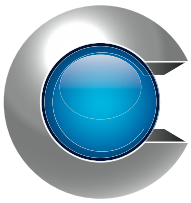


Manual

Cryptify Rendezvous Server

Cryptify AB CAB-12:047, Rev AR 2026-06-02

Contents

1 Scope

This document describes how to install, configure and maintain the Cryptify Rendezvous Server (CRS). Target audience is IT personnel responsible for the operations of the CRS. It is expected that the reader have basic knowledge in the following areas

- TCP/IP
- Ubuntu or RHEL Linux

2 Setup

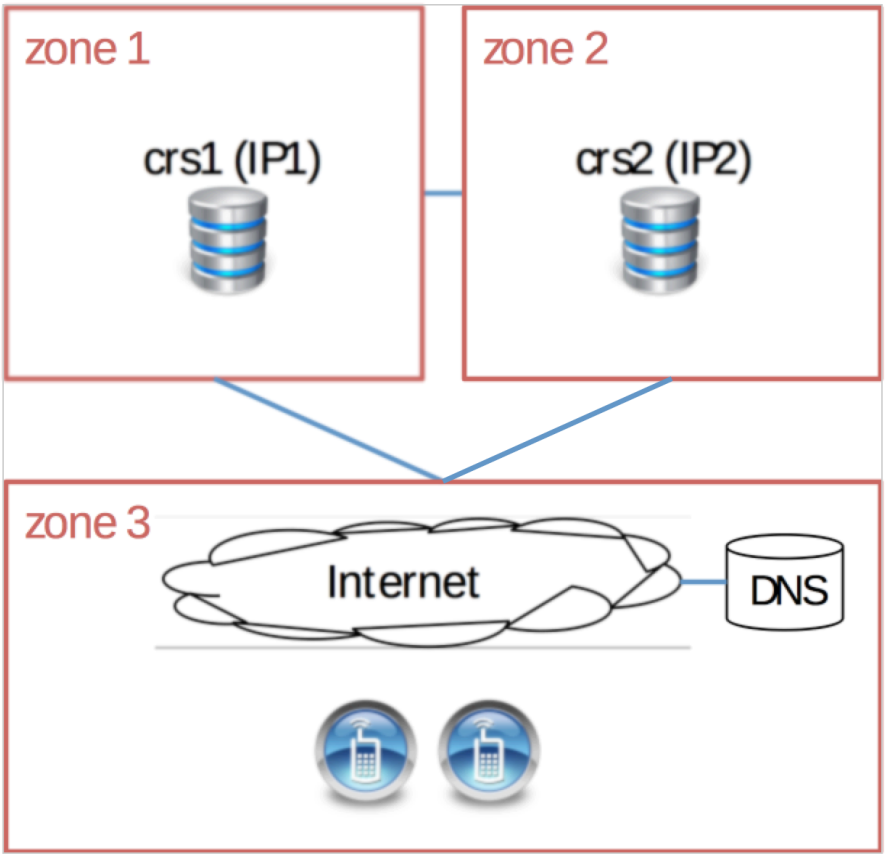


Figure 1: The picture shows an overview of the CRS setup with two servers in a pool.

IP1 and IP2 must be public IP addresses reachable from Internet on the following ports:

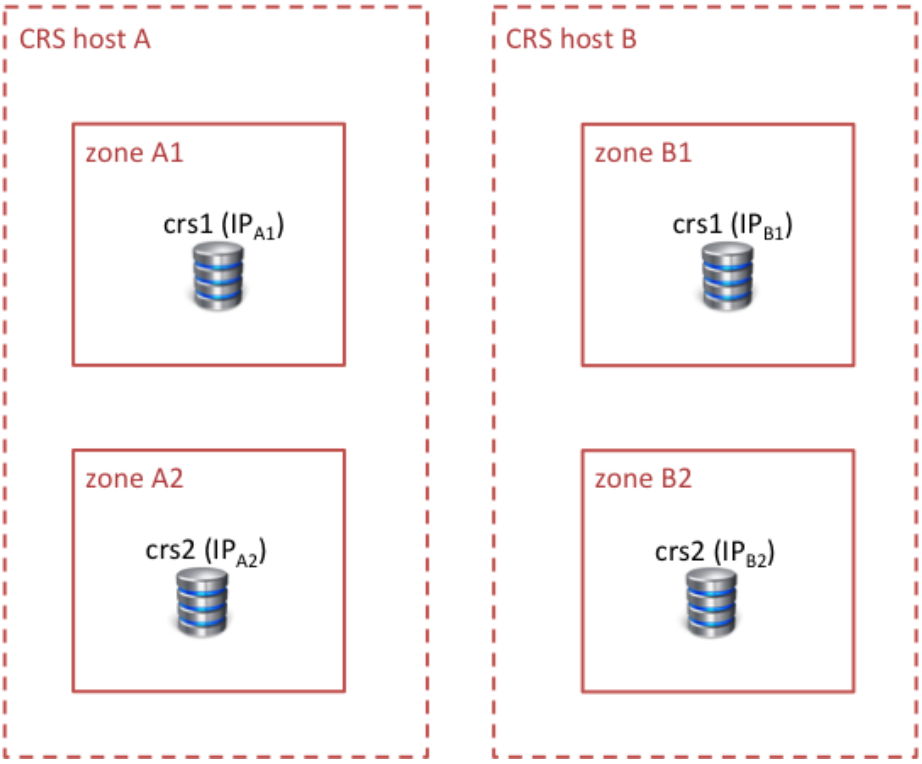
The DNS shall return a list containing both IP1 and IP2 when the clients query the CRS name, where the CRS name shall be the DNS name entered in the CMS.

From	To	IP Source	IP dest	Protocol	Port	Comment
zone 3	zone 1	ANY	IP1	TCP	5223	Client signaling/TLS
zone 3	zone 1	ANY	IP1	UDP	146	Client media / SRTP
zone 3	zone 2	ANY	IP2	TCP	5223	Client signaling/TLS
zone 3	zone 2	ANY	IP2	UDP	146	Client media / SRTP
zone 1	zone 2	IP1	IP2	TCP	8090	CRS failover/TLS
zone 2	zone 1	IP2	IP1	TCP	8090	CRS failover/TLS
zone 1	zone 2	IP1	IP2	TCP	8091	CRS synchronizing/TLS
zone 2	zone 1	IP2	IP1	TCP	8091	CRS synchronizing/TLS
zone 1	zone 3	IP1	cps.cryptify.net	TCP	5030	Cryptify Push Server/TLS
zone 2	zone 3	IP2	cps.cryptify.net	TCP	5030	Cryptify Push Server/TLS

Table 1: Routing, single host.

2.1 Inter-Working CRS hosts

In case the CRSs shall be peered with CRSs on another host please make sure routing is allowed in accordance with Table 2 below.



2.2 Firewall service

The CRS package for RHEL includes a firewall service configuration which can be enabled by issuing the following commands:

```
$ sudo firewall-cmd --permanent --add-service=crs
```

From	To	IP Source	IP dest	Protocol	Port	Comment
zone A1	zone B1	IP_{A1}	IP_{B1}	TCP	8080	Signaling/TLS
zone A1	zone B1	IP_{A1}	IP_{B1}	UDP	146	Media/SRTP
zone A1	zone B2	IP_{A1}	IP_{B2}	TCP	8080	Signaling/TLS
zone A1	zone B2	IP_{A1}	IP_{B2}	UDP	146	Media/SRTP
zone A2	zone B1	IP_{A2}	IP_{B1}	TCP	8080	Signaling/TLS
zone A2	zone B1	IP_{A2}	IP_{B1}	UDP	146	Media/SRTP
zone A2	zone B2	IP_{A2}	IP_{B2}	TCP	8080	Signaling/TLS
zone A2	zone B2	IP_{A2}	IP_{B2}	UDP	146	Media/SRTP
zone B1	zone A1	IP_{B1}	IP_{A1}	TCP	8080	Signaling/TLS
zone B1	zone A1	IP_{B1}	IP_{A1}	UDP	146	Media/SRTP
zone B1	zone A2	IP_{B1}	IP_{A2}	TCP	8080	Signaling/TLS
zone B1	zone A2	IP_{B1}	IP_{A2}	UDP	146	Media/SRTP
zone B2	zone A1	IP_{B2}	IP_{A1}	TCP	8080	Signaling/TLS
zone B2	zone A1	IP_{B2}	IP_{A1}	UDP	146	Media/SRTP
zone B2	zone A2	IP_{B2}	IP_{A2}	TCP	8080	Signaling/TLS
zone B2	zone A2	IP_{B2}	IP_{A2}	UDP	146	Media/SRTP

Table 2: Routing, between CRS hosts

```
$ sudo sudo firewall-cmd --add-service=crs
```

3 Pre-requisites

The CRS hosting environment is according to the setup described above.

Before installing the CRS you will need two servers dedicated for running CRS services.

These servers shall be installed with either 64-bit Ubuntu Server 22.04 LTS, Ubuntu Server 24.04 LTS, Red Hat Enterprise Linux (RHEL) 8 or RHEL 9 with the most recent patch level.

3.1 Suggested dimensioning

By way of example, dimensioning figures for systems of different size in a typical traffic model are also given. The reader is encouraged to adapt the traffic model to their own circumstances, please visit section 8 for a detailed explanation.

USERS	STORAGE	RAM	BANDWIDTH EXCLUDING VIDEO CONFERENCING
<500	20 GB	8 GB	10 Mbit/s
<2 500	100 GB	16 GB	100 Mbit/s
<5 000	200 GB	32 GB	1 Gbit/s
<20 000	800 GB	104 GB	1 Gbit/s
<50 000	2000 GB	256 GB	1 Gbit/s

Table 3: Baseline dimensioning examples

PARTICIPANTS VIDEO CONFERENCING	INGRESS MBIT/S	INGRESS PACKET/S	EGRESS MBIT/S	EGRESS PACKET/S
10	6	1 440	38	8 200
25	14	3 600	95	20 500
50	28	7 200	191	41 000
100	55	14 400	381	82 000
200	110	28 800	762	164 000
300	165	43 200	1 143	246 000
400	220	57 600	1 524	328 000
500	275	72 000	1 905	410 000

Table 4: Video conference network requirements

4 Procedures

4.1 Initial installation and configuration

For both crs1 and crs2 do the following:

4.1.1 Preparations

Copy the latest version of the CRS installation package to the home directory `/root`.

4.1.2 Installation

Ensure the authenticity of a software before installing. To maintain system integrity and prevent installation of tampered or malicious software, all updates must be verified as originating from Cryptify AB before installation.

Install the package using dpkg or rpm as root.

Ubuntu command:

`$ sudo dpkg -i crs_`*`VERSION`*`-1_amd64.deb`

RHEL command:

`$ sudo rpm -U crs_`*`VERSION`*`-1_rhel_x86_64.rpm`

When installing the CRS, a new user – crsuser – is added to the system and used to run all services.

Next, there are two configuration files that needs to be adapted to the local settings; /opt/crs/crs.conf and /opt/crs/crs-local.conf

4.1.3 crs.conf

The crs.conf file must be identical on all CRS instances in the pool.

A shared secret is required for secure communication between CRSs in the pool. In the GLOBAL section, enter a shared secret accordingly:

`key <shared secret> ["128bit"|"256bit"]`

The bit argument determines the key length and is optional. A key length of 256 bits is assumed if no key length is specified.

You could use the CLI command to generate a shared secret (please see command description below).

```
$ crs-cli create sharedSecret
```

To generate a shared secret with n bytes, add the optional argument "-Bn". For instance "-B32" will generate a shared secret with 32 bytes of random data.

```
$ crs-cli create sharedSecret -B32
```

IP addresses (in the POOL section)

Please enter the IP1 and IP2 for crs1 and crs2 accordingly. Please note that IP1 and IP2 must be public addresses.

```
ip crs1 <IP1>
ip crs2 <IP2>
```

Hosts

Please make sure that this section declares all the hosts in the crs pool.

```
host  crs1
host  crs2
```

4.1.4 crs-local.conf

The crs-local.conf file shall only contain local configurations specific to the crs instance. This includes instance name and database backup configuration of the local server.

Please edit the file /opt/crs/crs-local.conf

```
crsname crs1 (must match the name in crs.conf)
```

db-backup [none, replicate-from-peer, read-from]

> **none (default):** Do not serve as database backup for peer crs
>
> **replicate-from-peer:** Serve as database backup for peer crs
>
> **read-from:** The crs reads from the backup database. For use when the peer crs is permanently down

4.1.5 crs-relay.conf

The crs-relay.conf file allows one to limit the number of relay resources of a specific type which can exist simultaneously on a single server. To configure this please edit the file /opt/crs/crs-relay.conf and then restart the relay service to make any changes take affect. The crs-relay.conf has the following configurations:

> MAX_RTP_RELAYS=<limit>
>
> MAX_RTP_PROXY_RELAYS=<limit>
>
> MAX_OPAQUE_RELAYS=<limit>
>
> MAX_OPAQUE_PROXY_RELAYS=<limit>
>
> MAX_OPAQUE_MULTICAST_RELAYS=<limit>

Where <limit> is the max number of that specific resource type which can exist simultaneously on a single server. By default the limit is unspecified which results in no limit being enforced.

4.2 Add an account

Each CMS hosted by the CRS have a unique account, identified by the account ID. The account ID and shared secret must match the settings used by the corresponding CMS!

To add an account use the CLI command "create account" (for more detail please see command description below).

```
$ crs-cli create account <account ID> [options]
```

In case the shared secret is already agreed upon with the CMS administrator please use the options flag "-s". Use –s secret or –s "secret with spaces".

4.3 List accounts

To view the shared secret and other account details for accounts use CLI command "list account" (for more detail please see command description below).

```
$ crs-cli list account
```

4.4 Delete an account

To delete an account use the CLI command "delete account" (for more detail please see command description below).

This procedure will erase the account on all CRS servers and remove all users belonging to the deleted account.

```
$ crs-cli delete account <account ID>
```

4.5 Add an Inter-Working CRS

To enable communication between Security Domains (accounts) on this CRS with Security Domains hosted on an Inter-Working CRS instance a trust relationship must be established. A TLS session is established between the CRSs using TLS with PSK, where the PSK is the shared secret between the CRS hosts. Please note that some CRS hosts might operate with two CRS instances, in that case each CRS instance must be added, and the shared secret must be the same.

Default the CRS will listen for incoming Inter-Working connections on port 8080, to configure another port please add the following line to the CRS configuration file (/opt/crs/crs.conf): iw-listen-port <port> where <port> is the desired port to receive Inter-Working connections on.

To add an Inter-Working CRS instance use the CLI command "add iw-crs" (for more details please see command description below).

```
$ crs-cli add iw-crs <IP> <shared-secret> <key-length> [options]
```

In case a comment should be added please use the options flag "-c". Use –c comment or –c "comment with spaces".

Please note that incoming connection requests will be matched against the IP, i.e. the source IP of the Inter-Working CRS instance must be identical to the IP configured.

4.6 List Inter-Working CRS Instances

To view the IP, shared secret and other Inter-Working CRS details use CLI command "list iw-crs" (for more detail please see command description below).

```
$ crs-cli list iw-crs
```

4.7 Delete an Inter-Working CRS Instance

To delete an Inter-Working CRS instance use the CLI command "delete iw-crs" (for more detail please see command description below).

```
$ crs-cli delete iw-crs <IP>
```

4.8 Configure push notifications

Push notifications are required by CCA version 3.30.0 and later for iPhone. Unless push notifications are enabled, iPhone users must manually enter the app to be notified of incoming calls and messages. The push notifications carry no plaintext data that can identify neither the initiator nor the responder.

Step 1: Configure cps connection

To enable push notifications, first setup a connection to the Cryptify Push Server (CPS) with the line

```
cps cps.cryptify.net 5230 <id> <shared-secret> <key-length>
```

in the GLOBAL section of the configuration file /opt/crs/crs.conf on all servers, where *id*, *shared-secret*, and *key-length* should be replaced by the values provided to you by Cryptify. The CRS server needs to be able to establish connections to cps.cryptify.net at tcp port 5230.

Step 2: Add push to the features list

Finally, add "push" to the features list in the GLOBAL section of the configuration file:

```
features push
```

Step 3: (Optional, Recommended) – Configure 'Direct Push'

By configuring 'Direct Push' the CRS will handle the communication with APNS and will only use the CPS to receive APNS authentication tokens. Add type direct to apns in the push-config section of the configuration file:

```
push-config apns type direct
```

Direct push requires that the DNS name “api.push.apple.com” can be resolved and that outgoing tcp connections toward the resolved address at port 443 are allowed. Note that Apple uses a large pool of IP-addresses for “api.push.apple.com” and that the DNS result changes regularly; please see `https://support.apple.com/en-us/102266` for guidance on the IP ranges that should be allowed.

Direct push will also place time requirements on the CRS to be within 10 minutes of correct UTC time as the authentication tokens are only valid for a specified time period in respect to their issued timestamp. Should the CRS not follow this requirement it will lead to APNS rejecting push notifications received due to expired authentication tokens.

4.8.1 Disable push notifications

To disable push notifications, simply change the “features” line in the configuration file on all servers so that it does not contain “push”:

```
features
```

Then restart the CRS instances.

4.9 Add a Cryptify Remote Admin

A cryptify remote admin is allowed to manage a set of CRS commands remotely via our Crypify Remote Admin software (CRA), such as view statistics or apply a monthly update.

Step 1: create a remote admin

Run the CLI command "create admin" (described in section 5.4.3) to create a new remote admin.

```
$ crs-cli command create admin <name> <key-length>
```

Step 2: add account access to admin

By default an admin has no access to any account. Run the CLI command "add admin-account" (described in section 5.10.2) to give an admin access to manage an account.

```
$ crs-cli add admin-account <name> <accountID>
```

Step 3: add account authorizations

By default an admin with access to an account will have no authorizations for that account. Run the CLI command "add admin-account-authorization" (described in section 5.10.3) to give an admin authorization for an account. Example of authorizations are statistics (manage account statistics) and update (apply monthly updates).

```
$ crs-cli add admin-account-authorization <name> <accountID>
    <authorization>
```

4.10 Configure client keep-alive methods (advanced, optional)

All clients maintain a persistent TCP connection to the CRS, so that they can be notified on incoming calls or messages. To prevent intermediate network devices, such as routers implementing NAT, from purging the connection due to inactivity, the clients will periodically – usually about every 10–15 minutes, at a time selected by the OS in order to minimize energy consumption – ping the CRS.

If the client does not ping the CRS in the expected time interval, the CRS itself will issue a ping, to be able to clear out connection resources for clients that have gone missing.

By default, pings in both directions are so called WebSocket pings, which are carried end-to-end in the TLS-protected connection, and the CRS will initiate a ping if the client has remained silent for a little over 20 minutes. An alternative ping method is a so-called TCP keep-alive, which is essentially an empty TCP-packet.

In certain special network configurations, it is desirable to change the ping interval or method. Using the "keep-alive" configuration directive in the crs.conf, makes it possible to specify a different keep-alive method or interval for a specific IP-range or client credentials. The general syntax is "keep-alive ip/mask method interval" or "keep-alive uri@account-id method interval", where ip/mask specifies the IP-range in standard CIDR-syntax and uri@account-id specifies client credentials, method is either "tcp" (TCP keep-alive) or "ws" (WebSocket ping), and interval is the interval in milliseconds.

If the method is set to "ws" one can specify the optional argument grace-period in milliseconds with the following syntax "keep-alive ip/mask ws interval grace-period". Grace period is used to determine how long to wait for a response to a WebSocket ping before deciding that the connection is stale.

If the configuration directive is repeated, the first matching directive is used when a client connects, and the CRS will fallback to the default method and interval if no directive matches. If multiple CRSes are used, any configuration changes must be manually synchronized between the instances.

Note: This is an advanced feature that may negatively impact the battery life and connectivity of users. It should only be used in very special network conditions; please consult Cryptify's technical support before making changes.

4.11 Configure address maps (advanced, optional)

By default, each CRS instance is assumed to be reachable through a single IP-address that is used for both CRS-to-client and CRS-to-CRS communication. In certain network topologies, a CRS may instead have multiple network interfaces with different traffic policies. For instance, if the CRSes are connected via a VLAN, it may be desirable or event required that the CRS-to-CRS communication uses the VLAN rather than the public IP-addresses

For such topologies, the CRS can be configured to apply an address translation when initiating an outgoing connection or when forwarding media packets.

Example

A CRS pool with public IP-addresses 203.0.113.101 and 203.0.113.102 are also connected via a private VLAN, with IP-addresses 198.51.100.1 198.51.100.2. The public addresses are configured in crs.conf as usual:

```
ip crs1  203.0.113.101
ip crs2  203.0.113.102
```

and to ensure that the CRS will connect to the other CRS using the private VLAN, an address map is added to crs.conf:

```
addr-map 203.0.113.101  198.51.100.1
addr-map 203.0.113.102  198.51.100.2
```

Please note:

- The configured address map applies to both signaling (via TCP) and media data (via UDP).
- Address maps are also applied for Inter-Working CRS communication.
- The CRS may still use the loopback address to connect to itself.

4.12 Configure detach cleanup (advanced, optional)

Each client must register once connected to be allowed to send/receive messages to/from the CRS. By default, each client will remain connected if they are denied to register or if the CRS decides to unregister them.

In certain special scenarios, it is desirable to close the connection when the client is no longer registered. Using the "detach-cleanup" configuration directive in crs.conf makes it possible to specify a detach cleanup delay for a specific IP-range or client credentials. The general syntax is "detach-cleanup *ip/mask delay*" or "detach-cleanup *uri@account-id delay*", where ip/mask specifies the IP-range in standard CIDR-syntax and *uri@account-id* specifies client credentials and delay is the time in milliseconds to wait after a client is unregistered before closing the connection.

If the configuration directive is repeated, the first matching directive is used when a client is unregistered, and the CRS will fallback to the default behavior if no directive matches. If multiple CRSes are used, any configuration changes must be manually synchronized between the instances.

4.13 Upgrade from $<$ 4.31.0

First upgrade to CRS 4.31.0.

4.14 Upgrade from 4.31.0

The upgrade procedure is the following.

Step 1: check database backup configuration

Skip step 2 if no CRS in the pool is configured as a database backup. For details about database backups, see section 4.1.4.

Step 2: stop redis_backup

Stop all database backup services in the CRS pool. Ubuntu/RHEL command:

```
$ sudo systemctl stop redis_backup
```

Step 3: upgrade CRS instances

Ensure the authenticity of a software before installing. To maintain system integrity and prevent installation of tampered or malicious software, all updates must be verified as originating from Cryptify AB before installation.

Ubuntu command:

```
$ sudo dpkg -i crs_VERSION-1_amd.deb
```

RHEL command:

```
$ sudo rpm -U crs_VERSION-1_rhel_x86_64.rpm
```

4.15 Upgrade from 4.35

Beginning with CRS version 4.36.0 and later, the default TLS key length used for communication within a CRS pool is 256-bit (see section 4.1.3). The required upgrade procedure depends on whether any instances in the pool will remain below version 4.36.0 after the upgrade:

- If any instance within the pool will remain below version 4.36.0, follow the procedure in section 4.15.1.
- If all instances within the pool will be upgraded to version 4.36.0 or later, follow the procedure in section 4.15.2.

Inter-Working CRS connections can also use 256-bit protection if both sides are running CRS 4.36.0 or later but must be manually updated. Follow the procedure in section 4.15.3 to update Inter-Working CRS connections.

4.15.1 Upgrade procedure when any instance remains below version 4.36.0

Follow this procedure if at least one CRS instance in the pool will still run a version earlier than 4.36.0 after the upgrade.

Step 1: update the configuration

On each instance being updated to a version later than 4.35, update the configuration with the key length set to "128bit". See section 4.1.3 for configuration details.

Step 2: upgrade the instances

Ubuntu command:

```
$ sudo dpkg -i crs_VERSION-1_amd.deb
```

RHEL command:

```
$ sudo rpm -U crs_VERSION-1_rhel_x86_64.rpm
```

4.15.2 Upgrade procedure when all instances will run version 4.36.0 or later

Follow this procedure if all CRS instances in the pool will run version 4.36.0 or later after the upgrade.

Step 1: generate a new shared secret

For the best protection, generate a new shared secret that has at least 32 octets of entropy. See section 4.1.3 for instructions on generating a shared secret using the CLI.

Step 2: update the configuration file

Update the configuration file on all CRS instances with the new shared secret with the key length set to "256bit" (see section 4.1.3).

Step 3: restart any CRS instances already above 4.36.0

Ubuntu and RHEL command:

```
$ sudo systemctl restart crs
```

Step 4: upgrade any remaining CRS instances below version 4.36.0

Ubuntu command:

```
$ sudo dpkg -i crs_VERSION-1_amd.deb
```

RHEL command:

```
$ sudo rpm -U crs_VERSION-1_rhel_x86_64.rpm
```

4.15.3 Update procedure for Inter-Working CRS connections

Inter-Working CRS connections will remain at 128-bit protection after upgrading unless manually updated to 256-bit protection. 256-bit protection is only supported if both servers are running CRS version 4.36.0 or later, or CIR version 3.3.0 or later.

Step 1: generate a new shared secret

For the best protection, generate a new shared secret that has at least 32 octets of entropy that both parties must agree upon. Note that the shared secret should only be disclosed to the other party over a secure channel. See section 4.1.3 for instructions on generating a shared secret using the CLI.

Step 2: update the Inter-Working connection

Both parties must agree upon when to update the Inter-Working connection since the connection will be broken once one CRS has had their configuration updated until both CRSs have had their configurations updated. When performing the update, the existing Inter-Working configuration must be removed using

```
crs-cli iw-crs delete
```

see section 5.6.1 for details. The Inter-Working connection can then be reconfigured using

```
crs-cli iw-crs add
```

with 256-bit protection set. See section 5.10 for details.

4.16 Back up and restore

4.16.1 Back up

To create a backup archive

```
$ crs-cli backup
```

This will create a backup under /opt/crs/backup/crs-backup-<crs instance>-<date>.tar with the files:

File: `/opt/crs/crs.conf`

File: `/opt/crs/crs-local.conf`

Dir: `/opt/crs/accounts`

Dir: `/opt/crs/data`

Store the backup archive on separate hardware or external media.

4.16.2 Restore

After reinstalling the software, extract the back archive.

```
tar xfv crs-backup-<crs instance><date>.tar -C /
chown -R crsuser:crsuser /opt/crs
chmod 644 /opt/crs/crs.conf
chmod 644 /opt/crs/crs-local.conf
chmod -R 644 /opt/crs/accounts
chmod 755 /opt/crs/accounts
chmod 755 /opt/crs/accounts/*
```

Reload the CRS service with the restored configuration data.

Ubuntu/RHEL command:

```
$ sudo systemctl restart crs
```

4.17 Monthly key updates

Monthly updates can be handled either by the CRS operator or by a remote admin via our Cryptify Remote Admin software (CRA).

If this is handled by the CRS operator the CMS administrator provides the protected update file to the CRS operator (for instance by email).

Save the update file on a suitable location on one CRS instance, e.g. /opt/crs/accounts/ on crs1.

Apply the update

```
$ crs-cli apply <path to update file>
```

(e.g. crs-cli apply /opt/crs/accounts/2025-10.upd)

4.18 Block users

In the current release blacklisting users needs to be handled by the CRS operator.

Add a user to the blacklist

```
$ crs-cli blacklist add <account ID> <URI>
```

Remove a user from the blacklist

```
$ crs-cli blacklist remove <account ID> <URI>
```

List blacklisted users

```
$ crs-cli show blacklist <account ID>
```

4.19 Trace

To trace events that occur in real-time use the CLI command "trace". Elements that can be traced are *attach*, *session*, *message*, *event*, *connection*, *statistics* and *client*. (for more detail please see command description below).

```
$ crs-cli trace <element>
```

To stop a trace press "ctrl-c".

5 CLI commands

crs-cli is a tool to manage accounts, show status of the system, and show user and usage statistics.

crs-cli itself is controlled entirely via command line parameters, which consist of one action and one or more options. The action parameter tells crs-cli what to do and options controls the behavior of the action in some way.

An autocompletion script for the bash shell is bundled with the CRS software, enabling tab completion for the crs-cli tool. Bash autocompletion is enabled by

default on Ubuntu, but on RHEL one must install the bash-completion package ("yum install bash-completion") and restart the shell to enable tab completion.

5.1 apply

```
$ crs-cli apply <path/to/file>
```

Unprotects the update file, places the unprotected update file in the corresponding account directory on all CRS instances, and trigger the CRS service to reload the accounts.

5.2 backup

```
$ crs-cli backup
```

This will create a backup under /opt/crs/backup/crs-backup-<crs instance>-<date>.tar with the files:

File: `/opt/crs/crs.conf`

File: `/opt/crs/crs-local.conf`

Dir: `/opt/crs/accounts`

Dir: `/opt/crs/data`

5.3 show

```
$ crs-cli show <element>
```

shows information of specified element.

5.3.1 attach

URI `the mobile number of the client.`

ACCOUNT `the account/security domain the client belongs to`

TIME `time attached in seconds`

IP:PORT `the IP address and port number of the client`

CRS `the CRS that handles the client`

5.3.2 blacklist <account-id>

List of blacklisted users, listed in the following format

URI `the mobile number of the client.`

5.3.3 session

List of active session, in the following format

CALLER `The URI of the party initiating the call.`

CALLEE `The URI of the called party.`

TIME `Total time of the session, in seconds.`

STATE `The state of the session, could be one of the following:`

invite: `Call invite sent to called party.`

ringing: `Ringing at the called party.`

talk: `Session established, parties talking.`

responder not found: `CRS cannot locate responder.`

responder not connected: `Responder not on-line.`

hangup(reason, who): `Session ended with reason and who ended the session.`

5.3.4 message

List of posted and delivered messages per users, listed in the following format

URI `the mobile number of the client.`

POST `the number of successfully posted messages by the client.`

DELIV `the number of successfully delivered messages sent by the client.`

READ `The last occasion, in UTC time format, where the client looked for messages.`

ACCOUNT `the account/security domain the client belongs to.`

5.3.5 system

Show the system components and allocations, listed in the following format

SERVER `the CRS instance crs1, crs2`

PROCESS `<app: <module> >`

PID `Process id`

SLOTS `Allocated slots`

5.3.6 version

Show the version of the installed CRS software

5.3.7 iw-crs

Show the state of any Inter-Working CRS connections, listed in the following format

IW-CRS `Address of Inter-Working connection`

SIGNAL-LINK `State of signal connection`

SIGNAL-OUTGOING `State of outgoing signal connection`

SIGNAL-INCOMING `State of incoming signal connection`

DATA-LINK `State of data connection`

DATA-OUTGOING `State of outgoing data connection`

DATA-INCOMING State of incoming data connection

VERSION Version of Inter-Working CRS

5.3.8 account

Show information about loaded accounts, listed in the following format

SERVER Server name (as configured in crs-local.conf)

MODULE Module which handles client connections for accounts (e.g app:attach)

PID Process id which runs module

LOADED ACCOUNTS Loaded accounts which are handled by this process id

5.3.9 relay

Show relay statistics for the CRS

For each relay type:

rx Received media packets

rxBytes Received bytes

tx Transmitted media packets

txBytes Transmitted bytes

Sent:

packets Total transmitted packets for all relay types

bytes Total transmitted bytes for all relay types

errorNoDest Failed transmissions due to invalid address

error Failed transmissions

Received:

packets Total transmitted packets for all relay types

bytes Total transmitted bytes for all relay types

errorNoDest Failed transmissions due to invalid address

error Failed transmissions

5.3.10 relays

Show active relay resources, listed in the following format

TYPE The type of the relay resource

NODES Number of nodes currently active in resource

LAST ACTIVE The time when the resource was last active

IDENTIFIERS `The identifiers which references the resource`

NOTE: Communication is interrupted if the CRS software is updated while a relay is in use.

5.3.11 admins

Show existing admins on the CRS, listed in the following format

Admin `Admin name`

Accounts `Accounts the admin has access to`

5.3.12 admin <name>

Show an existing admin on the crs

Name `Name of admin`

Secret `Secret of the admin`

TLS key length `Key length used for the admin TLS connection`

5.3.13 admin-accounts <name>

Show all accounts an admin has access to manage and what authorizations the admin has on each account

Account `CMS account id`

Authorizations `What authorizations the admin has for the account`

5.3.14 updates

Show applied monthly updates for current and next month for each account, listed in the following format

Account `CMS account id`

Current Month `Timestamp in UTC indicating when the monthly update was created or 'Not Available'`

Next Month `Timestamp in UTC indicating when the monthly update was created or 'Not Available'`

5.3.15 services

Check status of all CRS services, listed in the following format

Server `Name of the CRS server running the service`

Service `Name of service. (crs, crs_relay, redis_high/low/backup`

Status `Status of service. (active, inactive)`

5.3.16 client-info <account-id>

Show client information from specified account, listed in the following format

Uri	The mobile number of the client
Timestamp	The time when the client stored the information
Client Time	Client time compared to the time of the CRS server
Scan Time	The time when client scanned the QR code
OS	The operating system which the client is running on
OS Version	The version of the operating system
CCA Version	The version of the cryptify call application the client is running

5.3.17 push-token <account-id> <uri>

Show client push token and its type

Created	Timestamp when client uploaded the push tokens
Saved	Timestamp when CRS saved the push tokens
Age	The age of the push tokens based on when they where saved on the CRS
Client-Type	The type of client which uploaded the push tokens
Tokens	The push tokens

5.3.18 client-storage <account-id>

Show information about client stored data.

Server	Name of the CRS server storing the information
Uri	The mobile number of the client
Bytes	Number of bytes stored for client
Last Updated	Time when client last updated the information
Type	Type of information stored

5.3.19 team-code <account-id> <team-code>

Show information for the team code in the specified account.

URI	The mobile number of the client
SETTINGS	The settings stored by the client for the specified team code

5.4 create

```
$ crs-cli create <element>
```

create resource of specified element

5.4.1 account <account-id> [options]

Creates an account.

The account ID and shared secret shall be shared with and used by the corresponding CMS. Unless a shared secret is provided, using the '-s' option, a shared secret will be generated. Unless a specific key length is requested, using '-b' or '-B' option, the key length will be 128 bits.

account ID `string uniquely identifying a CMS/security domain. Valid characters are [a-z, A-Z, 0-9, -, _]. Length less than 20`

options:

`-s shared secret. Base64 encoded string`

`-b number of bits, rounded up to whole bytes`

`-B number of Bytes`

5.4.2 sharedSecret [options]

creates a random string encoded to base64, default 128 bits

options:

`-b number of bits, rounded up to whole bytes`

`-B number of Bytes`

5.4.3 admin <name> <key-length>

Creates an administrator to use with CRS Remote Admin tool. Output will show the generated secret for the created admin.

name `Unique name which identifies administrator. Valid characters are [a-z, A-Z, 0-9, -, _]. Length must be less than 20.`

key-length `TLS key length. Must be either ‘‘128bit’’ or ‘‘256bit’’.`

5.4.4 conference-room <number> <account-id> <uri>

Create a conference room number for a tel-uri in a specific account.

number `The conference room number to create. Valid characters are [0-9]. Length = 6.`

account-id `The account which the conference room owner belongs to`

uri `Tel-uri which should own the conference room`

5.5 reload account

```
$ crs-cli reload account
```

reloads the CRS, e.g. after account updates such as modified blacklists or monthly key updates

5.6 delete

```
$ crs-cli delete <element>
```

delete resource of specified element

5.6.1 account <account-id>

Permanently deletes the account.

```
account-id The account name
```

5.6.2 iw-crs <ip>

Delete the Inter-Working CRS instance configuration

```
ip IP address of the Inter-Working CRS instance
```

5.6.3 admin <name>

Delete an admin from the CRS

```
name Name of the admin to delete
```

5.6.4 admin-account <name> <account-id>

Delete account access for an admin

```
name Name of admin to delete authorization for
account-id The account id to delete authorization on
```

5.6.5 admin-account-authorization <name> <account-id> <authorization>

Delete authorization for admin on an account

```
name Name of the admin to delete authorization for
account-id The account id to delete authorization on
authorization The specified authorization to delete
```

5.6.6 client-info <account-id> <uri>

Delete client info for uri in specified account.

```
account-id The account id for the uri
       uri The uri to delete client info for
```

5.6.7 conference-room <number> <account-id> <uri>

Delete a conference room number for a tel-uri in a specific account.

```
number The conference room number to delete
account-id The account which the conference room owner belongs to
       uri Tel-uri which owns the conference room
```

5.7 list

```
$ crs-cli list <resource>
```

List a specific resource.

5.7.1 account

List accounts registered on the CRS in the following format

```
        ID account ID
      path path to the configuration and update files for the account
shared secret the shared secret between the CRS and corresponding CMS
```

5.7.2 iw-crs

List Inter-Working CRS instances in the following format

```
        IP IP address
shared secret the shared secret
TLS key length key length used for the TLS connection
   comment comment, if any
```

5.7.3 conference-room

List all the conference rooms on the CRS in the following format.

```
CONFERENCE-ROOM The conference room number
       ACCOUNT The account id that the conference room belongs to
           URI Tel-uri which owns the conference room
```

5.7.4 team-code <account-id>

List all team codes in the current month update file for the provided account in the following format.

```
      CODE The team code
   MEMBERS Comma-separated list of member numbers belonging to the team code
```

5.8 blacklist

5.8.1 add <account-id> <uri>

Add a user to the blacklist, i.e. stopping the user from using the service

```
 account-id The account id for the uri
        uri the mobile number of the client
```

5.8.2 remove <account-id> <uri>

Removes a user from the blacklists.

account-id The account id for the uri

uri the mobile number of the client

5.9 purge

5.9.1 client <account-id> <uri>

Purges the TLS connections of a client from the CRS. The client will automatically reconnect again.

account-id The account id for the uri

uri the mobile number of the client

5.9.2 clients <account-id>

Purges the TLS connections of all clients from the CRS. The client will automatically reconnect again.

account-id The account id for the uri

5.9.3 observed <account-id>

Purge observed clients for specified account. Observed clients are considered part of account even if not connected nor listed. (Used when moving unlisted to a connected domain)

account-id The account id for the uri

5.10 add

$ crs-cli add <element>

Add resource of specified element

5.10.1 iw-crs <ip> <shared-secret> <key-length> [options]

Adds an Inter-Working CRS Instance. The shared secret shall be shared with the host of the Inter-Working CRS

ip IP address.

shared secret Shared secret.

key-length TLS key length. Must be either ‘‘128bit’’ or ‘‘256bit’’. This CRS and the Inter-Working CRS must use the same key length for this connection.

options syntax <flag> <value>:

-c: Comment. Use -c comment or -c ‘‘comment with spaces’’.

5.10.2 admin-account <name> <account-id>

Add account which an admin has access to manage

name Name of admin to add authorization for

account-id The account id to add authorization on

5.10.3 admin-account-authorization <name> <account-id> <authorization>

Add authorization for admin on an account

name Name of the admin to add authorization for

account-id The account id to add authorization on

authorization The specified authorization to add:

update: Authorization to apply monthly updates for an account

statistics: Authorization to manage statistics for an account

client-info: Authorization to view client information for an account

5.11 lookup

```
$ crs-cli lookup <resource>
```

Lookup a resource

5.11.1 client <account-id> <uri> [options]

Shows information about the given uri.

account-id The account id for the uri

uri the mobile number of the client

options syntax <flag> <value>:

-v verbose flag, does not need a value. Prints additional information

Output:

Source Where the client is located. Prints which crs pool the client is connected to

Uri The mobile number of the client

Domain The domain the client belongs to

Client token Information uploaded by the client

Domain token The domain token for the domain the client belongs to

5.11.2 conference-room <number> <account-id>

List all found conference rooms matching the number based on the provided account id.

number The conference room number to lookup

account-id The account id for the uri

Output:

CONFERENCE-ROOM Conference room number

DOMAIN The domain the conference room was found in

URI The tel URI which owns the conference room

5.12 trace

```
$ crs-cli trace <element>
```

trace information of specified element. Press "ctrl-c" to stop a trace

5.12.1 attach

Events associated with registration states of the clients are traced, in the following format:

Header event

Time UTC time for the logged event

Connection-Id id of the connection the client has attached with

Type the type of the registration, can be [primary, secondary]

URI the mobile number of the client

Account the account the client belongs to

IP the IP address of the client for the TLS connection with the CRS

Port the IP port of the client for the TLS connection with the CRS

State the state of the attach event, can be [registered, unregistered, denied]

Description a description of the event state

5.12.2 session

Events associated with calling are traced, in the following format:

Header event

Time UTC time for the logged event

Caller the mobile number of the initiator

Callee the mobile number of the receiver

State the state of the session, can be:

invite: Call invite sent to called party

ringing: Ringing at the called party

talk: Session established, parties talking

responder not found: CRS cannot locate responder

responder not connected: Responder not on-line

hangup(reason; who): Session ended with reason and who ended the session

Account the account the client belongs to

Session-ID the identity of the session

5.12.3 message

Events associated with messaging are traced, in the following format:

Header event

Time UTC time for the logged event

From the mobile number of the initiator

To the mobile number of the receiver

State the state of the message, can be:

not-found: the number of the receiver could not be found

posted content: the message, with content, has been received by the CRS

posted ack: the delivery acknowledgement, sent by the receiver to confirm delivery of the message, has been received by the CRS

delivered: the message has been fetched by the receiver

forward content: the message, with content, has been forwarded to another CRS instance. Please check Account to see if the message has been forwarded to another CRS instance within the own CRS pool (Account = internal), or if the message has been forwarded to a peered CRS instance (Account = external)

forward ack: the delivery acknowledgement has been forwarded to another CRS instance. Please check Account to see if the message has been forwarded to another CRS instance within the own CRS pool (Account = internal), or if the message has been forwarded to a peered CRS instance (Account = external

Account the account the receiver, can be [account-id, internal, external], where:

Account ID: if the receiver is located on the current CRS instance

internal: if the receiver is located on another CRS instance within the same CRS pool

external: if the receiver is located on peered CRS instance, i.e. the receiver belongs to a security domain located on a peered CRS

Message-ID the identity of the Message

5.12.4 object

Events associated with client object (e.g client token) are traced, in the following format:

Header event

Time UTC time for the logged event

From the mobile number of the initiator

Event Object event description. (e.g update client_token)

5.12.5 event

Various debug events, not specifically associated with session, attach or message event will be displayed in the following format:

Module logging module

Type the type of the registration, can be [dbg]

Time UTC time for the logged event

Information log information

5.12.6 connection

Events associated with connection states of the clients/admins are traced, in the following format:

Header event

Time UTC time for the logged event

Id id of the connection

Type type of the user which the connection event is for, can be [client, admin, unknown]

User description of the user which the connection event is for. Format will depend on User-Type, <URI>@<Account> for type client, the admin name for type admin or ‘‘unknown’’ when user type is unknown.

IP the IP address of the client for the TLS connection with the CRS

Port the IP port of the client for the TLS connection with the CRS

State the state of the attach event, can be:

connected: TCP connection established

```
authenticated: TLS handshake has finished
negotiated: WebSocket upgrade has finished
closed: TCP connection is closed
```

5.12.7 statistics

Various statistic events such as outgoing/incoming messages, calls made and more, statistics event will be displayed in the following format:

Header `event`

Time `UTC time for the logged event`

Type `the type of statistics event`

URI `the mobile number of the client`

Account `the account the client belongs to`

5.12.8 client <account ID> <URI> [options]

Events related to a specific URI on a specific account.

options:

--verbose `Prints additional information.`

5.13 stats

```
$ crs-cli stats
```

Manage statistics for an account. (Disabled by default)

5.13.1 enable <account-id>

Enable statistics collection for account.

5.13.2 disable <account-id>

Disable statistics collection for account. (Default)

5.13.3 clear <account-id>

Clear statistics for account.

5.13.4 export <account-id>

Export statistics for account in tab separated values format

tel-uri `Client uri`

session-outgoing `Number of outgoing sessions`

session-incoming `Number of incoming sessions`

message-outgoing `Number of outgoing messages`

message-incoming	Number of incoming messages
system-outgoing	Number of outgoing system messages, e.g group invites and updates
system-incoming	Number of incoming system messages, e.g group invites and updates
conference-calls	Number of conference calls
initiated-multi-session	Number of initiated multi sessions, e.g sessions with multiple media streams (audio and video)
accepted-multi-session	Number of accepted multi sessions, e.g sessions with multiple media streams (audio and video)
sent-group-call-invite	Number of group call invites sent
accepted-group-call-invite	Number of group call invites accepted
current-offered	States if crs has current months crypto update to offer.
current-confirmed	States if client has accepted current months crypto update
next-offered	States if crs has next months crypto update to offer
next-confirmed	States if client has accepted next months crypto update

5.14 summarize

```
$ crs-cli summarize client-info [options]
```

Summarize client information from all accounts options:

--max-age <days>	Specify the max age from when the client stored the information to current time in days. Defaults to 90 days
--os	Only show summary of OS versions
--client-version	Only show summary CCA versions

Output listed in the following formats:

OS	The operating system which the client is running on
CCA Version	The version of the Cryptify Call application the client is running
Users	The number of users for the preceding column

5.15 client-info

```
$ crs-cli client-info <account-id>
```

Show client information from specified account, listed in the following format

Uri	The mobile number of the client
Timestamp	The time when the client stored the information
Client Time	Client time compared to the time of the CRS server
Scan Time	The time when client scanned the QR code
OS	The operating system which the client is running on

OS Version The version of the operating system

CCA Version The version of the cryptify call application the client is running

5.16 check

$ crs-cli check slot <uri>

Check the slot number for a mobile number

uri The mobile number of the client

5.17 health-check

$ crs-cli health-check <type> [option]

Perform a health check of a specified type. *Exit codes: 0=ok, 1=error, 3=warning*

5.17.1 system [options]

Perform general system tests such as check that CRS services are running and that connections to other CRS instances are up.

5.17.2 account <account-id> [options]

Perform tests for the specified account such as making a call and sending a message using temporary test clients as well as verifying that update files are up to date. Test clients will connect to the CRS with a URI in the format *"urn:x-crs:client:test-<index>-<id>"*

options:

--output-format <format> Supported formats:

summary: Result will be either [ok, warning or error]

json: Result will be in JSON format and includes both the summary of the overall result as well as the result and description of each individual tests that where performed. (Default)

tsv: Result will be in TSV format and includes the result and description of each individual tests that where performed

--local-only Limit the account tests to the local CRS only. This will ensure that the test clients only connect to the loopback interface on the local CRS

5.18 update-notification

$ crs-cli update-notification <cmd>

Manage crypto update notifications for an account. (Disabled by default)

5.18.1 enable <account-id>

Enable crypto update notifications for account.

5.18.2 disable <account-id>

Disable crypto update notifications for account.

6 Logs

Logs are stored at /var/log/crs/

The logs shall be reviewed regularly for unexpected entries in order to detect malfunctioning software.

6.1 crs.log

System events from the CRS is logged in the crs.log in the following format:

Time `UTC time for the logged event`

Type `can be [DEBUG, INFO, WARN, INIT]`

Module `logging module`

PID `process ID`

Information `log information`

6.2 crs_relay.log

This log contains event of the CRS relay function. The following events are logged

Start, i.e. start of the CRS relay service

Stop, i.e. stop of the CRS relay service

CONNECTION, i.e. when the CRS connects to the CRS relay

CLOSE, i.e. when the CRS connection is closed

Command add, i.e. when a new conversation is created

Command poll, i.e. a heartbeat between the CRS and CRS relay

6.3 redis_high.log, redis_low.log and redis_backup.log

Log files generated by the redis service

7 Fault Management

7.1 System

Every time a system event occurs a notification script will be run.

The notification script is located at

/opt/crs/scripts/notify.sh <unit/modulename> <event>

By default each event is logged to syslog, but it is possible to add commands in this script to integrate to external fault management systems as well, e.g. by triggering snmp traps.

7.2 Push notifications

To allow for push notifications the CRS is required to establish a connection towards the CPS. This will in "proxy" mode allow the CRS to send push notifications via the CPS, and for "direct" mode, the connection is required to allow the CRS to request authentication tokens to be allowed to send push notifications directly to the push servers.

Note: To get all the logs mentioned in the next sections regarding push notifications at once, run the following command:

```
$ grep "CPS connection\|Push service error\|Push notification
    failed" /var/log/crs.log
```

7.2.1 CPS connection

To troubleshoot for any CPS connection issues in the CRS log file run the following command:

```
$ grep "CPS connection" /var/log/crs/crs.log
```

This will show any of the following outputs:

CPS connection failed due to: <reason>

Indicates that the CRS failed to connect to the CPS and the reason why

CPS connection error (<error>)

Indicates that the CPS connection has reported an error

CPS connection down (<error>) (<description>)

Indicates that the CPS connection was closed and the error and description of why this occurred

CPS connection up

Indicates that a connection to the CPS was established

7.2.2 Push server connection

When the CRS is configured in "direct" mode it is required to establish a connection towards the push server to be able to send push notifications to the clients.

To troubleshoot for any push server connection issues in the CRS log file run the following command:

```
$ grep "Push service error" /var/log/crs/crs.log
```

This will show any issues with the push server connection in the following format: • Push service error <error> ◦ This indicates that the push service connection has failed with error.

7.2.3 Failed push notification

When the CRS is configured in "direct" mode it will send push notifications directly to the push servers to be delivered to the clients.

To troubleshoot for failed push notification requests in the CRS log file run the following command:

```
$ grep "Push notification failed" /var/log/crs/crs.log
```

This will show any failed attempts at sending push notifications toward clients in the following format:

> Push notification failed - <reason> ("<uri>@<account-id>"), ("<token>")

8 Dimensioning guide

The dimensioning guideline below is per CRS server.

8.1 Storage

When a user adds an attachment to a letter draft, the attachment is stored on the CRS for 2 week pending to be sent. When a letter is received, its attachments are stored on the receiving CRS for 5 weeks, pending user download. In the latter case, the attachments are also stored on the peered CRS, if available. Furthermore, some temporary storage is used when the attachments are transferred within the CRS.

As a rule of thumb, each CRS should have at least R×S×5(week) storage available, where R is the estimated rate of attachments received, S the average attachment size. If we, for example, estimate that each of the N users receive one 1 MB-sized attachment per day, we see that there should be at least 4 GB of storage available for every 100 users. The CRS administrator should ensure that the CRS never runs out of storage space.

8.2 System memory

The CRS is optimized to operate even if only 4 GB of application memory is available. However, adding more system memory will optimize the performance, and it is recommended to have 1 (GB) + 250 (MB) × C + 5 (MB) × N, where C is the number of federated CRSes and N is the number of users.

8.3 Transfer speed

8.3.1 Messages

When a user sends a message, the client expects to receive a confirmation that the message has been successfully stored within a certain time. Hence, it is important that the transfer speed to other CRS servers, both peered and interworking, is

sufficiently fast. Obviously, the link speed L must satisfy L » R×S, where R is the estimated rate of attachments transferred and S is the average attachment size. For the best user experience, a client should receive a response within 30 seconds even during usage spikes.

Consider a system consisting of N users, where during a 10-hour period each user sends one 1MB attachment during a 10-hour period. If the N messages are evenly distributed throughout the period, there are N/1200 messages in a 30 second slot. Hence, if L » N/1200 × 1 MB/30 (s), or L » N × 0.22 kbit/s, each attachment upload will complete well within 30 seconds.

Even though L refers to the actual transfer speed, rather than the theoretical maximum bandwidth, we see that just a 10 Mbps link can handle 10 000 users in such a system and still have plenty of bandwidth left for attachment downloads.

8.3.2 Calls

Voice calls have very low bandwidth requirements – less than 20 kbit/s per call – but are sensitive to packet loss and latency. The same is true for conference calls, which scale linearly with the number of participants. That is, it is important to prevent congestion by having enough bandwidth overhead.

The bandwidth requirements for video calls are much higher, and depend on how well the video compresses. As a rule of thumb, each video call consumes of the order of 1 Mbit/s, but will packet loss and lower bandwidth settings are gracefully handled by reducing the video quality.

8.3.3 Conference calls

In a conference call, each user transmits audio RTP packets to the CRS, generating about 10 kbit/s with 16–20 packets per second. Since the downlink to each user utilizes discontinuous transmission, the ingress and egress traffic for conference call with N participants are both merely N times that of the ingress traffic from a single user.

When a user enables video in a conference call, that user also transmits HD-audio and video to the CRS, together generating ingress traffic of about 500 kbit/s with 120 packets per second. In the case of video, it is of course not possible to use discontinuous transmission. On the other hand, it is only necessary to receive the streams that are visible on the screen, which in current versions of Cryptify Call is limited to 6 per user.

That is, in a conference call with N participants where each user has video activated, the ingress traffic to the CRS will be N times that of the ingress traffic from a single client, whereas the egress traffic from the CRS will be 6N times that of the ingress traffic from a single client.

Screencast works similar to video, but uses slightly more network resources; 800 kbit/s at 80 packets per second.

	INGRESS	EGRESS
Basic audio	N * 10 kbit/s N * 20 packets/s	N * 10 kbit/s N * 20 packets/s
Video	n * 500 kbit/s n * 120 packets/s	N * min(n,6) * 500 kbit/s N * min(n,6) * 120 packets/s
Screencast	m * 800 kbit/s m * 80 packets/s	N * min(m,6) * 800 kbit/s N * min(m,6) * 80 packets/s

Table 5: Conference call network requirements for N participants, n and m of which are transmitting video and screencast, respectively.