EVERTRUST

EverTrust Stream documentation v2.1

Administration Guide

EVERTRUST

Table of Contents

1. Introduction

1.1. Description

Stream is EverTrust Certificate Authority solution and is powered up by:

- Pekko
- BouncyCastle
- MongoDB
- Kamon
- Play! Framework
- Scala
- NGINX
- Vue.js
- Quasar

This document is specific to Stream version **2.1**, and may apply to follow-up minor releases.

1.2. Scope

This document is an administration guide detailing how to configure and operate Stream.

1.3. Out of Scope

This document does not describe how to install and bootstrap a Stream instance. Please refer to the installation guide for installation related tasks.

2. Managing Certification Authorities

2.1. Importing an External Certification Authority

1. Log in to the Stream Administration Interface.

2. Go to **Certification Authorities** > **External CAs** and click on  .

3. You need to provide the X509 CA Certificate, either by pasting it directly into the box or by importing the file. **PEM** and **DER** formats are supported. Then click "Next".

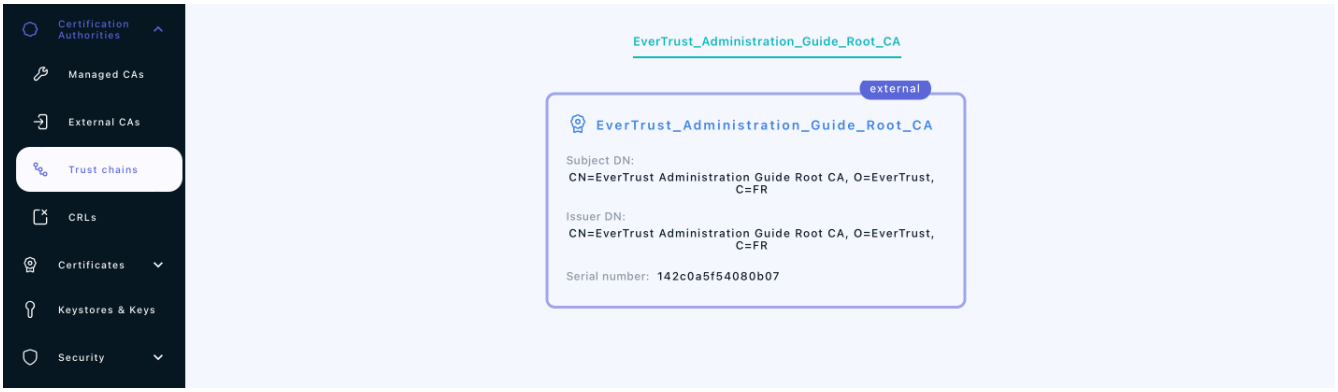
4. In the **Details** tab, check if the details that were parsed from the certificate match those of the CA you wish to import. If it does, click "Next".

5. In the Configuration tab, you can

- Add a **CRL**
- Edit the **Refresh period**
- Edit the **Timeout timer**
- Configure a **proxy**
- Toggle whether the external CA should be trusted for **server authentication** or **client authentication**
- Specify the **Outdated Revocation Status Policy**
- Enable OCSP and configure a **Default OCSP Signer**

6. You can then click the "Import" button in the bottom right corner to import your CA.

If everything was ok, you should see your CA marked as *external* if you go to **Certification Authorities** > **Trust chains**:



2.2. Importing an existing Managed Certification Authority

1. Log in to the Stream Administration Interface.

2. Go to **Import existing CA** from the menu on the left

3. Import your CA certificate file or paste the content of the file in the *Copy/paste the certificate* box. If you decide to paste the file's content, don't forget to click the parse button on the right before continuing.

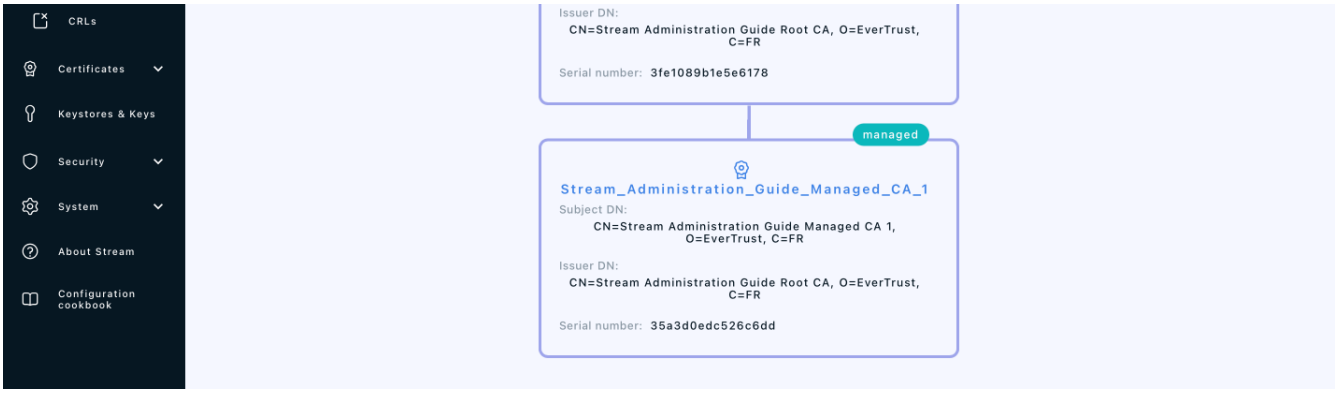
4. Scroll down to the bottom of the page and check the certificate's information. If everything is correct, click "Next".

5. Select the Keystore where your CA's key is stored. If you do not have a keystore set up yet, please refer to the ***Managing Keystores & Keys*** section.

6. Select the key that was used to generate the CA from the selected keystore and click "Next".

7. Upload your CA's CRL file and click "Add".

If everything was ok, you should see your CA marked as *managed* if you go to **Certification Authorities** > **Trust chains**:



2.3. Issuing a new Root Certification Authority

1. Log in to the Stream Administration Interface.

2. Go to **Create a new CA** from the menu on the left.

3. Input your CA's **internal name** and manage the **DNs** that you want to add (using the `Add a DN element` button on the bottom left corner) or to remove (using the icon).

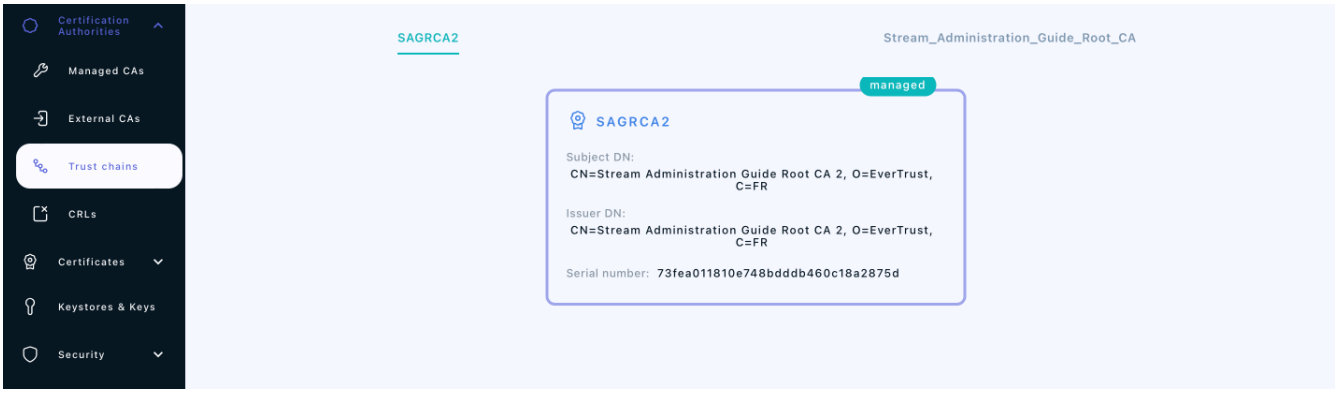
4. Select the **Keystore** that contains the key you want to use to generate this CA, then select the key that you want to use. If you do not have a keystore set up yet, please refer to the ***Managing Keystores & Keys*** section.

5. Select **Selfsigned** as a signing method, and pick the hash algorithm of your choice. Optionally, if you picked a PKCS#11 Keystore and an RSA key, you have the ability to use a PSS signature instead of the classic PKCS#1 one : if you wish to do so, just turn on the toggle. Note that your HSM must support the `CKM_RSA_PKCS_PSS` mechanism.

6. Set the **lifetime** of your CA in days. Optionally, you can set up a **backdate** and a **path length**. Once you are done, click "Add".

7. You can directly configure your CA from this menu, by turning on or off **enrollment**, trusting the CA for **client authentication** or **server authentication**, enabling **OCSP** or **enforcing key unicity**. Once you're satisfied with your settings, click "Add".

If everything was ok, you should see your CA marked as **managed** on a new trust chain if you go to **Certification Authorities** > **Trust chains**:



2.4. Issuing a subordinate Certification Authority

2.4.1. Signed locally

1. Log in to the Stream Administration Interface.

2. Go to **Create a new CA** from the menu on the left.

3. Input your CA's **internal name** and manage the **DNs** that you want to add (using the `Add a DN element` button on the bottom left corner) or to remove (using the 🗑 icon).

4. Select the **Keystore** that contains the key you want to use to generate this CA, then select the key that you want to use. If you do not have a keystore set up yet, please refer to the ***Managing Keystores & Keys*** section.

5. Select **Signed with an internal CA** as the signing method.

6. Select the **Managed CA** you want to sign the certificate with.

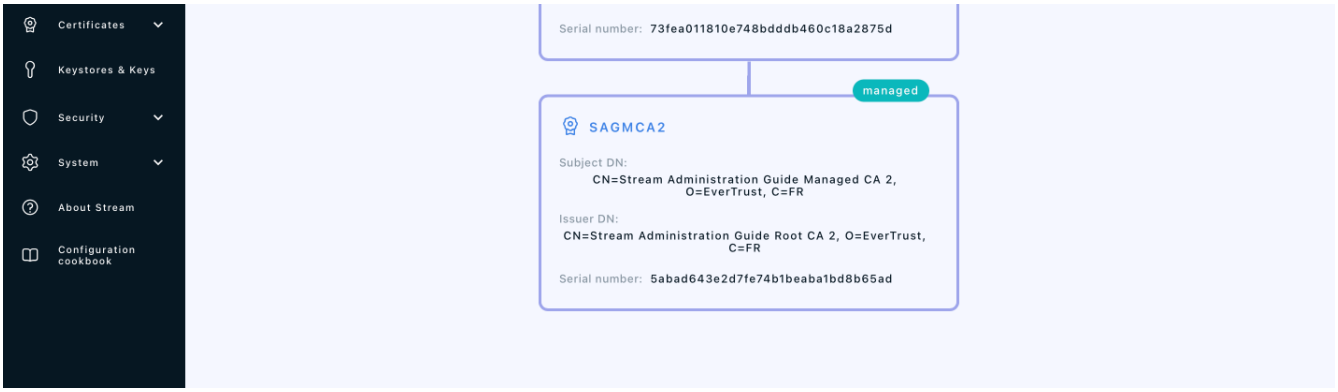
7. Set the **lifetime** or your CA in days. Optionally, you can set up a **backdate** and a **path length**.

8. Optionally, you can set up an **OID Policy**, a **CPS Pointer**, add **CRLDPs** and the CA's **AIA**. Once you are finished with the settings, click "Issue CA".

9. You can directly configure your CA from this menu, by turning on or off **enrollment**, trusting the CA for **client authentication** or **server authentication**, enabling **OCSP** or **enforcing key unicity**. Additionally, if you issued this CA using an RSA key from a PKCS#11 keystore, you can choose to use the PSS signature algorithm instead of the default PKCS#1 one to sign new certificates. To do so, simply turn on the toggle. Note that your HSM must support the `CKM_RSA_PKCS_PSS` mechanism.
Once you're satisfied with your settings, click "Add".

If everything was ok, you should see your CA marked as **managed** on a new trust chain if you go to

Certification Authorities > Trust chains:



2.4.2. Signed externally

1. Log in to the Stream Administration Interface.

2. Go to **Create a new CA** from the menu on the left.

3. Input your CA's **internal name** and manage the **DNs** that you want to add (using the `Add a DN element` button on the bottom left corner) or to remove (using the icon).

4. Select the **Keystore** that contains the key you want to use to generate this CA, then select the key that you want to use. If you do not have a keystore set up yet, please refer to the ***Managing Keystores & Keys*** section.

5. Select **Signed with an external CA** as the signing method.

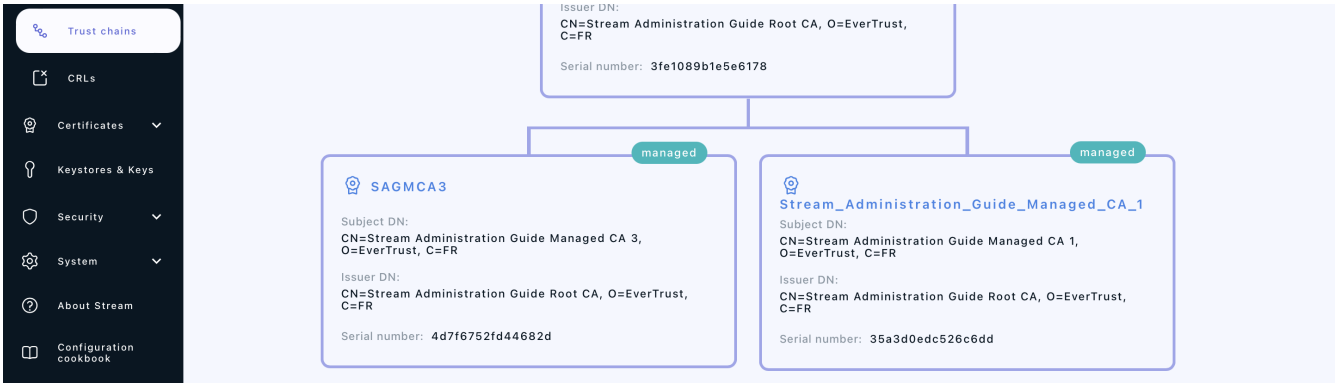
6. Click the link in the **Export** section to download the **CSR** for your CA, then sign it using your external CA and export the signed certificate under PEM or DER format.

7. Upload the signed certificate in the **Import** section.

8. Scroll down to the bottom of the page and check the certificate's information. If everything is correct, click "Next".

9. You can directly configure your CA from this menu, by turning on or off **enrollment**, trusting the CA for **client authentication** or **server authentication**, enabling **OCSP** or **enforcing key unicity**. Once you're satisfied with your settings, click "Add".

If everything was ok, your should see your CA marked as **managed** on a new trust chain if you go to **Certification Authorities > Trust chains**:



2.5. Note on CRLDP and AIA settings

NOTE	Regardless of the CA type, the setting "CRLDP" refers to the CRL of the CA you are configuring, and **NOT** the one of the issuing CRL. Same goes for the AIA: you need to specify the certificate of the CA you are configuring, and not the certificate of its issuing CA.

2.6. AIA Certificate Issuer

Stream allows you to download the **Certificate** of the **CAs** (external and managed). This is usually used in AIA issuer certificate extension to be able to download the certificate of the issuing Certificate Authority.

The standard download URL format is *http(s)://[stream_url]/aias/CA_internal_name*. This URL can be accessed by anyone without prior authentication, either through HTTP or HTTPS.

You need to specify the **Internal name** of the CA to download its **certificate** and not its **Common Name (CN)**.

The certificate format depends on the request `ACCEPT` header:

- `application/x-pem-file`: returns the certificate in **PEM**
- `application/pkix-cert`: returns the certificate in **DER**
- `application/x-pkcs7-certificates`: returns the certificate in **PKCS7**

If no `ACCEPT` header is specified, return the certificate in **DER**.

The certificate is returned with the following headers:

- `Content-Type`:
  - `application/x-pem-file` for **PEM**
  - `application/pkix-cert` for **DER**
  - `application/x-pkcs7-certificates` for **PKCS7**
- `Content-Disposition`: 'attachment; filename=<ca name>'

3. Managing Certificate Revocation

3.1. Configuring Certificate Revocation Lists for an External CA

1. Log in to the Stream Administration Interface ;

2. Go to **Certification Authorities** > **External CAs** and click on next to the name of the CA you want to import the CRL of ;

3. Select a valid CRL file that has been signed by your CA ;

4. If everything went through correctly, the CRL of that external CA should be available to download from Stream ;

5. Additionally, if you want to push the CRL into a CRL storage, click on the external CA ;

5.1 In the **Configuration** tab, select one or several previously created external storages from the drop-down menu:

- **On CRL update**: this will be triggered every time a new CRL is uploaded (see step **2**).
- **On CRL sync**: this will trigger every 15 minutes to ensure CRL is up to date on the storage, and push the new one if needed

5.2 Click the **Save** button at the top.

The CRL should now also be pushed in the CRL storage(s) whenever you manually import it into Stream. Note that the CRL will still be accessible from the standard Stream CRLDP.

3.2. Configuring Certificate Revocation Lists for a Managed CA

To manage the **CRLs** of a managed CA, you first need to set up a **CRL Policy**:

1. Log in to the Stream Administration Interface.

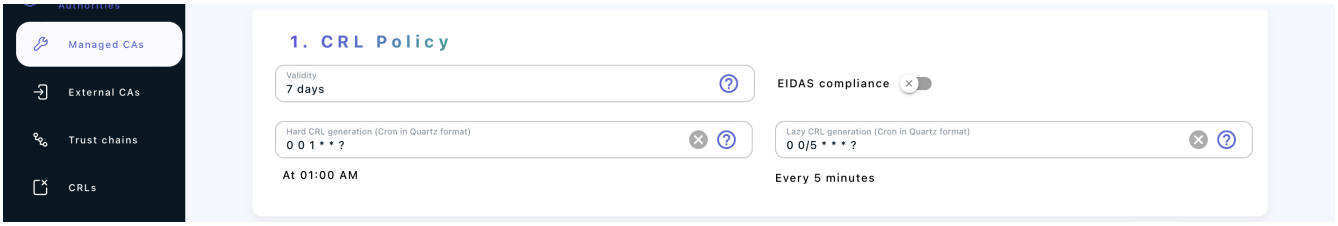
2. Go to **Certification Authorities** > **Managed CAs** and click on next to the name of the CA you want to edit the CRL policy of.

3. Go under the **CRL/OCSP** tab.

4. First, you need to define the validity period of your CRL, i.e. the period of time while your CRL is considered valid. The countdown starts at the moment the CRL is generated. If you want your CRLs to be valid for a week, you can type **7 days**.

5. You can then automate the **CRL generation** using either the **Hard CRL generation**, the **Lazy CRL generation** or both of them in combination:

- The **Hard CRL generation** parameter takes a cron expression in Quartz format and generates the CRL every time that cron expression is valid, without any condition. It is recommended to generate the **CRLs** every day. To generate a new **CRL** every day at 1 A.M., the cron expression is: 0 0 1 * * ?
- The **Lazy CRL generation** parameter takes a cron expression in Quartz format and checks if the CRL needs to be updated, i.e. if a certificate has been revoked, since the last CRL generation. If a certificate has been revoked since the last generation then a new CRL will then be generated, otherwise it will do nothing. It is recommended to have a short time span for the lazy generation so that the CRL always stays up to date. To check for possible CRL updates every 5 minutes, the cron expression is: 0 0/5 * * * ?



6. Click the **Save** button at the top of the page.

Now your CRL policy has been configured, and you've been redirected to the Managed CAs page.

You can then generate manually the CA's first CRL using the ↴ button next to the CA's name that you just configured. If you configured the **Hard** or the **Lazy** generation, your CRL will then automatically be updated according to the cron quartz expression you specified.

7. Additionally, if you want to push the CRL into other storages, click ✎ on the managed CA ;

7.1 In the **Configuration** tab, select one or several previously created external storages from the drop-down menu:

- **On CRL generation**: this will be triggered every time a new CRL is generates (manually or via the configuration at step **5**).
- **On CRL sync**: this will trigger every 15 minutes to ensure CRL is up to date on the storage, and push the new one if needed

7.2 Click the **Save** button at the top.

The CRL should now also be pushed in other storages. Note that the CRL will still be accessible from the standard Stream CRLDP.

3.3. Viewing CRLs

1. Log in to the Stream Administration Interface.

2. Go to **Revocation Management** > **CRLs**.

3. You can then see information regarding your CAs' CRLs that are going to be detailed below:

- The **CA** column indicates the name of the CA whose CRL is detailed in the line
- The **Number** column indicates the serial number of the CRL. It starts at 1 for the very first CRL generated and is incremented by 1 at each generation. It is displayed in hexadecimal format.
- The **Last update** column indicates the date and time when the current CRL was generated.
- The **Valid Until** column indicates the date and time when the current CRL will expire. It should be equal to *Last update* + the validity period you set in the **CRL policy** field.
- The **Next refresh** column indicates the date and time when the current CRL will be refreshed. It should be equal to the nearest date matching either cron quartz expression you set in the **CRL policy** field (lazy or hard).
- The **download** **button** allows you to download your CRL. It also serves as a CRLDP. For more information about CRLDPs in Stream, please refer to next section.
- The **generate** **button** allows you to manually refresh the CRL and generates a new one.
- The **refresh** **button** refreshes the information displayed in the tab, in case a generation happened in between. It **does not** refresh the CRLs, only the displayed information.

3.4. Downloading CRLs

Stream allows you to download the **CRLs** of the **CAs** it manages. The standard download URL format is *http(s)://[stream_url]/crls/CA_internal_name*. This URL can be accessed by anyone without prior authentication, either through HTTP or HTTPS.

You need to specify the **Internal name** of the CA to download its **CRL** and not its **Common Name (CN)**.

CRLs are by default generated and thus downloaded in **DER** format. You can specify *?form=PEM* at the end of the previously given URL to download the CRL in PEM format.

As an example, here are the **CRLDPs** of 2 different CAs that were set up through this guide:

- *https://stream.evertrust.fr/crls/SAGMCA2* will download the CRL for SAGMCA2 through HTTPS in DER format
- *http://stream.evertrust.fr/crls/SAGMCA3?form=PEM* will download the CRL for SAGMCA3 through HTTP in PEM format

3.5. External CRL Storages

3.5.1. Creating a Stream External Storage

Stream allows you to push your CRLs into other Stream instances upon generation, but it requires to create an external Stream CRL storage in the product first. This section also assumes that you have already configured `Password` or `Certificate` credentials for the desired stream instance.

To configure an external Stream CRL storage:

1. Log in to the Stream Administration Interface ;

2. Go to **Revocation management** › **External CRL Storage** and click on  ;

3. Fill in the information :

- Select the **Type*** of external CRL storage, Stream for a Stream storage
- The **Name*** to give to that external storage
- The **Description** to add more details about this storage
- Select a list of notifications to send **On execution error** to be alerted if the push to the Stream instance fails
- Add the technical name of the **CA** you wish to push the CRL to. 3 cases can occur:
  - the technical name of the CAs are aligned on both instances: the field should be left blank, as the trigger will by default use the technical name of the CA the CRL is linked to.
  - the technical name of the CAs on the other instance can be deduced from the technical name on the current instance. A template string can be used to format the name correctly.
  - the technical names are not linked in any way. The technical name on the other instance should be fully spelled out, and a trigger defined for each CA (using duplication)
- Enter the **Endpoint*** of your other Stream instance. This should include the protocol (https://).
- Select a **Credential*** to connect to the Stream instance. Only credentials on the `Stream` target can be selected.
- Choose a **Timeout** for the push request
- Add a **Proxy** to use to connect to the instance, if any

4. Once you've filled all the information, click "Add"

The External CRL Storage is now created and can be used in CA configuration.

3.5.2. Creating an S3 External CRL Storage

Stream allows you to push your CRLs into S3 buckets upon generation, but it implies to configure an external storage first. This section also assumes you have already configured `Password` credentials for a cloud provider if you want to use a cloud storage solution.

To configure an external S3 CRL storage:

1. Log in to the Stream Administration Interface ;

2. Go to **Revocation management › External CRL Storage** and click on  ;

3. Fill in the information :

- Select the **Type*** of external CRL storage, Amazon S3 for an S3 storage
- The **Name*** to give to that external storage
- The **Description** to add more details about this storage
- Select a list of notifications to send **On execution error** to be alerted if the push to the CRL storage fails
- Add the **Bucket*** of your S3 storage
- Select a **Credential** to connect to the S3 server (AWS format). Only credentials on the `AWS` target can be selected. If no credentials are specified, environment variable values will be used to establish connection.
- Add a **Role Arn** to use when connecting to the S3 provider (only applicable for AWS)
- Select the **Region** to use if the S3 is in the cloud (AWS, GCP)
- Add a **Proxy** to use to connect to the external storage, if any
- If not using an AWS S3 Bucket, add the S3 **Endpoint**
- Choose whether to **Force path style** in URL name
- Reconfigure the **CRL Alias**. By default, the S3 object key will be the technical name of the CA with `.crl` extension. Using template strings, this name can be modified. For example, if the file should be named with an uppercase of the CA's CN with the `.pem` extension, CRL Alias will be `{{ Upper({{ca.signer.dn.cn.1}}) }}.pem`

4. Once you've filled all the information, click "Add"

The External CRL Storage is now created and can be used in CA configuration.

3.5.3. Creating an LDAP External Storage

Stream allows you to push your CRLs into LDAP directories upon generation, but it requires to create an external LDAP storage in the product first. This section also assumes that you have already configured `Password` credentials for the desired LDAP directory.

To configure an external LDAP CRL storage:

1. Log in to the Stream Administration Interface ;

2. Go to **Revocation management › External CRL Storage** and click on  ;

3. Fill in the information :

- Select the **Type*** of external CRL storage, LDAP for an LDAP storage
- The **Name*** to give to that external storage
- The **Description** to add more details about this storage
- Select a list of notifications to send **On execution error** to be alerted if the push to the CRL storage fails
- Add the **Host***, IP or hostname of the LDAP server where the CRL will be pushed into. Don't add "ldap://" or "ldaps://" in the beginning
- Add the **Port*** on which the LDAP server is running (default is 389 for LDAP and 636 for LDAPS)
- Select a **Credential*** to connect to the LDAP server. Only credentials on the `LDAP` target can be selected.
- Add a **Proxy** to use to connect to the external storage, if any
- Enter a **Base DN*** that points the LDAP category to publish the CRL into
- Enter a LDAP search **Filter*** to find the resource where to publish the CRL into. **Example** : *(objectclass=cRLDistributionPoint)*
- Define the **CRL Attribute***, the resource attribute to publish the CRL into
- Choose whether to allow Stream to **follow LDAP referral** URLs
- Choose whether to use the **Secure** LDAPS protocol instead of the regular LDAP protocol
- Choose whether to **Disable hostname validation**, allowing Stream to connect to the LDAP server in LDAPS even if the server certificate does not have the specified hostname as a DNS SAN **(only if Secure is turned on)**

4. Once you've filled all the information, click "Add"

The External CRL Storage is now created and can be used in CA configuration.

3.5.4. Creating an SCP External Storage

Stream allows you to push your CRLs into any server supporting the SCP protocol upon generation. This section also assumes that you have already configured `SSH` credentials for the desired server.

To configure an external SCP CRL storage:

1. Log in to the Stream Administration Interface ;

2. Go to **Revocation management** › **External CRL Storage** and click on  ;

3. Fill in the information :

- Select the **Type*** of external CRL storage, SCP for an SCP storage
- The **Name*** to give to that external storage
- The **Description** to add more details about this storage
- Select a list of notifications to send **On execution error** to be alerted if the push to the CRL

storage fails

- Add the **Host***, IP or hostname of the SCP server where the CRL will be pushed into.
- Add the **Port*** on which the SCP server is running (default is 22 for SSH)
- Select a **Credential*** to connect to the SCP server. Only credentials on the `SCP/SFTP` target can be selected.
- Choose a **Timeout** for the SCP request
- Choose whether to **Use compression** when pushing the CRL
- Enter a known **Fingerprint** to use mutual authentication. If nothing is specified, no fingerprint check will occur.
- Define the **Path*** where to push the CRL. Using template strings, this path can be dynamically set. For example, if the crl should be pushed to the `crls` root folder with a filename being an uppercase of the CA's CN with the `.pem` extension, path will be `/crls/{{ Upper({{ca.signer.dn.cn.1}}) }}.pem`

4. Once you've filled all the information, click "Add"

The External CRL Storage is now created and can be used in CA configuration.

3.5.5. Creating an SFTP External Storage

Stream allows you to push your CRLs into any server supporting the SFTP protocol upon generation. This section also assumes that you have already configured `SSH` credentials for the desired server.

To configure an external SFTP CRL storage:

1. Log in to the Stream Administration Interface ;

2. Go to **Revocation management › External CRL Storage** and click on  ;

3. Fill in the information :

- Select the **Type*** of external CRL storage, SFTP for an SFTP storage
- The **Name*** to give to that external storage
- The **Description** to add more details about this storage
- Select a list of notifications to send **On execution error** to be alerted if the push to the CRL storage fails
- Add the **Host***, IP or hostname of the SFTP server where the CRL will be pushed into.
- Add the **Port*** on which the SFTP server is running (default is 22 for SSH)
- Select a **Credential*** to connect to the SFTP server. Only credentials on the `SCP/SFTP` target can be selected.
- Choose a **Timeout** for the SFTP request
- Choose whether to **Use compression** when pushing the CRL

- Enter a known **Fingerprint** to use mutual authentication. If nothing is specified, no fingerprint check will occur.
- Define the **Path*** where to push the CRL. Using template strings, this path can be dynamically set. For example, if the crl should be pushed to the `crls` root folder with a filename being an uppercase of the CA's CN with the `.pem` extension, path will be `/crls/{{ Upper({{ca.signer.dn.cn.1}}) }}.pem`

4. Once you've filled all the information, click "Add"

The External CRL Storage is now created and can be used in CA configuration.

3.6. Configuring OCSP

To configure an **OCSP** responder, you first need an OCSP signer.

1. Log in to the Stream Administration Interface.

2. Go to **Revocation Management** › **OCSP Signers** and click on ⊕ at the bottom of the page.

3. Fill in the fields to create an OCSP signer that will sign OCSP requests:

- The **Name** of the OCSP signer: a technical name to identify this signer.
- The **Keystore** where to find the key for this signer.
- The **Key** that this signer will sign with.
- The **DN** of this signer, in X500 format with key=value separated by commas.
- The **Notification on signer expiration** that will notify users via Email or REST.

4. You must then generate the CSR ↴, sign it using the CA you wish to verify certificate for, and upload the signed certificate back to Stream ↥

CAUTION	The certificate must be signed with the Key Usage `digitalSignature` (critical) and the Extended Key Usage `OCSPSigning`

5. The OCSP Signer is now uploaded. Additional options are now available:

- The **Response Signing Algorithm**, the hash algorithm that wil be used on responses signed by this signer

6. Click the **Save** button at the bottom of the page.

Now your OCSP signer has been configured, OCSP must be enabled on a **Certification Authority**:

7. Go to **Certification Authorities**:

- **Managed CAs**, in the **CRL/OCSP** tab
- **External CAs**, in the **Configuration** tab

8. Toggle the **Enable OCSP** option. New options appear:

- **Compromised CA?** can be toggled if the CA was compromised to make all certificates on this CA act as revoked
- The **Default OCSP signer** to use if no explicit signer is defined in the OCSP request
- The **Archive Cutoff mode** to use on OCSP responses:
  - **Issuer**: the archive cutoff date will be this CA emission date
  - **Retention**: the archive cutoff date will be the OCSP request date plus the retention period

9 Click the **Save** button at the top.

4. Managing Keystores & Keys

4.1. Keystores in Stream

In Stream, keys are grouped in key containers called **Keystores**.

Stream handles 3 types of Keystores: Software keystores, PKCS#11 HSMs and Cloud KMS. Note that some restrictions apply regarding the supported key types of the HSMs, namely:

- The software keystore supports:
  - RSA key sizes above 512 bits (the web administration console only offers RSA 2048, RSA 3072, RSA 4096 and RSA 8192);
  - 3 elliptic curves: ECC NIST P-256, ECC NIST P-384 and ECC NIST P-521;
  - 2 Edward curves: ED-448 and ED-25519;
- The PKCS#11 keystore crypto capabilities are entirely reliant on the HSM that is used. Generally, RSA keys are all supported, while elliptic curves are not all supported by every HSM vendor. Currently, Edward curves are also not supported by some HSM vendors;
- Stream can consume the following key types from an AWS KMS instance:
  - RSA 2048, RSA 3072, RSA 4096;
  - ECC NIST P-256, ECC NIST P-384, ECC NIST P-521;
  - The AWS KMS currently does not support Edward Curves;
  - Stream currently does not support the ECC SECG P-256k1;
- Stream can consume the following key types from an AKV instance:
  - RSA 2048, RSA 3072, RSA 4096;
  - ECC NIST P-256, ECC NIST P-384, ECC NIST P-521;
  - Azure Key Vaults (even the Premium ones) currently do not support Edward Curves;
  - Stream currently does not support the ECC SECG P-256k1;
- Stream can consume the following key types from a GCP CKM instance:
  - RSA 2048, RSA 3072, RSA 4096;
  - ECC NIST P-256 and ECC NIST P-384;
  - The GCP CKM currently does not support Edward Curves.

4.2. Software keystore

Stream comes installed with a software keystore that can be used to generate RSA and ECDSA keys. To set up a software keystore:

1. Log in to the Stream Administration Interface.

2. Go to **Keystores and keys** and click  .

3. In **Type**, select **Software**. In **Name**, set the name you want to give to your keystore. Optionally, you can add a description to your keystore.

4. Click the **Add** button.

Your keystore should appear in your keystores list with a green circle next to its name.

NOTE	When using the software keystore, private keys are at some point stored in memory in **plain text**. That represents a huge security flaw since it would just take a memory dump of the Stream machine to be able to recover the private keys.

CAUTION	It is **not recommended** to use the software keystore except for testing or development purposes due to the safety reasons detailed above.

4.3. PKCS#11 HSM

Stream supports key management through **PKCS#11 HSMs**.

Stream has been qualified to work with the following **HSMs** but should be working with any **PKCS#11 HSM**:

- Entrust nShield Solo, Entrust nShield Connect, Entrust nShield as a Service
- Atos Proteccio
- Thales Luna (including DPoD), Thales Protect Server
- Utimaco CryptoServer

To set up a PKCS#11 keystore:

1. Log in to the Stream Administration Interface.

2. Go to **Keystores and keys** and click  .

3. In **Type**, select **PKCS#11**. In **Name**, set the name you want to give to your keystore. Optionally, you can add a description to your keystore.

4. Input the **full path** of the **PKCS#11 library** (ending in *.so*) of your HSM, then click the parse button. If your HSM's library was succesfully loaded into Stream, you should be seeing your HSM's information. If you get an HSM error, please check the configuration of your HSM. Click "Next".

5. Select the **HSM slot** that you will be using on your HSM for this keystore and input its **PIN code**;

6. Optionally, you can set a Pool Size to your PKCS#11 interface. If disabled, Stream will open a PKCS#11 session every time it needs to sign a certificate, then close it afterwards. If enabled, Stream will open the number of connections specified in the pool size value and maintain them open as long as Stream is running, to be able to directly sign certificates without having to open a PKCS#11 session. This feature comes particularly handy whenever working with a slow HSM, where opening a session is a pretty long operation that can completely ruin performance.

Once you are done, click "Save". Your keystore should appear in your keystores list with a green circle next to its name.

4.4. Cloud KMS

Stream supports 3 types of Cloud KMS: Google Cloud Platform (GCP), AWS Key Management Service (KMS) and Microsoft Azure Key Vault (AKV).

4.4.1. Setting up a Google Cloud Key Management (GCP CKM)

1. Log in to the Stream Administration Interface.

2. Go to **Keystores and keys** and click  .

3. In **Type**, select **Google Cloud Platform**. In **Name**, set the name you want to give to your keystore. Optionally, you can add a description to your keystore.

4. Select the GCP credential to use to connect to the Cloud Key Management service. If you do not have your GCP CKM credentials set up in Stream yet, please refer to the *Credentials* part of the ***Managing Security*** section.

5. Input the **GCP Project name** in **Project**, the **GCP Server location** to use and the **GCP Key Ring** to use. Additionally, you can specify the **proxy** to use as well as the **timeout period**. Once you are done, click "Add".

Your keystore should appear in your keystores list with a green circle next to its name.

4.4.2. Setting up an AWS Key Management Service (AWS KMS)

1. Log in to the Stream Administration Interface.

2. Go to **Keystores and keys** and click  .

3. In **Type**, select **AWS**. In **Name**, set the name you want to give to your keystore. Optionally, you can add a description to your keystore.

4. Select the AWS credential to use to connect to the AWS Key Management Service. If you do not have your AWS KMS credentials set up in Stream yet, please refer to the *Credentials* part of the ***Managing Security*** section.

5. Input the **AWS server's region** in **AWS Region**. Optionally, you can specify which AWS Role ARN that should be impersonated for that KMS. Additionally, you can specify the **proxy** to use as well as the **timeout period**. Once you are done, click "Add".

NOTE	To make Stream able to use the keys in the AWS KMS for signature, you need to give it the proper permissions in the AWS console. For more information regarding this topic, please refer to this link, under the "Asymmetric KMS keys for signing and verification".

Your keystore should appear in your keystores list with a green circle next to its name.

4.4.3. Microsoft Azure Key Vault (AKV)

1. Log in to the Stream Administration Interface.

2. Go to **Keystores and keys** and click  .

3. In **Type**, select **Azure Key Vault**. In **Name**, set the name you want to give to your keystore. Optionally, you can add a description to your keystore.

4. Select the AKV credential to use to connect to the Microsoft Azure Key Vault. If you do not have your Microsoft AKV credentials set up in Stream yet, please refer to the *Credentials* part of the ***Managing Security*** section.

5. Specify your Azure vault URL in the **Vault URL** box and the Azure tenant in the **Azure Tenant** box. Additionally, you can specify the **proxy** to use as well as the **timeout period**. Once you are done, click "Add".

Your keystore should appear in your keystores list with a green circle next to its name.

4.5. Managing keys in Stream

Regardless of the type of keystores you set up, you can manage the keys through Stream the same way

4.5.1. Adding a key into a keystore

1. Log in to the Stream Administration Interface.

2. Go to **Keystores and keys** and click + on the keystore you want to add the key into.

3. Set the name of the key as well as the key type (RSA, ECDSA or EDDSA) and the key size (for RSA)/key parameter (for ECDSA/EDDSA).

4. For the **Cloud KMSs**, you can set the key to be **Hardware protected** through the dedicated toggle. For the **PKCS#11 HSM**, you can set the key to be **exportable** through the dedicated toggle.

5. Once you set up the key parameters as you want them, click "Add".

The page should refresh and show you the list of keys for the keystore you pushed the key into, where you should see the key you just added.

4.5.2. Viewing the keys of a keystore

1. Log in to the Stream Administration Interface.

2. Go to **Keystores and keys** and click ⌄ on the keystore you want to view.

3. You should see the list of keys on your keystore.

You can then see information about the keys in the keystore:

- The **name** column where you can see the name of the key ;
- The **type** column where you can see the type of algorithm that was used to generate the key. Both RSA and ECDSA are part of the *suiteb* type algorithms ;
- The **key type** column where you can see the algorithm that was used to generate the key as well as the key size/parameter ;
- The **exportable** column indicates if the key is exportable or not.

4.5.3. Deleting a key from a keystore

1. Log in to the Stream Administration Interface.

2. Go to **Keystores and keys** and click ⌄ on the keystore you want to delete the key from.

3. Click the 🗑 icon on the key that you want to delete and click "Confirm" on the prompt.

NOTE	You cannot delete a key from a keystore if this key is currently used by a CA in Stream. You must first delete the CA that references it and then go over the deleting procedure.

5. Managing Notifications

5.1. Email

This section details how to configure the email notifications.

5.1.1. How to create an email notification

1. Log in to Stream Administration Interface.

2. Access emails from the drawer or card: **Notifications** › **Emails**.

3. Click on  .

4. Fill in all mandatory fields in the Notifications details panel.

- The **Name** of the notification.
- The **Lifecycle event** triggering the notification.
  - First, select the entity **Type** can be one of:
    - CA: events on the Certification Authority lifecycle, like expiration of a CA
    - CRL: events on the CRL lifecycle, like a generation or update
    - OCSP Signer: events on the OCSP Signers lifecycle, like expiration of an OCSP signer
    - System: events concerning Stream global objects, like credentials or license expiration
    - Timestamping Signer: events on the Timestamping Signers lifecycle, like expiration of a Timestamping signer
  - Then, select the **Event** you wish to send the notification on.
- Select the notifications to send **On execution error** of this Notification to be alerted if it failed.
- The **Delay before notification sending** is only enabled when the **Event** is about the expiration of an entity. This is the period where the notification will be sent before the expiration date.

5. Fill in all mandatory fields in the Notifications Email details panel.

- The email sender **From** which the mail will be sent.
- The email targets **To** send email.
- The email **Subject**, a template string that will be dynamically evaluated upon email generation.
- The email **Body**, a template string that will be dynamically evaluated upon email generation.
- Set whether the email body **is HTML**. The default value is set to false.

NOTE	You can click on the ⌄ next to the "Dynamic attributes" section, in order to get a range of possibilities on which dictionaries and functions are available.

6. Click on the save button.

You can edit , duplicate or delete the Email Notification.

5.2. REST

This section details how to configure REST notifications.

5.2.1. How to create a REST notification

1. Log in to Horizon Administration Interface.

2. Access REST from the drawer or card: **Notifications** › **REST**.

3. Click on  .

4. Fill in all mandatory fields in the Notifications details panel.

- The **Name** of the notification.
- The **Lifecycle event** triggering the notification.
  - First, select the entity **Type** can be one of:
    - CA: events on the Certification Authority lifecycle, like expiration of a CA
    - CRL: events on the CRL lifecycle, like a generation or update
    - OCSP Signer: events on the OCSP Signers lifecycle, like expiration of an OCSP signer
    - System: events concerning Stream global objects, like credentials or license expiration
    - Timestamping Signer: events on the Timestamping Signers lifecycle, like expiration of a Timestamping signer
  - Then, select the **Event** you wish to send the notification on.
- Select the notifications to send **On execution error** of this Notification to be alerted if it failed.
- The **Delay before notification sending** is only enabled when the **Event** is about the expiration of an entity. This is the period where the notification will be sent before the expiration date.

5. Fill in all mandatory fields in the Notifications Rest details panel.

- The REST **HTTP method and URL** that the HTTP request will use
- The **Proxy** used by Stream to send the HTTP request.
- The **Timeout** to wait before stopping listening to an answer.
- The **Accepted response HTTP code(s)** to consider the request in success state.
- The **Authentication type and credentials** used by the HTTP request, for example, the basic authentication.
- The **Headers** sent along with the HTTP request. Each header value is a template string that will be dynamically evaluated when sending the notification.
- The **Payload** of the HTTP request. It is a template string that will be dynamically evaluated when sending the notification.

NOTE	You can click on the ⌄ next to the "Dynamic attributes" section, in order to get a

> range of possibilities on which dictionaries and functions are available.

6. Click on the save button.

You can edit , duplicate or delete the Email Notification.

6. Managing Certificate Templates & EKUs

6.1. Certificate Templates

Stream uses the notion of **Certificate Templates** to add additional verifications when enrolling a certificate.

To define a new certificate template:

1. Log in to the Stream Administration Interface.

2. Go to **Certificates** > **Templates** and click  .

3. In the **General** tab, you can set the template's name, the **path length** it will tolerate, turn the template on or off and check for proof of possession when enrolling with a CSR. In the **Duration** part of the tab, you can edit the lifetime of the certificates that will enroll on this template, as well as backdate them should you need to. In the **Private Key policy** part of the tab, you can choose whether to enforce a usage period for the private key that is detached from the validity of the certificate. Should it be defined, this period must be within the validity period of the certificate. This field is optional in the RFC 5280 but mandatory in the ICAO MRTD 9303 norm (section 7.1.1) and should only be used for signature certificates.

4. In the **KU & EKU** tab, you can set the **Key Usages** and **Extended Key Usages** of the certificates that will enroll on this template. You can also use your own **EKUs** here. If you want to set up your own **EKUs**, please refer to the *Extended Key Usages* part of this section.

5. In the **Extensions** tab, you can edit the **CRLDPs**, **AIA**, **Authority Information Access**, **Policy**, **Qualified Certificate Statement** of the certificates that will enroll on this template. If you want to, the certificates could use the information of the **CA** they will enroll on, otherwise, you can set specific values in the template. These values will then override those retrieved from the CA.

- If you want to issue Qualified Certificates:
  - ETSI QC Compliance Statement declares that the certificates is a Qualified Certificate.
  - ETSI QC SSCD Statement declares that the private key related to the certified public key resides in a Secure Signature Creation Device.
  - ETSI Retention Period Statement indicates the duration of the retention period of material information.
  - ETSI QC Type Statement indicates which type of document can be signed by the certificate (possible values are: ESEAL, ESIGN, WEB, NONE).
  - ETSI Transaction Limit Statement indicates the limits of the transactions, you must fill every field if enabled.
  - ETSI QC PDS Statement is the PKI Disclosure Statement URI for a specified language.
  - ETSI QC Legislation Statement is an array of country codes.

6. In the **Data Fields** tab, you can enforce your **DNs**, **SANs** and **Extensions** to match certain criteria

that can be defined in this section. By default, everything is accepted, meaning that any type and amount of **DNs**, **SANs** and **Extensions** can be used in the certificates and it would successfully enroll on the template.

- If you want to enforce a **Subject DN** policy, then click  in **Subject DN composition**, then select the DN element that you want to put a policy on. You can set this element to be mandatory or not, to use a default value for that element that can be editable or not, you can also add a whitelist of elements that are accepted values for this DN, or you can instead use a regex to match the DN values that are accepted for this element.
- If you want to enforce a **Subject Alternate Names** policy, you can either click **None** to forbid the use of **SANs** in certificates or you can click **Some** to configure the policy. If you clicked **Some**, click  and select the **SAN element** that you want to enforce a policy upon. You can then input a minimum and maximum number of this **SAN element** to be present in the certificate that will enroll: as an example, if you want to make the use of at least one **DNS SAN** mandatory, use 1 as a minimum number. Finally, you can enforce your **SANs** to match a regex to be considered valid on a certificate.
- If you want to enforce an **Extension** policy, you can either click **None** to forbid the use of **Extensions** in certificates or you can click **Some** to configure the policy. If you clicked **Some**, click  and select the **Extension** that you want to enforce a policy upon. You can then set it mandatory or not, and if supported, give it a default value that can be edited or not.

7. Once you've configured your template, you can click **Save** at the top of the page.

NOTE	As mentioned previously, if you want your certificates to inherit the **CRLDP**, the **AIA** and the **Policy** from the CA, you must toggle on the **Get from CA** swiches and not specify any policy, CRLDP or AIA in the template.

6.2. Extended Key Usage

Stream allows you to create and manage your own EKUs as long as you have an OID for it.

To create a custom EKU:

1. Log in to the Stream Administration Interface.

2. Go to **Certificates** > **EKU** then go at the bottom of the page and click  .

3. Specify the name you want to give to your custom EKU as well as its OID in the menu, then click "Add".

The EKU should show in the list with the custom switch turned on, as opposed to the standard EKUs that have the custom switch turned off.

7. Managing Security

7.1. Authorizations

This section details how to configure the permissions granted to an account, either directly or through a configured role.

7.1.1. Prerequisites

According to the context, you might need to set up:

- Roles
- Local accounts

7.1.2. How to add an authorization manually or from a certificate

1. Log in to Stream Administration Interface.

2. Access Authorizations from the drawer or card: **Security** › **Authorizations**.

3. Click on  .

4. Click on  Add Authorization Manually

5. Fill the mandatory fields:

- Either:
  - Fill in an **Identifier***: it can be either a local account identifier or an OpenID Connect identifier (usually email address).
  - Import a certificate by clicking on certificate button  .

6. Click on add button.

7.1.3. How to add an authorization from a search

1. Log in to Stream Administration Interface.

2. Access Authorizations from the drawer or card: **Security** › **Authorizations**.

3. Click on  .

4. Click on  Search and Add Authorization

5. Search by **Identifier** for a local account previously defined.

6. Click on search button.

7. Choose the identifier you want to add.

8. Click on add button.

You can update , see connexion information , or delete Authorization.

7.1.4. How to grant a permission

1. Click on .

Role

2. Select a role previously created (if needed).

Permissions

Stream allows you to manage 2 types of permissions: configuration and lifecycle.

Stream uses wildcard permissions which means you can configure the permissions very thoroughly.

Configuration

For configuration permissions, you can specify:

- the **Section** (ex: Security)
- the concerned **Module** (only for select modules)
- the type of permission: **Audit** (read-only) or **Manage** (read-write, equivalent to *All*).

4. Click on add button.

5. Select a section, then a module, then a submodule if there is, and a right.

6. Click on add button (Don’t forget to save).

7. Click on the save button if you are done.

Lifecycle

For lifecycle permissions, you can specify the concerned **CA** and the concerned **Template** then the type of permission: **Enroll**, **Revoke**, **Search** or **All** of these.

4. Click on add button.

5. Select a module, then a profile, and a right.

6. Click on add button. (don’t forget to save).

7. Click on the save button if you are done.

7.2. Credentials

This section details how to configure credentials. Credentials are where credentials for all integrations are regrouped.

7.2.1. How to create credentials

1. Log in to Stream Administration Interface.

2. Access Credentials from the drawer or card: **Security › Credentials**.

3. Click on ⊕.

4. Fill the fields.

- The **Type***: `Certificate` for certificate based authentication, `Login` for login with password credentials, `API Token` for a single value secret (JSON or other) or SSH for SSH Keys secret.
- The **Name*** of the credentials. It should clearly identify it.
- The **Description** to add additional information on these credentials.
- The **Expiration date**. This will be taken from the certificate for `Certificate` credentials, and will be used for notifications on expiration.
- The **Expiration notifications** are Email or REST notifications on event `Credentials expiration` that will run on expiration. Notifications configured here will be sent by the internal monitoring action.
- Certificate:
  - The **PKCS#12*** file containing the authentication certificate and its key.
  - The **PKCS#12 Password*** to open this PKCS#12.
- Credentials:
  - The **Login*** of the account.
  - The **Password*** of the account.
- API Token:
  - The **API Token*** to use.
- SSH:
  - The **SSH identifier***: username to use for SSH connection.
  - The **SSH key*** is the SSH private key in OpenSSH format for SSH connection.

5. Click on the save button.

You can update ✎ or delete 🗑 the Credentials.

7.3. Identity Providers

7.3.1. How to configure an Identity Provider

1. Log in to Stream Administration Interface.

2. Access Identity Providers from the drawer or card: **Security › Access Management › Identity Providers**.

3. Click on  .

General tab

4. Select an identity provider type. Currently only OpenID is supported

OpenID connect

5. Fill in all fields:

- The **Name*** will be used to identify this provider on Stream and on the login page.
- **Enabled*** allows to disable the identity provider when access from this authentication source is not needed.
- **Enabled on `UI`*** allows to hide this provider on the login page, but it will still be available via direct API calls.
- The **Provider metadata URL*** is the url where the OIDC provider provides its metadata. For example `https://<oidc server>/.well-known/openid-configuration`.
- The **Client Credentials*** are `Password` credentials containing the client id and secret used to connect to the OIDC provider. They can be created on the go using the `+`.
- The **Scope*** used by Stream during authentication on the identity provider to authorize access to user's details.
- The **Proxy** used to access Provider metadata URL, if any.
- The **Timeout** used for authentication on the identity provider. Must be a valid finite duration. The default value is 10 seconds.
- The **Identifier Claim*** is a template string defining how to construct the identifier from the OpenID Connect claims. For example, if the user identifier is contained in the `login` claim, and should be lower case, then the configured value should be `{{Lower({{login}})}}`.
- The **Name Claim*** is a template string defining how to construct the user name from the OpenID Connect claims. For example, if the user name must be constructed as `family name, GIVEN NAME` and family name is available in the `family_name` claim, given name is available in the `given_name` claim, then the configured value should be `{{family_name}}, {{Upper({{given_name}})}}`

6. Click on the save button.

You can update  or delete  the Identity Provider.

CAUTION	You won't be able to delete an Identity Provider if it is referenced in any other configuration element.

7.4. Local Accounts

7.4.1. How to create local accounts

1. Log in to Stream Administration Interface.

2. Access Local accounts from the drawer or card: **Security** › **Access Management** › **Local Accounts**.

3. Click on  .

4. Fill in the fields:

- The **Identifier***, a meaningful identifier for the account holder. It will be used as a login to access to the solution.
- The **Name*** for the account holder. It will be displayed on the interface when logged in.

5. Click on the create button. The account is created and a password is generated.

7.4.2. How to reset a password on a local account

1. Once a local account is created. Click on  .

You can edit  or delete  a local account. You can reset  a local account password.

NOTE	You can not delete yourself from local accounts.

7.5. Roles

Roles are a way to factor permissions making it easier to configure accounts and track permissions.

7.5.1. Creating a new role

1. Log in to the Stream Administration Interface.

2. Go to **Security** > **Roles** and click  ;

3. Set the **name** of the role you want to create.Optionally, you can add a **description** to the role.

4. Add the **configuration permissions** you want the members of this role to have using the  from **Configuration permissions**. If the role is supposed to have no configuration permission, leave this section empty.

5. Add the **lifecycle permissions** you want the members of this role to have using the + from **Lifecycle permissions**. If the role is supposed to have no lifecycle permission, leave this section empty.

Once everything is set up, you can click **Save**.

7.5.2. Managing roles

1. Log in to the Stream Administration Interface.

2. Go to **Security** > **Accounts**. From there, you can see all Stream roles and their associated information.

- The **name** column displays the role's name;
- The **description** column displays the role's description;
- The **permissions** column shows the **straight permissions** that are set up for the role. If the account has any **configuration** permission, it will display ⚙ and if it has any lifecycle permission it will display ↻;
- You can **view all the members of a role** using the ☰ button;
- You can **delete a role** using the 🗑 button;
- You can **edit a role**'s information using the ✎ button.

7.6. Enforce Certificate Authentication

It is possible to enable x509_enforcing parameter in order to authorize only certificate authentication.

WARNING	This means local accounts will no longer be able to connect on Stream.
CAUTION	When logging in using an X509 certificate, there is no logout option, meaning that the only way to log out is to change the presented certificate in your browser, or to switch to private browsing.

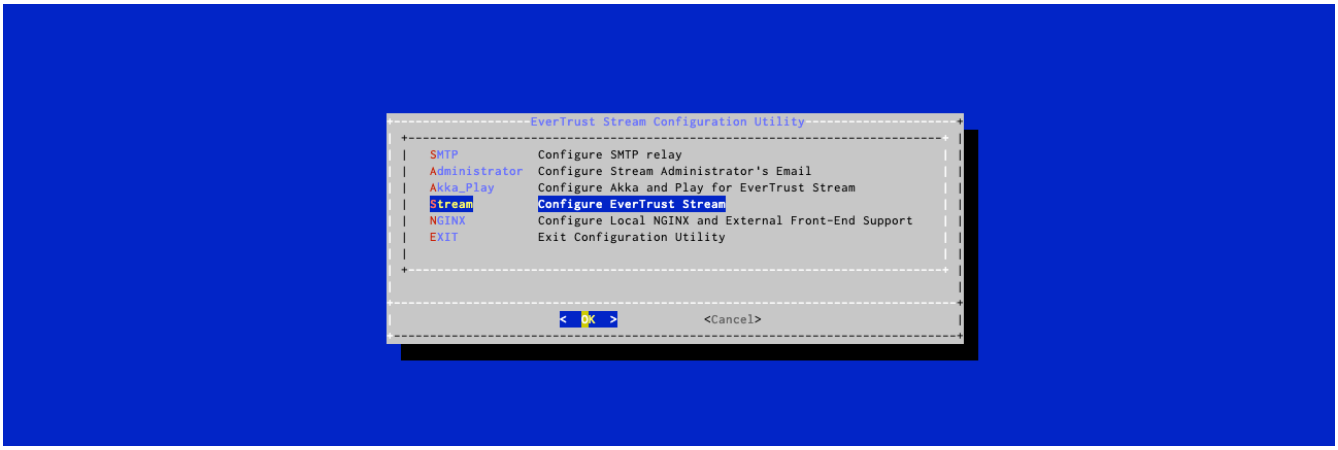
7.6.1. Using Stream configuration utility

Connect to the server with an account with administrative privileges;

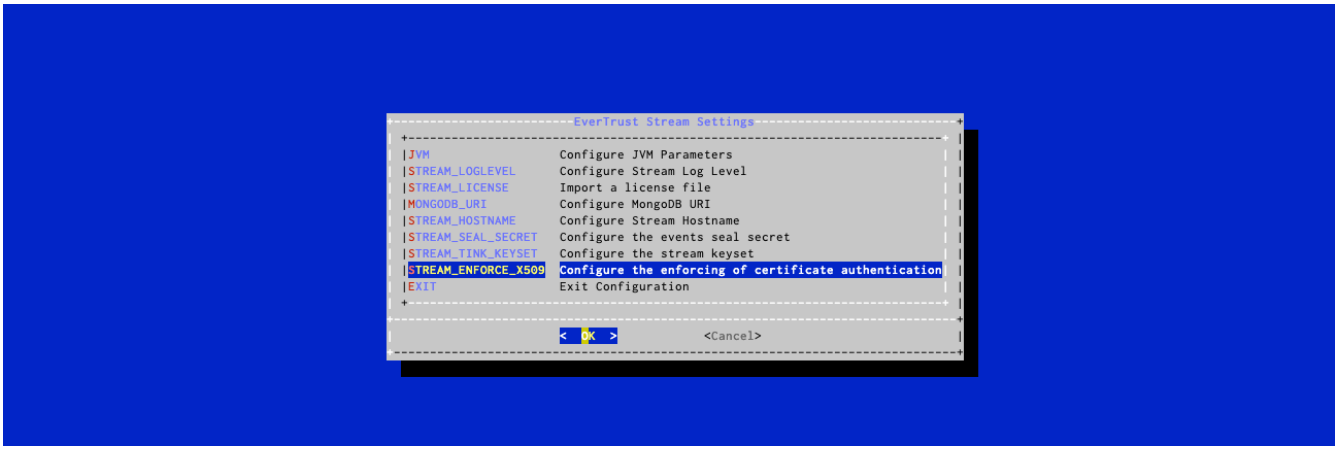
Start the Stream configuration utility by running:

```
# /opt/stream/sbin/stream-config
```

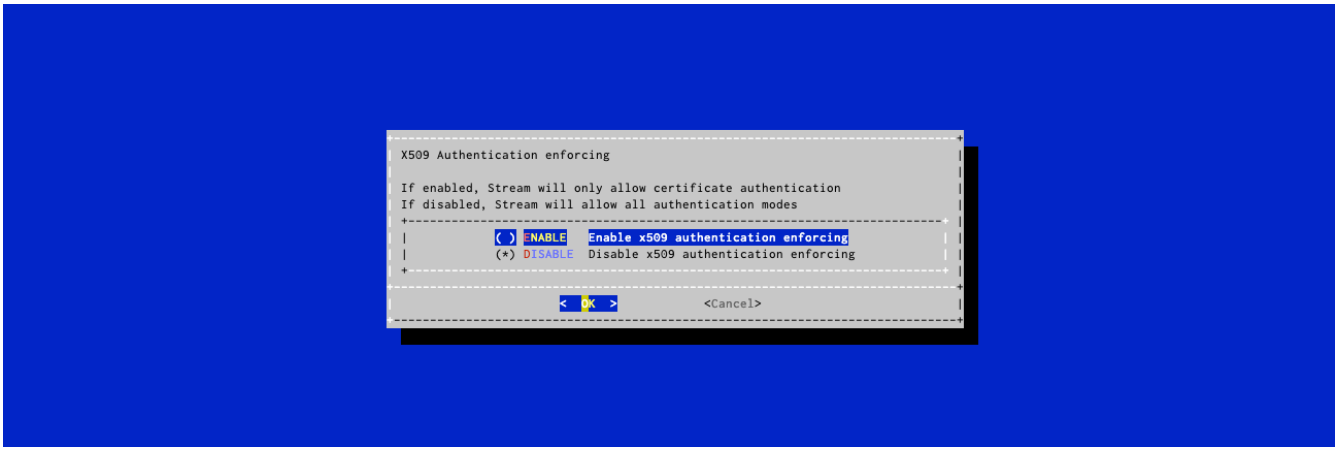
In the main menu, select '**Stream**':



In the Stream menu, select '**STREAM_ENFORCE_X509**':



In the X509 Authentication Enforcing menu, select '**ENABLE**':



For the changes to take effect, you must restart the Stream service by running:

```
# systemctl restart stream
```

X509 Authentication is now enforced.

7.6.2. Re-enable local authentication

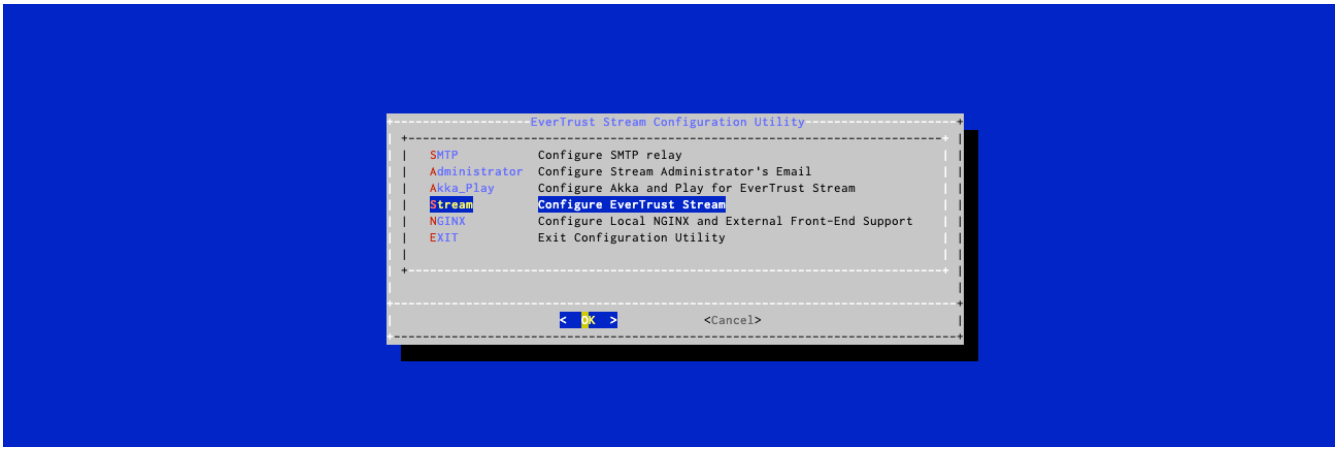
CAUTION	This should be done in a confined and secure environment.

If you lose all available authentication certificates to Stream and want to re-gain access to the administration console, please follow these steps:

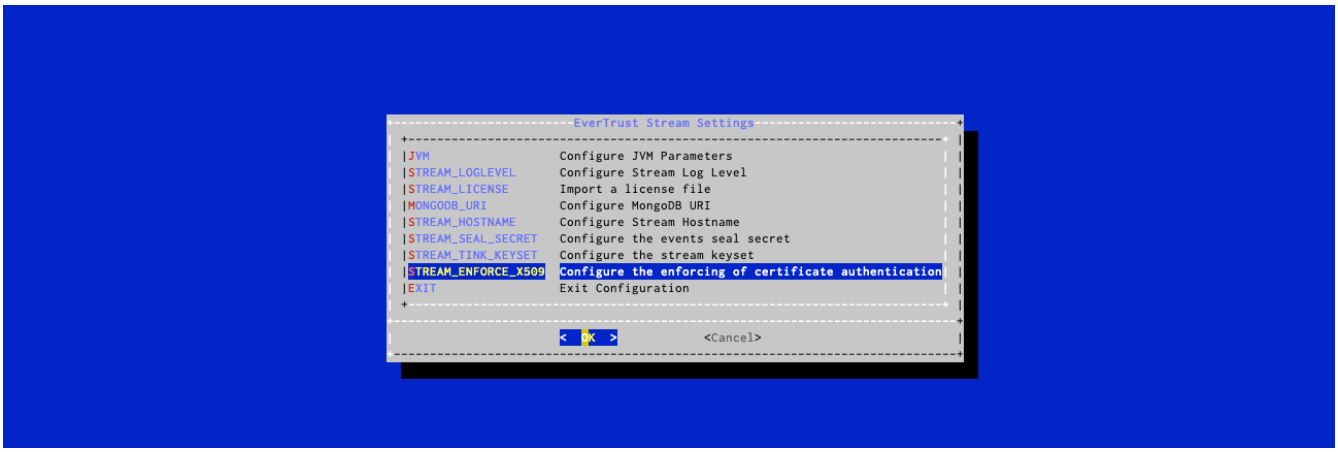
Connect to the server with an account with administrative privileges;

Start the Stream configuration utility by running:

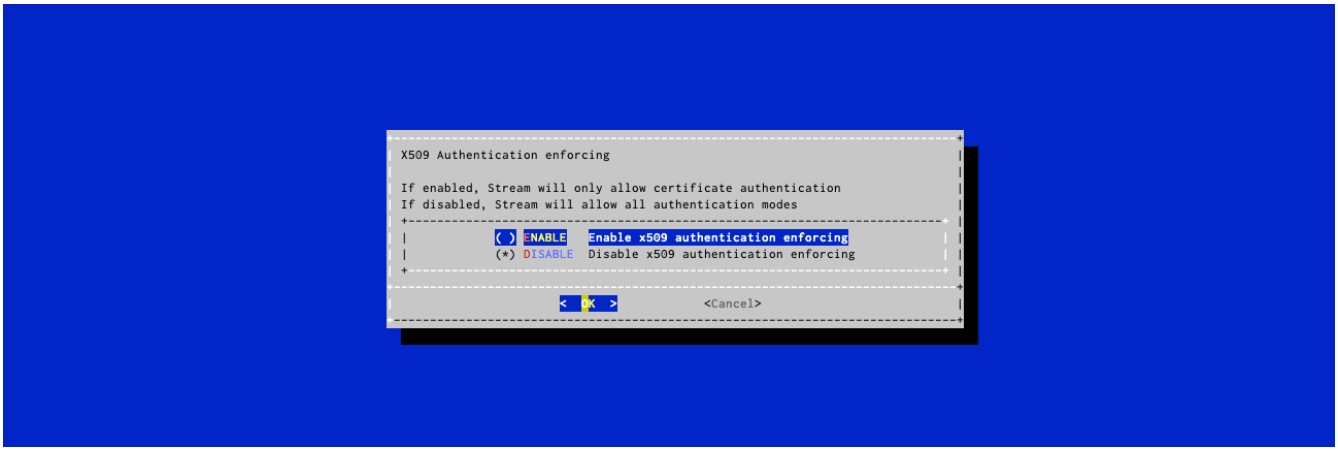
```
# /opt/stream/sbin/stream-config
```



In the Stream menu, select '**STREAM_ENFORCE_X509**':



In the X509 Authentication Enforcing menu, select '**DISABLE**':



For the changes to take effect, you must restart the Stream service by running:

```
# systemctl restart stream
```

Now that the X509 enforcing is disabled, you can log in with the initial administrator account that was created during the bootstrap of the product. If you lost access to that account as well, or if you deleted it, please contact the EVERTRUST support.

8. Managing Stream instance

8.1. Events

The event system exists to overview the actions happening on Stream.

By default, the events are chained by the following rule: event `n` references event `n-1`. They are signed with the event seal secret set up during the stream installation.

To consult them:

1. Log in to the Stream Administration Interface.

2. Go to **System** > **Events**.

8.1.1. Event integrity reports

To check the integrity of the events, you can run an event integrity report:

1. Log in to the Stream Administration Interface.

2. Go to **System** > **Events Integrity Reports**.

3. Click  ;

4. Click `Run`

The integrity of the event chain is checked and can take some time depending on the number of events in the database. Once finished, the report may have different status:

- `Running`: the integrity of the events is currently being checked.
- `Verified`: the event chain is not compromised.
- `Report integrity failure`: the report signature has been compromised.
- `Event integrity failure`: the event chain has been compromised, one event could have been modified or deleted. The event integrity report error provides details about the cause of the integrity failure.

WARNING	Any compromised object means an account with enough permission to write in the database has been compromised.

8.1.2. Purging/Backup event database

NOTE	Manual actions regarding the events manipulation should be done with stream turned off and in a confined environment.

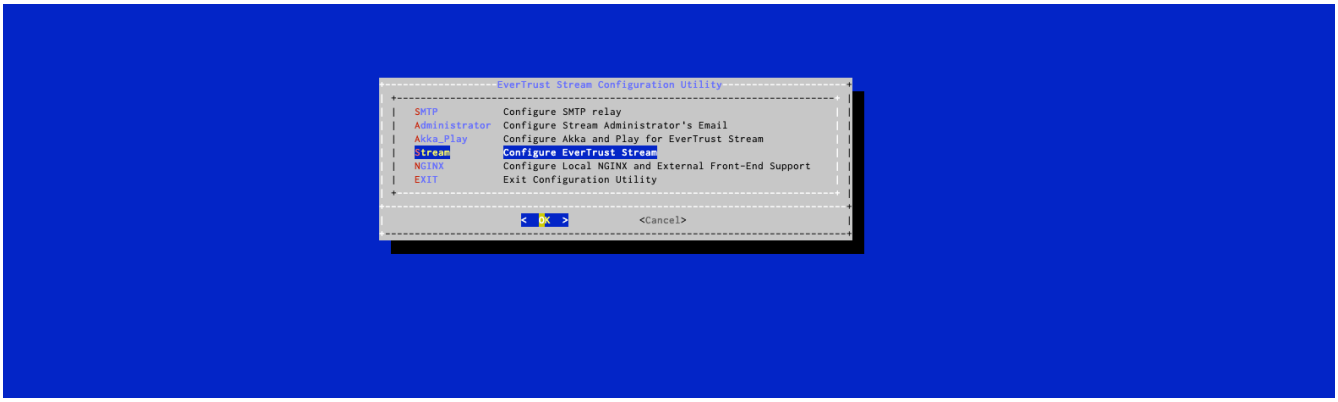
Follow the Backup guide to save your database. Once done, you might want to delete the events in databse.

Deletion of events can only be made from the oldest to the newest since events are chained. For example, you might want to delete every event before a date:

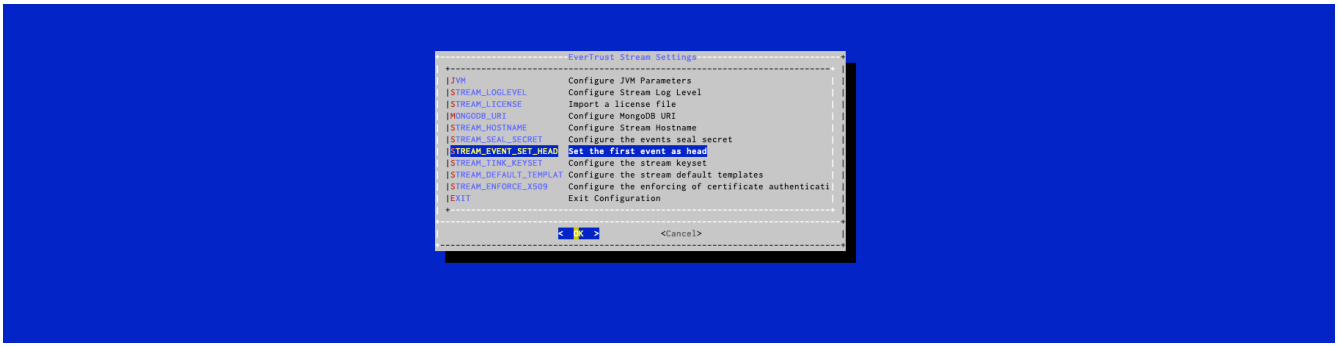
```
use stream;
db.events.deleteMany({"timestamp":{$lt: ISODate("2023-09-20")}});
```

After the deletion of events, the Head is still chained to a deleted event. In order to fix that, you will need to run the `Set the first event as head` in `/opt/stream/sbin/stream-config`:

In the main menu, select '**Stream**':



In the Stream menu, select '**STREAM_EVENT_SET_HEAD**':



8.1.3. Integrity compromised

If an event or event integrity report has been compromised, it means that someone had database access to Stream or one of its backups and manually edited the events to hide specific actions.

You should close all network access to the server and, if necessary, turn off stream. Once confined, you should follow these steps:

1. Follow the Backup guide to back up your database. It may be used to investigate the problem.

2. Analyze the logs (you may use an older verified backup to assess modifications).

NOTE	Since the database has been compromised, every event should be considered as a non trusted information

3. Based on your assessments, take the appropriate actions. This could mean changing the mongodb

password, changing the server password, revoking stream access certificates or other actions.

4. To resume a normal state, remove every corrupted event following the steps in the event purge guide.

8.2. Events

All the events displayed in this document work in a similar manner. In case of a failure, the event will display the reason of said failure. This behavior is also valid for warning-status events.

8.2.1. BOOTSTRAP

Bootstrap events relate to the initial setup of the Stream platform.

- **BOOTSTRAP-ADMINISTRATOR-ACCOUNT**
  This event is triggered when installing Stream, it corresponds to the creation of the administrator local identity on Stream.
- **BOOTSTRAP-ADMINISTRATOR-PRINCIPAL**
  This event is triggered when installing Stream, it corresponds to the creation of a link between the administrator account and its rights.
- **BOOTSTRAP-LOCAL-IDENTITY-PROVIDER**
  This event is triggered when installing Stream, it corresponds to the creation of a provider of type Local so that the administrator can connect after startup.
- **BOOTSTRAP-SYSTEM-CONFIGURATION**
  This event is triggered when installing Stream, it corresponds to the creation of internal configuration elements such as the CRON internal monitor.

8.2.2. CA

- **CA-CRL-GEN**
  This event occurs on a CRL Generation request on a CA.
- **CA-CRL-UPLOAD**
  This event occurs when a CRL is being uploaded on a CA.
- **CA-CSR**
  This event occurs when a CSR generation is requested on a CA. This is commonly part of the CA issuing process.
- **CA-ENHANCE**
  This event occurs when a legacy CA is being enhanced to a PQC-ready CA.
- **CA-ISSUE**
  This event occurs when a CA is being issued.
- **CA-KRL-GEN**
  This event occurs on a KRL Generation request on a CA.
- **CA-MIGRATE**
  This event occurs when an external CA is being migrated to a managed CA.

- **CA-REVOKE**
  This event occurs on a CA revocation attempt.

8.2.3. CONF

CONF events are triggered when users interact with configuration elements. This includes certificate templates, notification triggers, Certification Authorities…

- **CONF-ADD**
  This event is triggered when a user tries to add a configuration element.
- **CONF-DELETE**
  This event is triggered when a user tries to delete a configuration element.
- **CONF-UPDATE**
  This event occurs when a user tries to modify a configuration element.

8.2.4. CRL

- **CRL-GEN**
  This event occurs on a CRL generation attempt, either requested by application processes or the user.
- **CRL-GET**
  This event occurs on a CRL retrieval attempt from a CRLDP. These are attempted by the application.
- **CRL-SYNC**
  This event is triggered when a failure occurs on a CRL Synchronization.
- **CRL-UPLOAD**
  This event occurs when a user tries to upload a new CRL on a CA.

8.2.5. EVENT COMPLIANCE

- **INVALID-SEAL-PENDING-EVENT**
  This event occurs when a pending event has an invalid seal (indicating data corruption in the pending events collection).
- **UNSEALED-PENDING-EVENT**
  This event occurs when a pending event has no seal (indicating data corruption in the pending events collection).

8.2.6. INTERNAL MONITOR

- **INTERNAL-MONITOR-INIT**
  This event occurs when a bad initialization of the internal monitor happens. It is a failure case, happening for instance when it is not configured
- **INTERNAL-MONITOR-RUN**
  This event occurs when the internal monitor completes successfully.

8.2.7. KRL

- **KRL-GEN**
  This event occurs on a KRL generation attempt, either requested by application processes or the user.
- **KRL-SYNC**
  This event is triggered when a failure occurs on a KRL Synchronization.

8.2.8. LICENSE

- **LICENSE-EXPIRED**
  This event occurs when the license has expired.
- **LICENSE-INVALID**
  This event occurs when the license contains no entitled modules.
- **LICENSE-MODULE-NOT-ENTITLED**
  This event occurs when the requested module is not entitled on the license.

8.2.9. LIFECYCLE

- **LIFECYCLE-ENROLL**
  This event is triggered when an enrollment request for an end-entity certificate is received. The event specifies all the requested certificate fields, as well as CA, keystore and template information. In case of success, the issued certificate PEM is specified. In case of failure, the reason of the failure is specified (e.g.: "Unauthorized DN element").
- **LIFECYCLE-REVOKE**
  This event occurs when a user tries to revoke a certificate. Note that no event is triggered when a certificate expires.

8.2.10. OCSP

- **OCSP-CSR**
  This event is triggered when issuing a CSR for an OCSP Signer.

8.2.11. SECURITY

- **BOOTSTRAP-ADMINISTRATOR**
  This event is triggered when installing Stream, it corresponds to the creation of the initial administrator account (replaced by BOOTSTRAP-ADMINISTRATOR-PRINCIPAL & BOOTSTRAP-ADMINISTRATOR-ACCOUNT).

WARNING	Deprecated since version 2.0.0

- **SEC-AUTHENTICATION**
  This event is triggered when a user tries to connect. The identifier (local, OpenID, X509 DN, ...) is specified whether it is a failure or a success.

ACCOUNT

- **SEC-ACCOUNT-ADD**
  This event occurs when an account is created (replaced by authorizations & local accounts).

WARNING	Deprecated since version 2.0.0

- **SEC-ACCOUNT-DELETE**
  This event occurs when an account is deleted (replaced by authorizations & local accounts).

WARNING	Deprecated since version 2.0.0

- **SEC-ACCOUNT-UPDATE**
  This event occurs when an account is updated (replaced by authorizations & local accounts).

WARNING	Deprecated since version 2.0.0

AUTHORIZATION

NOTE	These events relate to the Security>Access Management>Authorizations tab under configuration.

- **SEC-AUTHORIZATION-ADD**
  This event is triggered when a user tries to create a an authorization object.
- **SEC-AUTHORIZATION-DELETE**
  This event is triggered when a user tries to delete an authorization object.
- **SEC-AUTHORIZATION-UPDATE**
  This event is triggered when a user tries to modify elements inside an authorization object. The event specifies the modified fields.

CREDENTIALS

NOTE	These events relate to the Security>Credentials tab under configuration.

- **SEC-CREDENTIAL-ADD**
  This event occurs when a user tries creating new credentials.
- **SEC-CREDENTIAL-DELETE**
  This event occurs when a user tries deleting credentials.
- **SEC-CREDENTIAL-UPDATE**
  This event occurs when a user tries updating credentials.

IDENTITY

NOTE	These events relate to the Security>Access Management>Identity tab under configuration.

- **SEC-IDENTITY-PROVIDER-ADD**
  This event occurs when a user tries creating an identity provider profile.
- **SEC-IDENTITY-PROVIDER-DELETE**
  This event occurs when a user tries deleting an identity provider profile.
- **SEC-IDENTITY-PROVIDER-UPDATE**
  This event occurs when a user tries modifying an identity provider profile. The modified fields are specified in the event.

LOCAL IDENTITY

NOTE	These events relate to the Security>Access Management>Local accounts tab under configuration.

- **SEC-LOCAL-IDENTITY-ADD**
  This event is triggered when a user tries creating a local account.
- **SEC-LOCAL-IDENTITY-DELETE**
  This event is triggered when a user tries to delete a local account.
- **SEC-LOCAL-IDENTITY-RESET**
  This event is triggered when executing the reset password workflow.
- **SEC-LOCAL-IDENTITY-UPDATE**
  This event is triggered when a user tries modifying a local account. The modified fields are specified. Updating the password falls in this event.

ROLE

NOTE	These events relate to the Security>Access Management>Roles tab under configuration.

- **SEC-ROLE-ADD**
  This event is triggered when a user tries to create a new role.
- **SEC-ROLE-DELETE**
  This event is triggered when a user tries to delete a role.
- **SEC-ROLE-UPDATE**
  This event is triggered when a user tries to modify a role. The modified fields are specified in the event.

8.2.12. SERVICE

- **SERVICE-START**
  This event is triggered when the Stream service is started.
- **SERVICE-STOP**
  This event is triggered when the Stream service is manually stopped.

8.2.13. TIMESTAMPING

- **TSA-CSR**
  This event is triggered when issuing a CSR for a Timestamping Signer.

8.2.14. TRIGGER

- **CRL-EXTERNAL-STORAGE**
  This event is triggered when a CRL External Storage runs (replaced by TRIGGER-RUN).

WARNING	Deprecated since version 2.0.0

- **TRIGGER-RUN**
  This event occurs when a trigger (External CRL/KRL Storage, Notification) runs.

8.3. Proxies

8.3.1. How to configure an HTTP Proxy

1. Log in to Stream Administration Interface.

2. Access HTTP Proxy from the drawer or card: **System › HTTP Proxies**.

3. Click on  .

4. Fill the fields:

- The **Name*** of the proxy.
- The **Host*** is the Hostname or IP Address of the proxy.
- The **Port*** of the proxy.

5. Click on the create button to save.

You can update or delete the HTTP Proxy.

CAUTION	You won't be able to delete an HTTP Proxy if it is referenced in any other configuration element.

8.4. Queue

8.4.1. Queue Configuration

1. Log in to Stream Administration Interface.

2. Access Queues from the drawer or card: **System › Queues**.

3. Click on  .

4. Fill in the fields:

- The **Name*** of the queue. It must be unique.
- The **Description** to add additional information on this queue.
- The **Throttle Duration** and **Throttle Parallelism**. The maximum number of **parallel** request during the **duration**.
- The **Max Size*** of the queue

CAUTION	If the queue is full every new request will be discarded.

- The **Cluster Wide** parameter defines the queue behavior in multi node setup. If not enabled, then the `throttleParallelism` and `throttleDuration` will be the same for all nodes in the cluster. If enabled, then the `throttleParallelism` and `throttleDuration` is generalized for all clusters.

8.5. Global configuration

These configurations handle various Stream global parameters directly via the Web Interface.

8.5.1. Internal monitoring

This parameter configures the internal monitoring execution interval. Internal monitoring refers to an action that will check the expiration and usage status of credentials and license, and send the configured notifications if needed.

By default, this action will be executed every day at midnight UTC. The notifications will keep being sent each day for as long as an action is needed.

8.5.2. License configuration

The license configuration panel allows to configure Email or REST notifications to be sent on license expiration: using a notification on the `License Expiration` event and the `Delay before notification sending` field in the notification configuration, notifications configured here will be sent by the internal monitoring action.

9. Timestamping

9.1. Timestamping Authorities

To configure a Timestamping Authority, a Timestamping Signer and an NTP Client(s) must already be configured.

1. Log in to the Stream Administration Interface.

2. Go to **Timestamping** › **Authorities** and click on ⊕ at the bottom of the page.

3. Fill the fields:

- The unique **Name*** of the Timestamping Authority
- Choose whether to **Enable** it to sign timestamping requests
- Enter the **Policy OID*** this authority manages
- Add a **Timestamping Signer*** that will sign the requests for this authority
- Select the **Accepted Hash Algorithms*** for signature
- Select the **NTP Client(s)*** that will be the time source for the timestamping
- Choose whether to **Check Revocation** of the signer certificate when processing timestamping requests

4. Click "Save" at the bottom.

If everything was ok, the authority now appears in the list.

9.2. NTP Clients

1. Log in to the Stream Administration Interface.

2. Go to **Timestamping** › **NTP** and click on ⊕ at the bottom of the page.

3. Fill the fields:

- The unique **Name*** of the NTP client
- Enter a **Description** for additional information about this NTP Client
- Enter the **Host*** where to find the NTP server
- Enter the **Port** where to join the NTP server on. Default is the standard 123 port.
- Select a **Timeout*** for NTP requests
- Choose the NTP parameters like the **Max Stratum**, defining the maximum authorized stratum, the **Max Offset**, defining the maximum offset allowed between local system clock and NTP clock and **Max RTT**, the maximum round trip time allowed.

4. Click "Save" at the bottom.

After these steps, the NTP client now appears in the list.

9.3. Timestamping Signers

1. Log in to the Stream Administration Interface.

2. Go to **Timestamping › Signers** and click on ⊕ at the bottom of the page.

3. Fill in the fields to create a Timestamping signer that will sign Timestamping requests:

- The **Name** of the Timestamping signer: a technical name to identify this signer.
- The **Keystore** where to find the key for this signer.
- The **Key** that this signer will sign with.
- The **DN** of this signer, in X500 format with key=value separated by commas.
- The **Notification on signer expiration** that will notify users via Email or REST.

4. You must then generate the CSR ⤵, sign it using your Timestamping CA, and upload the signed certificate back to Stream ⬆

CAUTION	The certificate must be signed with the Key Usage `digitalSignature` (critical) and the Extended Key Usage `timeStamping` (critical)

5. The Timestamping Signer is now uploaded. Additional options are now available:

- The **Response Signing Algorithm**, the hash algorithm that wil be used on responses signed by this signer

6. Click the **Save** button at the bottom of the page.

10. Managing Certificate Lifecycle

10.1. Enroll

NOTE	Stream's RA is not supposed to be a comprehensive registration authority and should only be used when necessary. This simple RA is made for "on the fly" generation only. If you want more advanced RA features to manually enroll certificates, you should consider using Stream's Web RA.

To enroll a certificate via Stream:

1. Log in to the Stream Administration Interface.

2. Go to **Certificates** > **Enroll**. You'll be prompted to fill the following information:

- **CA** *(select)* : The CA that will issue the certificate. The CA must be managed by Stream;
- **Template** *(select)* : The Stream certificate template to use to issue the certificate;
- **CSR type** : Whether the CSR to sign is in a dedicated file (**File** option) or in the clipboard (**Text** option);
- **CSR** field : The CSR to sign (file or PEM-string).

3. Click the **Enroll** button.

Your certificate should now be visible in the Stream search engine.

10.2. Revoke

To revoke a certificate in Stream:

1. Log in to the Stream Administration Interface.

2. Go to **Certificates** > **Search** then find the certificate you want to revoke.

3. Click ⃠ on the certificate you want to revoke. Alternatively, you can click on the certificate's DN then click **Action** > **Revoke**.

Your certificate status should turn red.

10.3. Search

To search for certificates in Stream, log in to the Stream Administration Interface and then go to **Certificates** > **Search**.

Here are all the search criteria you can use:

- **CA**: the issuing certificate authority
- **Status**: the validity status of the certificate (valid, revoked or expired)

- **Template**: the certificate template the certificate has been enrolled on
- **Certificate DNs**: information regarding the certificate's DNs
- **Expiration date**: the date when the certificate will expire
- **Issuer**: information regarding the certificate issuer's DNs
- **Serial**: the certificate's serial number

You can combine any number of them to refine your search.

11. Backup and Restore

This section details how to use the provided EverTrust Tools to back-up and restore Stream if deployed using the RPM package. If you deployed Stream using Docker/Kubernetes, the configuration should be backed-up using the Docker/Kubernetes management platform, and the database should be backed-up using MongoDB tools.

11.1. Backup Procedure

This section details how to back up Stream configuration elements (the /opt/stream/etc folder, that includes the Nginx configuration, and the /etc/default/stream configuration file) and the Stream MongoDB database.

The backup tool allows backing up these elements independently.

```
# /opt/stream/sbin/stream-backup --help
```

Stream Backup tool usage: stream-backup [-cdeho:q]
-c | --conf Backup the Stream configuration files
-d | --db Backup the Stream MongoDB Database
-e | --encrypt Encrypt the backup files with the specified passphrase
-h | --help Display the 'stream-backup' help
-o | --output [path] Specify the Stream backup output folder (default: '/opt/stream/var/backup')
-q | --quiet Quiet mode

To back up the configuration files, run the following command:

```
# /opt/stream/sbin/stream-backup -c
```

The configuration files backup consists of a compressed archive (`.tar.gz`) located under `/opt/stream/var/backup/`.

To back up the MongoDB database, run the following command:

```
# /opt/stream/sbin/stream-backup -d
```

The MongoDB database backup consists of a compressed file (`.gz`) located under `/opt/stream/var/backup/`.

To run a complete backup, execute the following command:

```
# /opt/stream/sbin/stream-backup -c -d
```

NOTE

- The backup output folder can be overridden using the `-o | --output` parameter

- The backup tool can operate in quiet mode (when scheduled in a cron job) using the `-q | --quiet` parameter
- If you want to encrypt your back-up files, use the `-e | --encrypt` parameter. The backup tool will prompt you for a passphrase. The back-up will be encrypted using AES-256.

11.2. Restoration Procedure

This section details how to restore a Stream back-up that was generated using the `stream-backup` tool. The restoration happens using the `stream-restore` tool.

```
# /opt/stream/sbin/stream-restore --help
```

Stream restore tool usage: stream-restore
-a | --archive [filepath] The encrypted backup file to restore
-c | --conf [filepath] The path to the Stream configuration backup file
-d | --db [filepath] The path to the Stream database backup file
-m | --mongo_uri [MongoDB URI] The MongoDB URI to back-up the database into (optional)
-h | --help Displays the 'stream-restore' help
-q | --quiet Quiet mode

Whenever trying to restore a backup, you need to stop the Stream service first:

```
# systemctl stop stream
```

To restore an unencrypted configuration backup, run the following command:

```
# /opt/stream/sbin/stream-restore -c [configuration backup archive path]
```

To restore an unencrypted MongoDB database backup, run the following command:

```
# /opt/stream/sbin/stream-restore -d [MongoDB backup archive path] -m [MongoDB URI]
```

The MongoDB URI is optional: if not provided, the script will try to infer it from the /etc/default/stream file. If it cannot be inferred and none is provided, the restore will fail.

To restore an encrypted backup archive, run the following command:

```
# /opt/stream/sbin/stream-restore -a [encrypted backup archive path] -m [MongoDB URI]
```

The restoration tool will prompt you for the passphrase that was used to encrypt the backup. If the archive contains only a configuration backup, the script will perform the equivalent of the `-c` parameter. If the archive contains only a database backup, the script will perform the equivalent of the `-d` parameter, and you might need to provide the MongoURI through the `-m` parameter. If the archive contains both a database and a configuration backup, both of them will be restored.

When the restoration is complete, you can start Stream again using the following command:

```
# systemctl start stream
```

12. Dictionaries

Here is the list of available dictionary keys to use in computation rules and template strings, depending on the usage.

12.1. Certificate Authority

This dictionary regroups the information of a Certificate Authority.

Key	Description	Type
ca.name	The technical name of the ca	Single value
ca.type	The type of ca (managed or external)	Single value
ca.signer	The values from the signer	Signer dictionary

12.2. OCSP Signer

This dictionary regroups the information of an OCSP signer.

Key	Description	Type
ocsp.name	The technical name of the ocsp signer	Single value
ocsp.signer	The values from the signer	Signer dictionary

12.3. Timestamping Authority

This dictionary regroups the information of a Timestamping authority.

Key	Description	Type
tsa.name	The technical name of the timestamping signer	Single value
tsa.signer	The values from the signer	Signer dictionary

12.4. CRL

This dictionary regroups the information of a CRL.

Key	Description	Type
crl.ca.name	The technical name of the ca that signed the CRL	Single value
crl.ca.type	The type of the ca that signed the CRL	Single value

Key	Description	Type
crl.number	The CRL number	Single value
crl.this_update	The value of this_update	Single value
crl.next_update	The value of next_update	Single value
crl.next_refresh	The value of next_refresh	Single value
crl.size	The number of certificates in the crl	Single value
crl.eidas	"true" if the CRL is eidas compliant, else "false"	Single value
crl.error	The value of the error if the CRL generation failed	Single value

12.5. Credentials

This dictionary regroups the information of a Credential.

Key	Description	Type
credentials.name	The credentials name	Single value
credentials.description	The credentials description	Single value
credentials.expires	The credentials expiration date	Single value
credentials.type	The credentials type	Single value
credentials.target	The credentials target	Single value

In a rest notification, headers can be enriched using the credentials values:

Key	Description	Type
credentials.login	The credentials login value for Password Credentials	Single value
credentials.password	The credentials password value for Password Credentials	Single value
credentials.secret	The credentials secret value for Raw Credentials	Single value

12.6. License

This dictionary regroups the information of the Stream license.

Key	Description	Type
license.expires	The license expiration date	Single value
license.modules	The enabled modules	Single value

12.7. Sub dictionaries

These dictionary cannot be used alone but can be completed with one of the other ones. For example, a valid key is:

```
ocsp.signer.dn.cn.1
```

12.7.1. Signer dictionary

Key	Description	Type
dn	The full dn of the certificate in TYPE=value form	Single valued
dn.<dn field type>	All values of subject field of type `dn field type`	Multi valued
dn.<dn field type>.<index>	Value of subject field of type `dn field type` at index `index`	Single value
sans.<sans field type>	All values of subject field of type `sans field type`	Multi valued
sans.<sans field type>.<index>	Value of subject field of type `sans field type` at index `index`	Single value
issuer	The full dn of the issuer of the certificate in TYPE=value form	Single valued
not_before	Value of the start date of the certificate	Single value
not_after	Value of the expiration date of the certificate	Single value
serial	The certificate serial	Single valued
thumbprint	The certificate thumbprint	Single valued
public_key_thumbprint	The certificate public key thumbprint	Single valued
key_type	The certificate key type	Single valued
signing_algorithm	The certificate signing algorithm	Single valued
pem	The PEM encoded certificate	Single valued

NOTE The valid dn field types are: cn, uid, serialnumber, surname, givenname, unstructuredaddress, unstructuredname, e, ou, organizationidentifier, uniqueidentifier, street, st, l, o, c, description, dc.

NOTE The valid san field types are: rfc822name, dnsname, uri, ipaddress, othername_upn,

	othername_guid.
NOTE	All indexes start at 1

13. Computation rule

Computation Rules are expressions that describe operations to apply to dictionary keys. These keys can come from diverse data sources such as a certification request or a user entry. The available operations and their usage are detailed in this part.

13.1. Example

Let's start by an example:

My CSR contains a DNSNAME subject alternate name with the following value:

```
host.evertrust.fr
```

I want my final certificate to have 2 SANs, this value and its short name: "host".

In order to do that, in **Profile › Certificate Template › Subject Alternate Names** , I add a DNSNAME SAN with the following computation rule:

```
[{{csr.san.dnsname.1}}, Extract({{csr.san.dnsname.1}}, "(.*?)\.", 1)]
```

This will output, in my final certificate, two SANs with values:

```
host.evertrust.fr, host
```

To explain this result, the value "host.evertrust.fr" was retrieved by choosing the first DNSNAME SAN of the CSR: `{{csr.san.dnsname.1}}`. The function `Extract` extracted the first catching group from the regex `(.*?)\.`, resulting in the "host" value.

The computation rule language has a lot more possible operations, allowing complex use cases to become reality.

13.2. Dictionary keys

Dictionary keys are a way to name the information from the available sources. For instance, for a webra enroll, the available sources are the given csr, the webra enroll form data and the principal information if it is authenticated. The full list of available dictionary keys is available on the dictionary page.

13.2.1. Enrollment

A key can reference a single element or a list of elements. It is separated in three main parts: the source of data (csr, webra enroll data form), the section of the data, and an optional number

For example, the following is a valid key with these 3 parts:

`{{csr.subject.cn.1}}`

The csr is the data source, the subject.cn the requested information and the 1 is the index. It allows to retrieve the first, common name from the subject, from the CSR.

Without an index, the key is still valid, but it will output all the corresponding values. For example

`[[csr.subject.ou]]`

This retrieves all the ou from the subject, from the CSR.

WARNING	When a key is expected to output a single value it should be written as a single dictionary key, and one outputing a list of values as a multi dictionary key, otherwise it will be none.

13.3. Basic expressions

Basic string expressions
The following expressions are evaluated as a string or None.

Expression Name	Syntax	Allowed Values	Description	Example
Single dictionary key	{{<key>}}	key: a-zA-A-._	This retrieves a key value from the dictionary, none if it does not exist	{{csr.subject.cn.1}}
Number	<number>	number: -\d+	This will output the given number	-4
Literal	"<literal>"	literal: any string	This will output the given literal	"iAmAString"
Null	NULL	NULL	This will output None	NULL
Now	NOW	NOW	This will output the current instant	NOW

Basic list expressions
The following expressions are evaluated as a list of string or None.

Expression Name	Syntax	Allowed Values	Description	Example
Multi dictionary key	[[<key>]]	key: a-zA-A-._	This retrieves all values that start with key from the dictionary	[[csr.subject.cn]]

Expression Name	Syntax	Allowed Values	Description	Example
Array	[<simpleExpression>, ... <simpleExpression>]	simpleExpression: any expression that will be evaluated to a single element	This will output a multi expression composed of all inserted simple expressions	["iAmAString", {{csr.san.dnsname.1}}]

Quick reference

NOTE Function names are not case sensitive but keys are

Function Name	Syntax
Upper	Upper(`expression`:<expression>)
Lower	Lower(`expression`:<expression>)
Trim	Trim(`expression`: <expression>)
Substr	Substr(`expression`: <expression>, `start`: <number>)
Substr	Substr(`expression`: <expression>, `start`: <number>, `end`: <number>)
Concat	Concat(`expression`: <expression>, ... <expression>)
Extract	Extract(`expression`: <expression>, `regex`: <literal>)
Extract	Extract(`expression`: <expression>, `regex`: <literal>, `group`: <number>)
Replace	Replace(`expression`: <expression>, `regex`: <literal>, `replacement`: <expression>)
OrElse	OrElse(`expression`: <expression>, ... <expression>)
Match	Match(`expression`: <simpleExpression>, `regex`: <literal>)
DateTimeFormat	DateTimeFormat(`expression`: <simpleExpression>, `format`: <literal>)
Get	Get(`expression`: <multiExpression>, `index`: <number>)
First	First(`expression`: <multiExpression>)
Last	Last(`expression`: <multiExpression>)
Filter	Filter(`expression`: <multiExpression>, `regex`: <literal>)

Function Name	Syntax
Slice	Slice(expression: <multiExpression>, start: <number>)
Slice	Slice(expression: <multiExpression>, start: <number>, end: <number>)

13.4. Any expression functions

13.4.1. Upper

```
Upper(expression:<expression>)
```

This outputs the result evaluated from `expression` with only upper case characters and None if no value was evaluated

```
Upper("string") => "STRING"
Upper(["string1", "string2"]) => ["STRING1", "STRING2"]
```

13.4.2. Lower

```
Lower(expression:<expression>)
```

This outputs the result evaluated from `expression` with only lower case characters and None if no value was evaluated

```
Lower("STRING") => "string"
Lower(["STRING1", "STRING2"]) => ["string1", "string2"]
```

13.4.3. Trim

```
Trim(expression:<expression>)
```

This outputs the trimmed result evaluated from `expression` and None if no value was evaluated

```
Trim(" STRING") => "STRING"
Trim(["string1 ", "  string2  "]) => ["string1", "string2"]
```

13.4.4. Substr

```
Substr(expression: <expression>, start: <number>)
```

This outputs the substring from index `start` to the end of the string evaluated from `expression` and None if no value was evaluated or the result of substring is empty. `start` can be negative and it will be computed from end of string.

```
Substr("STRING", 2) => "TRING"
Substr(["string", "longerString", "s"], -2) => ["ng", "ng", "s"]
Substr("tooShort", 15) => None
```

13.4.5. Substr

```
Substr(expression: <expression>, start: <number>, end: <number>)
```

This outputs the substring from index `start` to `end` of the string evaluated from `expression` and None if no value was evaluated or the result of substring is empty. `start` and `end` can be negative and it will be computed from end of string.

```
Substr("STRING", 2, 4)  => "TRI"
Substr(["string", "longerString", "s"], 2, -2) => ["tri", "ongerStri"]
Substr("tooShort", -2, 4) => None
```

13.4.6. Concat

```
Concat(expression: <expression>, ...<expression>)
```

This outputs the concatenation of evaluated expressions: if they are all simple expression, a string concatenation will take place, otherwise an array with all the values will be evaluated. If the final result is empty, None will be returned.

```
Concat("start", " middle ", "end") => "start middle end"
Concat(["string1", "string2", "string3"], "string4") => ["string1", "string2",
"string3", "string4"]
```

13.4.7. Extract

```
Extract(expression: <expression>, regex: <literal>)
```

This extracts from the evaluated `expression` string(s) the part that matches the `regex`

```
Extract("abcd@domain.com", ".*@") => "abcd@"
Extract(["string1", "string2", "string3"], "\d") => ["1", "2", "3"]
```

13.4.8. Extract

```
Extract(expression: <expression>, regex: <literal>, group: <number>)
```

This extracts from the evaluated `expression` string(s) the group at index `group` that matches the `regex`

```
Extract("abcd@domain.com", "(.*)@", 1) => "abcd"
Extract(["string1", "string2", "string3"], "(.*)\d", 1) => ["string", "string",
"string"]
```

13.4.9. Replace

```
Replace(expression: <expression>, regex: <literal>, replacement: <expression>)
```

This replaces parts of the evaluated `expression` string(s) that matches the `regex` with the evaluated `replacement`. If `replacement` is None, values will be replaced by an empty string.

```
Replace("abcdATdomain.com", "AT", "@") => "abcd@domain.com"
Replace(["string1", "string2", "string3"], "\d", CONCAT("This", " was ", " a number"))
=> ["stringThis was a number", "stringThis was a number", "stringThis was a number"]
```

13.4.10. OrElse

```
OrElse(expression: <expression>, ...<expression>)
```

This outputs the first non None result of the given expressions, or None if they are all None

```
OrElse({{not.a.value}}, "abcd@domain.com") => "abcd@domain.com"
OrElse([[no.values]], "value") => ["value"]
OrElse([[no.values]], {{not.a.value}}) => None
```

13.5. String functions

NOTE The following functions output a string or None.

13.5.1. Match

```
Match(expression: <simpleExpression>, regex: <literal>)
```

This outputs the expression if it matches the regex, otherwise None

```
Match("abcd", "[a-z]+") => "abcd"
Match("abcd", "\d+") => None
```

13.5.2. DateTimeFormat

```
DateTimeFormat(expression: <simpleExpression>, format: <literal>)
```

This outputs the expression formatted as format. If expression is not a date, no formatting takes place. Available formats are:

- Custom format in Java DateFormatter syntax
- MILLIS
- BASIC_ISO_DATE
- ISO_LOCAL_DATE
- ISO_OFFSET_DATE
- ISO_DATE
- ISO_LOCAL_TIME
- ISO_OFFSET_TIME
- ISO_TIME
- ISO_LOCAL_DATE_TIME
- ISO_ZONED_DATE_TIME
- ISO_DATE_TIME
- ISO_ORDINAL_DATE
- ISO_WEEK_DATE
- ISO_INSTANT
- RFC_1123_DATE_TIME

```
DateTimeFormat(NOW, "MILLIS") => "1709290260764"
DateTimeFormat(NOW, "hh:mm:ss") => "10:54:57"
```

13.5.3. Get

```
Get(expression: <multiExpression>, index: <number>)
```

This outputs the string at `index` index in the `expression` list, and None if the index does not exist. The index can be negative to get from the end of the list.

```
Get(["string1", "string2", "string3", "string4"], -2) => "string3"
Get(["string1", "string2"], 3) => None
```

13.5.4. First

```
First(expression: <multiExpression>)
```

This outputs the first string of the `expression` list, and None if it does not exist. The index can be negative to get from the end of the list.

```
First(["string1", "string2", "string3", "string4"]) => "string1"
First([[no.values]]) => None
```

13.5.5. Last

```
Last(expression: <multiExpression>)
```

This outputs the last string of the `expression` list, and None if it does not exist. The index can be negative to get from the end of the list.

```
Last(["string1", "string2", "string3", "string4"]) => "string4"
Last([[no.values]]) => None
```

13.6. List of string functions

NOTE	The following functions output a list of string or None.

13.6.1. Filter

```
Filter(expression: <multiExpression>, regex: <literal>)
```

This outputs a list of string from `expression` that matches the `regex`, None if none matches

```
Filter(["string1", "string2", "match"], "[a-z]+") => ["match"]
Filter(["string1", "string2"], "[a-z]+") => None
```

13.6.2. Slice

```
Slice(expression: <multiExpression>, start: <number>)
```

This outputs the slice of the `expression` list between `start` index and its end, or None if the slice is invalid. The index can be negative to get from the end of the list.

```
Slice(["string1", "string2", "string3", "string4"], -2) => ["string3", "string4"]
Slice(["string1", "string2"], 3) => None
```

13.6.3. Slice

```
Slice(expression: <multiExpression>, start: <number>, end: <number>)
```

This outputs the slice of the `expression` list between `start` and `end` index, or None if the slice is invalid. The index can be negative to get from the end of the list.

```
Slice(["string1", "string2", "string3", "string4"], 1, 3) => ["string1", "string2",
"string3"]
Slice(["string1", "string2"], 3) => None
```

14. Template Strings

Template Strings are augmented strings. They can be used as normal text but can also be augmented:

14.1. Using dictionary values

Using the following format, a dictionary key will be interpreted to its value when sending the notification:

```
{{<dictionary key>}}
```

Example:

```
I am enrolling on {{ca.name}}
```

Depending on the notification event, values will be added to context to be interpreted.

NOTE	If the value is not available in the context, the dictionary value will not be replaced

14.2. Using computation rules

Using the following format, a computation rule will be interpreted to its value when sending the notification:

```
{{<computation rule>}}
```

Example:

```
I am enrolling on {{ Lower({{ca.name}}) }}
```

Depending on the notification event, values will be added to context to be interpreted in the computation rule.

NOTE	If the computation rule result is None, an empty string will be displayed. If it is an array, it will be in a comma separated string