    a) dns.service.domain
    b) dns.service.service
    c) dns.service.service_entity_name

2. **Reference subservice entity**
  - Columns:
    a) service_entity_name →
    b) service →
    c) subservice →
  - Referenced columns:
    a) system.subservice_entity.service_entity_name
    b) system.subservice_entity.service
    c) system.subservice_entity.subservice
3. **https**
  - Columns:
    a) domain →
    b) port →
    c) https →
  - Referenced columns:
    a) web.https.domain
    b) web.https.port
    c) web.https.identifier
4. **server_access**
  - Columns:
    a) user →
    b) service_entity_name →
  - Referenced columns:
    a) server_access.user.user
    b) server_access.user.service_entity_name

Columns

domain

`Domain name`

- Type: dns.t_domain

service

`Service`

- Type: commons.t_key

service_entity_name

`ent. name`

- Type: dns.t_domain

subservice

`Subservice (e.g. account, alias)`

- Type: commons.t_key

backend_status

```
Status of database entry in backend. NULL: nothing pending,
'ins': entry not present on backend client, 'upd': update
pending on backend client, 'del': deletion peding on
backend client.
```

- Type: backend.t_status
- Can be *NULL*
- Default value: `ins'

option

`Free options in JSON format`

- Type: jsonb
- Default value: `{}'

port

`Port`

- Type: commons.t_port

user

`Server account under which the htdocs reside`

- Type: server_access.t_user

https

`If null, HTTPS is deactivated`

- Type: commons.t_key
- Can be *NULL*

11.2 Functions

11.2.1 Function “web”.“del_alias”

`del`

- Parameters:
  - **p_domain** *dns.t_domain*
  - **p_site_port** *commons.t_port*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

11.2.2 Function "web"."del_intermediate_chain"

`sdf`

- Parameters:
  - **p_domain** *dns.t_domain*
  - **p_port** *commons.t_port*
  - **p_identifier** *commons.t_key*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

11.2.3 Function "web"."del_site"

`del`

- Parameters:
  - **p_domain** *dns.t_domain*
  - **p_port** *commons.t_port*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

11.2.4 Function "web"."fwd_x509_request"

`x509 request`

- Parameters:
  - **p_domain** *dns.t_domain*
  - **p_port** *commons.t_port*
  - **p_identifier** *commons.t_key*
  - **p_x509_request** *web.t_cert*
  - **p_include_inactive** *boolean*
- Variables defined for body:
  - **v_machine** *dns.t_domain*
- Returns: void
- Execute privilege:
  - backend

11.2.5 Function "web"."ins_alias"

`Insert alias`

- Parameters:
  - **p_domain** *dns.t_domain*
  - **p_site** *dns.t_domain*
  - **p_site_port** *commons.t_port*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

11.2.6 Function "web"."ins_https"

`Ins HTTPS`

- Parameters:
  - **p_domain** *dns.t_domain*
  - **p_port** *commons.t_port*
  - **p_identifier** *commons.t_key*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

11.2.7 Function "web"."ins_intermediate_cert"

`Xxx`

- Parameters:
  - **p_subject_key_identifier** *varchar*
  - **p_authority_key_identifier** *varchar*
  - **p_x509_certificate** *web.t_cert*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

11.2.8 Function "web"."ins_intermediate_chain"

```
sdf
```

- Parameters:
  - **p_domain** *dns.t_domain*
  - **p_port** *commons.t_port*
  - **p_identifier** *commons.t_key*
  - **p_order** *integer*
  - **p_subject_key_identifier** *varchar*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

11.2.9 Function "web"."ins_site"

```
Insert site
TODO: check owner and contingent
```

- Parameters:
  - **p_domain** *dns.t_domain*
  - **p_port** *commons.t_port*
  - **p_user** *server_access.t_user*
  - **p_service_entity_name** *dns.t_domain*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

11.2.10 Function "web"."sel_alias"

```
Select alias
```

- Parameters: *non*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: TABLE
- Execute privilege:
  - userlogin

11.2.11 Function "web"."sel_https"

```
sel https
```

- Parameters: *non*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: TABLE
- Execute privilege:
  - userlogin

11.2.12 Function "web"."sel_intermediate_cert"

```
int
```

- Parameters:
  - **p_subject_key_identifier** *varchar*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: TABLE
- Execute privilege:
  - userlogin

11.2.13 Function "web"."sel_intermediate_chain"

```
sel
```

- Parameters: *non*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: TABLE
- Execute privilege:
  - userlogin

11.2.14 Function "web"."sel_site"

```
Owner defined via server_access
```

- Parameters: *non*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: TABLE
- Execute privilege:
  - userlogin

11.2.15 Function "web"."srv_alias"

`backend web.alias`

- Parameters:
  - **p_include_inactive** *boolean*
- Variables defined for body:
  - **v_machine** *dns.t_domain*
- Returns: TABLE
- Execute privilege:
  - backend

11.2.16 Function "web"."srv_https"

`Certs`

- Parameters:
  - **p_include_inactive** *boolean*
- Variables defined for body:
  - **v_machine** *dns.t_domain*
- Returns: TABLE
- Execute privilege:
  - backend

11.2.17 Function "web"."srv_site"

`backend web.site`

- Parameters:
  - **p_include_inactive** *boolean*
- Variables defined for body:
  - **v_machine** *dns.t_domain*
- Returns: TABLE
- Execute privilege:
  - backend

11.2.18 Function "web"."upd_https"

`upd https`

- Parameters:
  - **p_domain** *dns.t_domain*
  - **p_port** *commons.t_port*
  - **p_identifier** *commons.t_key*
  - **p_x509_certificate** *web.t_cert*
  - **p_authority_key_identifier** *varchar*
- Variables defined for body:
  - **v_owner** *user.t_user*

  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

11.2.19 Function "web"."upd_site"

```
set https identif.
```

- Parameters:
  - **p_domain** *dns.t_domain*
  - **p_port** *commons.t_port*
  - **p_identifier** *commons.t_key*
- Variables defined for body:
  - **v_owner** *user.t_user*
  - **v_login** *user.t_user*
- Returns: void
- Execute privilege:
  - userlogin

11.3 Domains

11.3.1 Domain "web"."t_cert"

```
PEM cert
```