Pandore – Database developer guide

This document is a developer guide for the Pandore project database

Clément LE GRUIEC

Salma CHAHMI

Nathan OLBORSKI

Hugo HOUILLON

Table des matières

[1. General informations 5](#_Toc98237131)

[1.1. Server 5](#_Toc98237132)

[1.2. Database architecture 5](#_Toc98237133)

[2. Versions 6](#_Toc98237134)

[2.1. Version 1 6](#_Toc98237135)

[2.2. Version 2 6](#_Toc98237136)

[2.3. Version 3 6](#_Toc98237137)

[2.4. Version 4 6](#_Toc98237138)

[3. Table Configuration 7](#_Toc98237139)

[3.1. Description 7](#_Toc98237140)

[3.2. Stored procedures 8](#_Toc98237141)

[3.2.1. ReadConfiguration 8](#_Toc98237142)

[3.2.2. UpdateConfiguration 9](#_Toc98237143)

[4. Table Service 11](#_Toc98237144)

[4.1. Description 11](#_Toc98237145)

[4.2. Stored procedures 12](#_Toc98237146)

[4.2.1. CreateService 12](#_Toc98237147)

[4.2.2. ReadAllServices 13](#_Toc98237148)

[4.2.3. ReadServiceByID 14](#_Toc98237149)

[4.2.4. ReadServiceByName 15](#_Toc98237150)

[4.2.5. UpdateService 16](#_Toc98237151)

[4.2.6. DeleteServiceByID 17](#_Toc98237152)

[4.2.7. DeleteServiceByName 18](#_Toc98237153)

[5. Table Server 19](#_Toc98237154)

[5.1. Description 19](#_Toc98237155)

[5.2. Stored procedures 20](#_Toc98237156)

[5.2.1. CreateServer 20](#_Toc98237157)

[5.2.2. CreateServerString 21](#_Toc98237158)

[5.2.3. ReadAllServers 22](#_Toc98237159)

[5.2.4. ReadServerByDNSID 23](#_Toc98237160)

[5.2.5. ReadServerByID 24](#_Toc98237161)

[5.2.6. ReadServerByAddress 25](#_Toc98237162)

[5.2.7. UpdateServer 26](#_Toc98237163)

[5.2.8. DeleteServerByID 27](#_Toc98237164)

[5.2.9. DeleteServerByAddress 28](#_Toc98237165)

[6. Table DNS 29](#_Toc98237166)

[6.1. Description 29](#_Toc98237167)

[6.2. Stored procedures 30](#_Toc98237168)

[6.2.1. CreateDNS 30](#_Toc98237169)

[6.2.2. ReadAllDNS 31](#_Toc98237170)

[6.2.3. ReadDNSByID 32](#_Toc98237171)

[6.2.4. ReadDNSByValue 33](#_Toc98237172)

[6.2.5. ReadDNSByServiceID 34](#_Toc98237173)

[6.2.6. ReadIncompleteDNS 35](#_Toc98237174)

[6.2.7. UpdateDNS 36](#_Toc98237175)

[6.2.8. DeleteDNSByID 37](#_Toc98237176)

[6.2.9. DeleteDNSByValue 38](#_Toc98237177)

[7. Table Capture 39](#_Toc98237178)

[7.1. Description 39](#_Toc98237179)

[7.2. Stored procedures 40](#_Toc98237180)

[7.2.1. CreateCapture 40](#_Toc98237181)

[7.2.2. ReadAllCaptures 41](#_Toc98237182)

[7.2.3. ReadCaptureByID 42](#_Toc98237183)

[7.2.4. ReadSavedCaptures 43](#_Toc98237184)

[7.2.5. ReadCaptureServicesStats 44](#_Toc98237185)

[7.2.6. ReadRunningCapture 45](#_Toc98237186)

[7.2.7. ReadCaptureTotalTrafic 46](#_Toc98237187)

[7.2.8. UpdateCapture 47](#_Toc98237188)

[7.2.9. UpdateCaptureEndTime 48](#_Toc98237189)

[7.2.10. DeleteCaptureByID 49](#_Toc98237190)

[8. Table Capture\_Request 50](#_Toc98237191)

[8.1. Description 50](#_Toc98237192)

[8.2. Stored procedures 51](#_Toc98237193)

[8.2.1. CreateRequest 51](#_Toc98237194)

[8.2.2. CreateRequestString 52](#_Toc98237195)

[8.2.3. ReadAllRequests 53](#_Toc98237196)

[8.2.4. ReadRequestsByCaptureID 54](#_Toc98237197)

[8.2.5. ReadRequestByID 55](#_Toc98237198)

[8.2.6. UpdateRequest 56](#_Toc98237199)

[8.2.7. DeleteRequestByID 57](#_Toc98237200)

[8.2.8. DeleteRequestByCaptureID 58](#_Toc98237201)

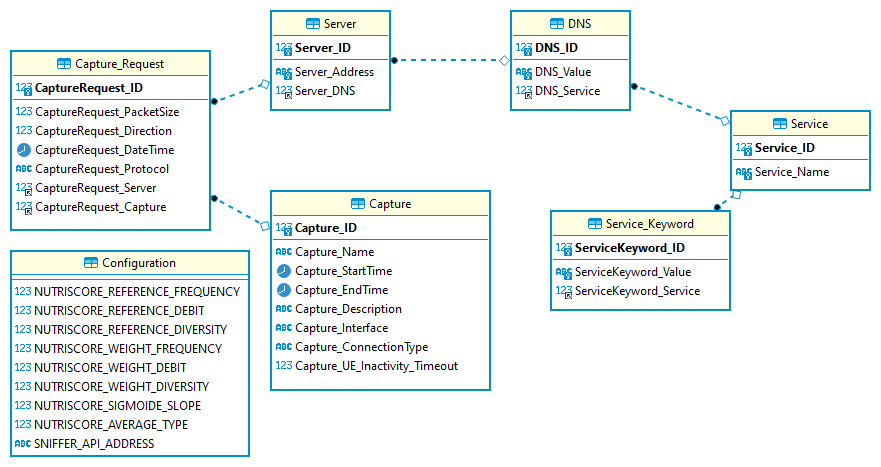
# General informations

## Server

The database server used for the Pandore project is ***MariaDB Server 10.6.4***

## Database architecture

The diagram below is a class diagram of the database architecture:



# Versions

## Version 1

* First database architecture implementation

## Version 2

* Fields *CaptureRequest\_DNS* and *CaptureRequest\_Server* now allow *NULL* values
* Stored procedure *CreateRequestString* edited. It allows now to call it with *NULL* values for *DNS* or *Server* fields
* Stored procedure *ReadAllRequests* edited. New procedure parameter *Details*. Allows now to get requests with DNS and Server details
* Stored procedure *ReadRequestsByCaptureID* edited. New procedure parameter *Details*. Allows now to get requests with DNS and Server details
* Stored procedure *ReadRequestByID* edited. New procedure parameter *Details*. Allows now to get requests with DNS and Server details

## Version 3

* Field *CaptureRequest\_DNS* deleted
* Remove field *DNS\_Server* (DNS doesn’t refer to service table anymore)
* Add field *DNS\_Service*. A DNS refers now to a server
* Multiple stored procedures edited
* Multiple stored procedures deleted
* New stored procedure *CreateServerString* which allows to add a server and a DNS from there names

## Version 4

Too much modification to list all modifications

# Table Configuration

## Description

This table is used to store the web server configuration

It consists of 9 attributes :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Type | Allow NULL | Constraint | Description |
| NUTRISCORE\_REFERENCE\_FREQUENCY | INT |  |  | Reference value for frequency for score calculation |
| NUTRISCORE\_REFERENCE\_DEBIT | FLOAT |  |  | Reference value for debit for score calculation (MB/s) |
| NUTRISCORE\_REFERENCE\_DIVERSITY | INT |  |  | Reference value for diversity for score calculation |
| NUTRISCORE\_WEIGHT\_FREQUENCY | INT |  |  | Weight of frequency parameter in the global score calculation |
| NUTRISCORE\_WEIGHT\_DEBIT | INT |  |  | Weight of debit parameter in the global score calculation |
| NUTRISCORE\_WEIGHT\_DIVERSITY | INT |  |  | Weight of diversity parameter in the global score calculation |
| NUTRISCORE\_SIGMOIDE\_SLOPE | FLOAT |  |  | Slope of the sigmoid for the score calculation |
| NUTRISCORE\_AVERAGE\_TYPE | INT |  |  | Score average type (0 = arithmetic, 1 = harmonic) |
| SNIFFER\_API\_ADDRESS | VARCHAR(1000) |  |  | Address of the probe (sniffer) API |

## Stored procedures

### ReadConfiguration

#### Description

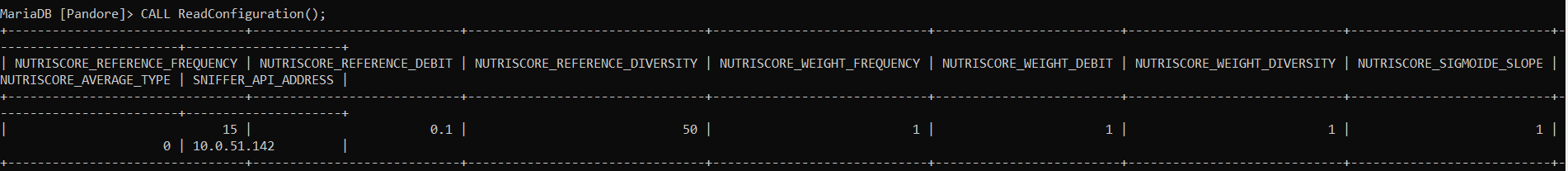
This procedure allows to get web server configuration

#### Prototype



#### Returns

Web server configuration



#### Example

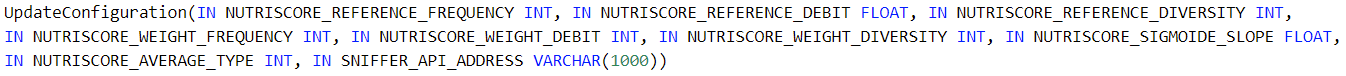


### UpdateConfiguration

#### Description

This procedure allows to update the current web server configuration

#### Prototype



#### Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Type | Allow NULL | Constraint | Description |
| NUTRISCORE\_REFERENCE\_FREQUENCY | INT |  |  | Reference value for frequency for score calculation |
| NUTRISCORE\_REFERENCE\_DEBIT | FLOAT |  |  | Reference value for debit for score calculation (MB/s) |
| NUTRISCORE\_REFERENCE\_DIVERSITY | INT |  |  | Reference value for diversity for score calculation |
| NUTRISCORE\_WEIGHT\_FREQUENCY | INT |  |  | Weight of frequency parameter in the global score calculation |
| NUTRISCORE\_WEIGHT\_DEBIT | INT |  |  | Weight of debit parameter in the global score calculation |
| NUTRISCORE\_WEIGHT\_DIVERSITY | INT |  |  | Weight of diversity parameter in the global score calculation |
| NUTRISCORE\_SIGMOIDE\_SLOPE | FLOAT |  |  | Slope of the sigmoid for the score calculation |
| NUTRISCORE\_AVERAGE\_TYPE | INT |  |  | Score average type (0 = arithmetic, 1 = harmonic) |
| SNIFFER\_API\_ADDRESS | VARCHAR(1000) |  |  | Address of the probe (sniffer) API |

#### Example



# Table Service

## Description

This table is used to identify the name of different services (ex : Facebook, Youtube, …).

It consists of 2 attributes :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Type | Allow NULL | Constraint | Description |
| Service\_ID | INT |  | PK | Service ID |
| Service\_Name | VARCHAR(255) |  | Unique | Service Name |

## Stored procedures

### CreateService

#### Description

This procedure allows to create a new service into the database

#### Prototype



#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| Name | VARCHAR(255) |  | Service name |

#### Example



### ReadAllServices

#### Description

This procedure allows to list all services in the database

#### Prototype



#### Returns

Service list ordered alphabetically

Une image contenant texte

Description générée automatiquement

#### Example



### ReadServiceByID

#### Description

This procedure allows to find a service in the database by its ID

#### Prototype



#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| ID | INT |  | Service ID to find |

#### Returns

The service found with the given ID

Une image contenant texte

Description générée automatiquement

#### Example



### ReadServiceByName

#### Description

This procedure allows to find a service by its name

#### Prototype

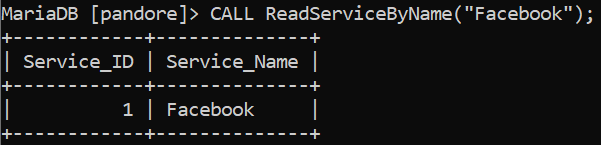


#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| Name | VARCHAR(255) |  | Service name to find |

#### Returns

Services found with the given service name



#### Example



### UpdateService

#### Description

This procedure allows to update an existing service

#### Prototype



#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| ID | INT |  | Service ID to update |
| Name | VARCHAR(255) |  | New service name |

#### Example



### DeleteServiceByID

#### Description

This procedure allows to delete a service by its ID

#### Prototype



#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| ID | INT |  | ID of the service to delete |

#### Example



### DeleteServiceByName

#### Description

This procedure allows to delete a service by its name

#### Prototype



#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| Name | VARCHAR(255) |  | Name of the service to delete |

#### Example



# Table Server

## Description

This table is used to identify different server addresses and associate them to services

It consists of 3 attributes :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Type | Allow NULL | Constraint | Description |
| Server\_ID | INT |  | Primary Key | Service ID |
| Server\_Address | VARCHAR(1000) |  | Unique | Service Name |
| Server\_DNS | INT | X | Foreign Key | ID of the associated domain name |

## Stored procedures

### CreateServer

#### Description

This procedure allows to create a new server in the database

#### Prototype



#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| Address | VARCHAR(1000) |  | Address of the server |
| DNS | INT | x | ID of the associated domain name |

#### Example



### CreateServerString

#### Description

This procedure allows to create a new server in the database and a new domain name if its value doesn’t already exist.

#### Prototype



#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| Address | VARCHAR(1000) |  | Address of the server |
| DNS | VARCHAR(1000) | x | Value of the associated DNS |

#### Example



### ReadAllServers

#### Description

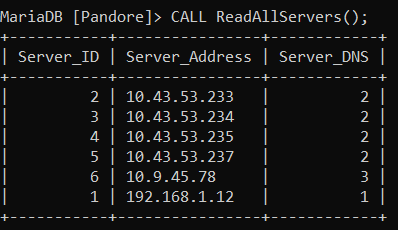
This procedure allows to list all servers in the database

#### Prototype



#### Returns

Server list ordered alphabetically



#### Example



### ReadServerByDNSID

#### Description

This procedure allows to list all servers for a given domain name

#### Prototype

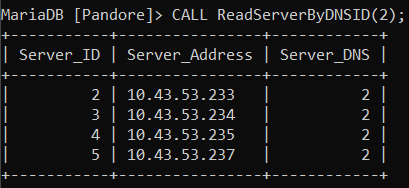


#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| DNS | INT |  | ID of the domain name |

#### Returns

List of servers associated to given service



#### Example



### ReadServerByID

#### Description

This procedure allows to find a server by its ID

#### Prototype

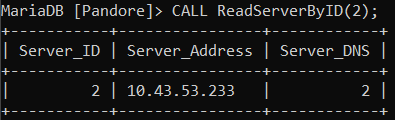


#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| ID | INT |  | ID of the server |

#### Returns

Server found with given ID



#### Example



### ReadServerByAddress

#### Description

This procedure allows to find a server by its address

#### Prototype

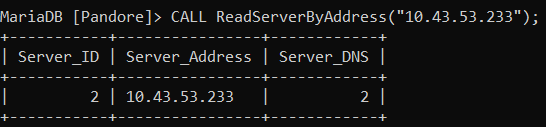


#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| Address | VARCHAR(1000) |  | Address of the server |

#### Returns

Server found with given address



#### Example



### UpdateServer

#### Description

This procedure allows to update a server

#### Prototype



#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| ID | INT |  | ID of the server to update |
| Address | VARCHAR(1000) |  | New server address |
| DNS | INT | x | New domain name ID |

#### Example



### DeleteServerByID

#### Description

This procedure allows to delete a server by its ID

#### Prototype



#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| ID | INT |  | ID of the server to delete |

#### Example



### DeleteServerByAddress

#### Description

This procedure allows to delete a server by its address

#### Prototype



#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| Address | VARCHAR(1000) |  | Address of the server to delete |

#### Example



# Table DNS

## Description

This table is used to identify different DNS and associate them to services

It consists of 3 attributes :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Type | Allow NULL | Constraint | Description |
| DNS\_ID | INT |  | Primary Key | DNS ID |
| DNS\_Value | VARCHAR(1000) |  | Unique | DNS Name |
| DNS\_Service | INT | x | Foreign Key | ID of the associated service |

## Stored procedures

### CreateDNS

#### Description

This procedure allows to create a new DNS in the database

#### Prototype



#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| Value | VARCHAR(1000) |  | Value of the domain name |
| Service | INT | x | ID of the associated service |

#### Example



### ReadAllDNS

#### Description

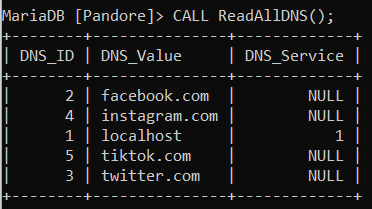
This procedure allows to list all DNS in the database

#### Prototype



#### Returns

DNS list ordered alphabetically



#### Example



### ReadDNSByID

#### Description

This procedure allows to find a DNS by its ID

#### Prototype

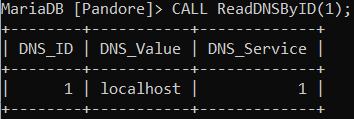


#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| ID | INT |  | ID of the DNS |

#### Returns

DNS found with given ID



#### Example



### ReadDNSByValue

#### Description

This procedure allows to find a DNS by its value

#### Prototype

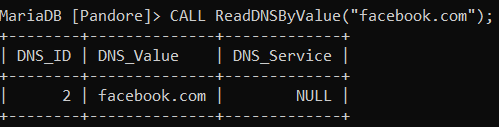


#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| Address | VARCHAR(1000) |  | Value of the DNS |

#### Returns

DNS found with given value



#### Example



### ReadDNSByServiceID

#### Description

This procedure allows to list all domain names for a given service

#### Prototype

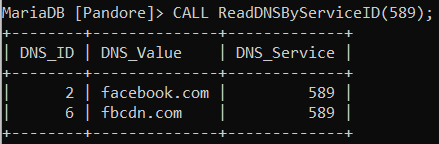


#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| Service | INT |  | Value of the service |

#### Returns

DNS found with given service ID



#### Example



### ReadIncompleteDNS

#### Description

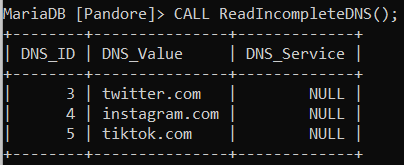
This procedure allows to list all domain names which are not associated to any service

#### Prototype



#### Returns

DNS found who are not associated to any service



#### Example



### UpdateDNS

#### Description

This procedure allows to update a DNS

#### Prototype



#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| ID | INT |  | ID of the DNS to update |
| Value | VARCHAR(1000) |  | New DNS address |
| Service | INT | x | New DNS service |

#### Example



### DeleteDNSByID

#### Description

This procedure allows to delete a DNS by its ID

#### Prototype



#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| ID | INT |  | ID of the DNS to delete |

#### Example



### DeleteDNSByValue

#### Description

This procedure allows to delete a DNS by its value

#### Prototype



#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| Value | VARCHAR(1000) |  | Value of the DNS to delete |

#### Example



# Table Capture

## Description

This table is used to identify captures done on a device

It consists of 7 attributes :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Type | Allow NULL | Constraint | Description |
| Capture\_ID | INT |  | Primary Key | Capture ID |
| Capture\_Name | VARCHAR(255) | x |  | Capture Name |
| Capture\_StartTime | DATETIME |  |  | Capture start time.  Format :  yyyy-MM-dd HH:mm:ss |
| Capture\_EndTime | DATETIME | x |  | Capture end time.  Format :  yyyy-MM-dd HH:mm:ss |
| Capture\_Description | VARCHAR(1000) | x |  | Capture description |
| Capture\_Interface | VARCHAR(255) | x |  | Capture interface (eth0, eth1, etc…) |
| Capture\_ConnectionType | VARCHAR(255) | x |  | Connection type description |
| Capture\_UE\_Inactivity\_Timeout | INT |  |  | Time in seconds after which the UE is disconnected from the network |

## Stored procedures

### CreateCapture

#### Description

This procedure allows to create a new capture in the database and returns the ID of the created capture

#### Prototype

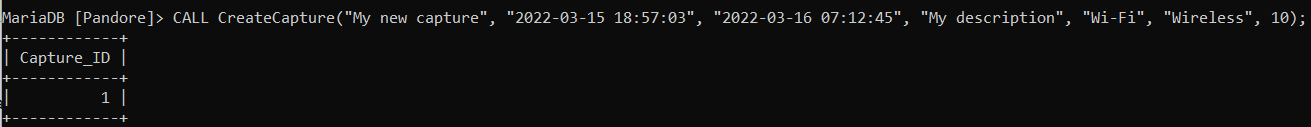


#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| Name | VARCHAR(255) | x | Capture Name |
| StartTime | DATETIME |  | Capture start time. Format : yyyy-MM-dd HH:mm:ss |
| EndTime | DATETIME | x | Capture end time. Format : yyyy-MM-dd HH:mm:ss |
| Description | VARCHAR(1000) | x | Capture description |
| Interface | VARCHAR(255) | x | Capture interface (eth0, eth1, etc…) |
| ConnectionType | VARCHAR(255) | x | Connection type description |
| InactivityTimeout | INT |  | Time in second after which UE is disconnected from the network |

#### Returns

Returns the ID of the capture created



#### Example



### ReadAllCaptures

#### Description

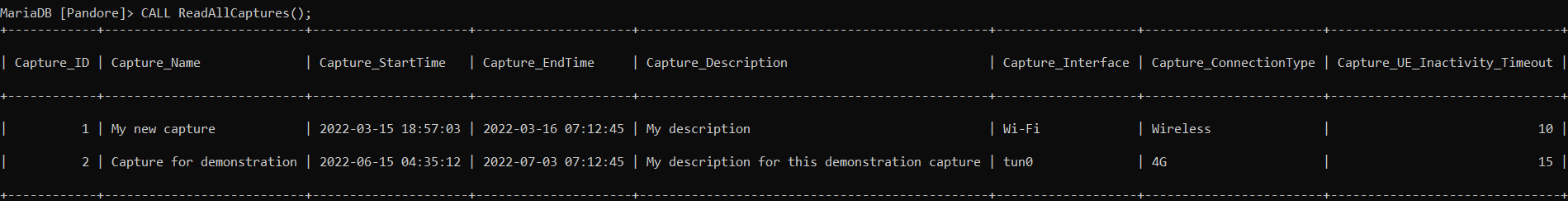
This procedure allows to list all captures

#### Prototype



#### Returns

List of all captures



#### Example



### ReadCaptureByID

#### Description

This procedure allows to find a capture by its ID

#### Prototype

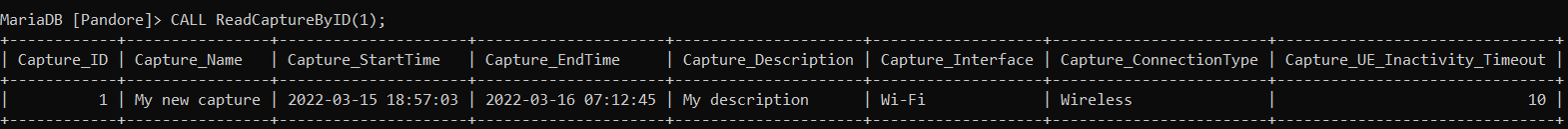


#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| ID | INT |  | Capture ID |

#### Returns

The capture found with given ID



#### Example



### ReadSavedCaptures

#### Description

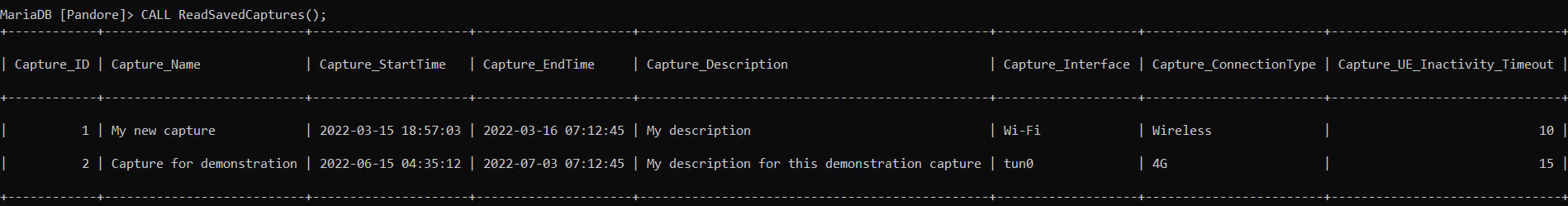
This procedure allows to list all captures who are finished (where capture end time is not null)

#### Prototype



#### Returns

All capture where Capture\_EndTime is not null



#### Example



### ReadCaptureServicesStats

#### Description

This procedure allows to find the list of services name contacted in a capture and the total download and upload traffic captures for each service. If no service name found, returns domain name instead. If no domain name found, returns server address instead.

#### Prototype

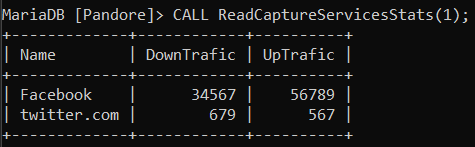


#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| ID | INT |  | Capture ID |

#### Returns

List of all services contacted in a capture and their associated amount of downloaded and uploaded traffic



#### Example



### ReadRunningCapture

#### Description

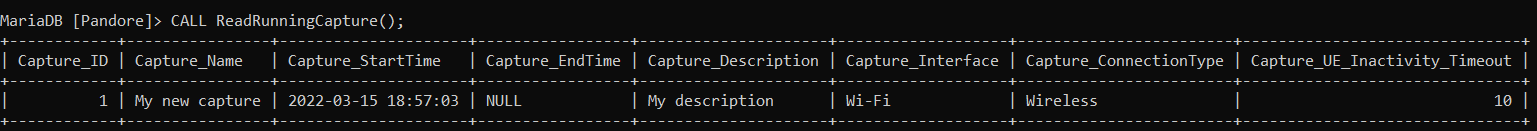
This procedure allows to list all captures who are not finished (where capture end time is null)

#### Prototype



#### Returns

All capture where Capture\_EndTime is null



#### Example



### ReadCaptureTotalTrafic

#### Description

This procedure allows to get the total amount of data downloaded and uploaded in bytes

#### Prototype

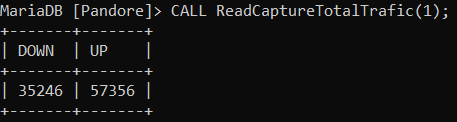


#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| ID | INT |  | Capture ID |

#### Returns

The total amount of data downloaded and uploaded in bytes for a given capture ID



#### Example



### UpdateCapture

#### Description

This procedure allows to update a capture by its ID

#### Prototype



#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| ID | INT |  | ID of the capture to update |
| Name | VARCHAR(255) | x | New capture name |
| StartTime | DATETIME |  | New capture start time.  Format : yyyy-MM-dd HH:mm:ss |
| EndTime | DATETIME | x | New capture end time.  Format : yyyy-MM-dd HH:mm:ss |
| Description | VARCHAR(1000) | x | New capture description |
| Interface | VARCHAR(255) | x | New capture interface (eth0, eth1, etc…) |
| ConnectionType | VARCHAR(255) | x | New connection type description |
| InactivityTimeout | INT |  | New UE inactivity timeout (in second) |

#### Example



### UpdateCaptureEndTime

#### Description

This procedure allows to update a capture end time by its ID

#### Prototype



#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| ID | INT |  | ID of the capture to update |
| EndTime | DATETIME | x | New capture end time.  Format : yyyy-MM-dd HH:mm:ss |

#### Example



### DeleteCaptureByID

#### Description

This procedure allows to delete a capture by its ID

#### Prototype



#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| ID | INT |  | ID of the capture to delete |

#### Example



# Table Capture\_Request

## Description

This table is used to identify requests for a specific capture

It consists of 7 attributes :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Type | Allow NULL | Constraint | Description |
| CaptureRequest\_ID | INT |  | Primary Key | Capture request ID |
| CaptureRequest\_PacketSize | FLOAT |  |  | Size of the packet |
| CaptureRequest\_Direction | TINYINT |  |  | Packet direction (0 = download, 1 = upload) |
| CaptureRequest\_DateTime | TIMESTAMP |  |  | Request date time.  Format :  yyyy-MM-dd HH:mm:ss |
| CaptureRequest\_Protocol | VARCHAR(255) |  |  | Request protocol description |
| CaptureRequest\_Server | INT |  | Foreign Key | ID of the server |
| CaptureRequest\_Capture | INT |  | Foreign Key | ID of the capture |

## Stored procedures

### CreateRequest

#### Description

This procedure allows to create a new capture request in the database

#### Prototype



#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| PacketSize | FLOAT |  | Size of the packet |
| Direction | TINYINT |  | Packet direction (0 = download, 1 = upload) |
| Protocol | VARCHAR(255) |  | Request protocol description |
| Server | INT |  | ID of the server |
| Capture | INT |  | ID of the capture |

#### Example



### CreateRequestString

#### Description

This procedure allows to create a new capture request in the database without knowing the id of the DNS and the server but only their values.

If DNS is not null and doesn’t exist in the database, new DNS is added in DNS table.

If Server is not null and doesn’t exist in the database, new server is added in Server table.

If DNS is null or empty, request DNS will also be null and nothing will be added in DNS table.

If Server is null or empty, request Server will also be null and nothing will be added in Server table.

#### Prototype



#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| PacketSize | FLOAT |  | Size of the packet |
| Direction | TINYINT |  | Packet direction (0 = download, 1 = upload) |
| Protocol | VARCHAR(255) |  | Request protocol description |
| Server | VARCHAR(1000) | x | Server address |
| DNS | VARCHAR(1000) | x | DNS value |
| Capture | INT |  | ID of the capture |

#### Example



### ReadAllRequests

#### Description

This procedure allows to list all capture requests

#### Prototype

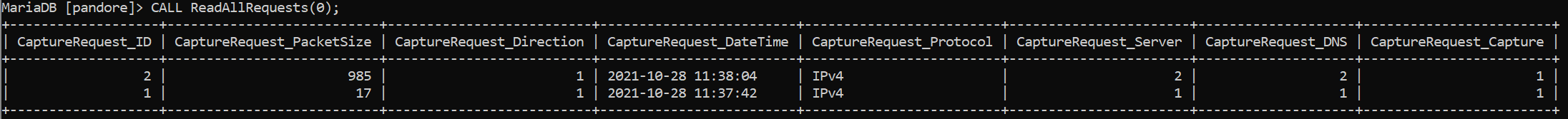


#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| Details | TINYINT(1) |  | 0 = returns list of requests  1 = returns list of requests with join with DNS and server tables |

#### Returns

List of all capture requests (details = 0)



List of all capture requests with DNS and server details (details = 1)



#### Example



### ReadRequestsByCaptureID

#### Description

This procedure allows to list all requests for a given capture

#### Prototype

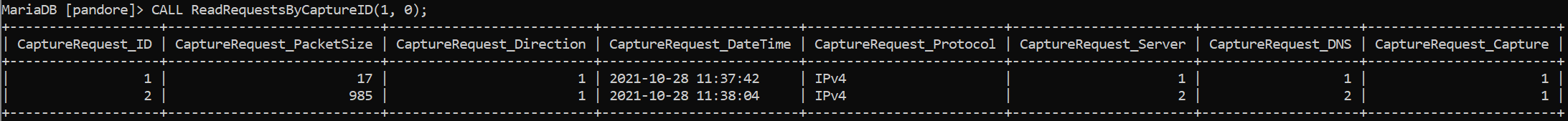


#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| Capture | INT |  | ID of the capture |
| Details | TINYINT(1) |  | 0 = returns list of requests  1 = returns list of requests with join with DNS and server tables |

#### Returns

List of all capture requests for the given capture ID (details = 0)



List of all capture requests for the given capture ID with DNS and server details (details = 1)



#### Example



### ReadRequestByID

#### Description

This procedure allows to find a request by its ID

#### Prototype

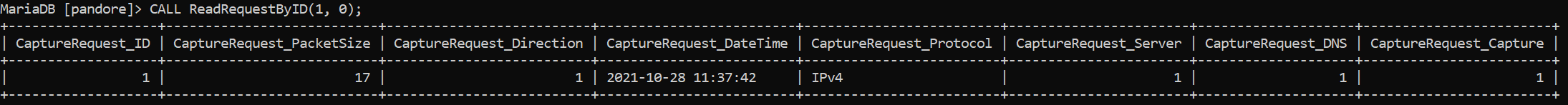


#### Parameters

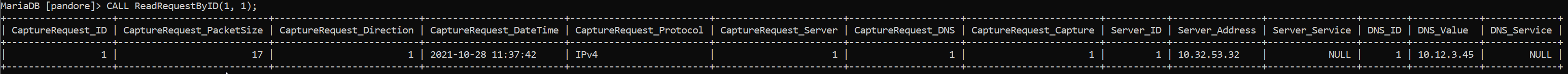
|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| ID | INT |  | ID of the request to update |
| Details | TINYINT(1) |  | 0 = returns list of requests  1 = returns list of requests with join with DNS and server tables |

#### Returns

The request found for the given request ID (details = 0)



The request found for the given request ID with DNS and server details (details = 1)



#### Example



### UpdateRequest

#### Description

This procedure allows to update an existing capture request

#### Prototype



#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| ID | INT |  | ID of the request to update |
| PacketSize | FLOAT |  | New packet size |
| Direction | TINYINT |  | New packet direction (0 = download, 1 = upload) |
| DateTime | TIMESTAMP |  | New request date time.  Format :  yyyy-MM-dd HH:mm:ss |
| Protocol | VARCHAR(255) |  | New protocol description |
| Server | INT |  | ID of the new server |
| DNS | INT |  | ID of the new DNS |
| Capture | INT |  | ID of the new capture |

#### Example



### DeleteRequestByID

#### Description

This procedure allows to delete a capture request by its ID

#### Prototype



#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| ID | INT |  | ID of the capture request to delete |

#### Example



### DeleteRequestByCaptureID

#### Description

This procedure allows to delete all requests for a given capture

#### Prototype



#### Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Allow NULL | Description |
| Capture | INT |  | ID of the capture |

#### Example

