Anapaya Knowledge Base

Anapaya Systems

Jul 19, 2024

GETTING STARTED

Welcome to the Anapaya Knowledge Base. Here you can find technical information about the Anapaya EDGE and CORE products. Refer to the index below to jump to the desired topic or use the search function to search the entire knowledge base for specific search terms.

CHAPTER

ONE

OVERVIEW

The Anapaya CORE and EDGE product lines together are designed to enable the deployment and operations of a SCION-based Internet. The CORE product is used by Internet Service Providers (ISPs) to provide the core functionality and the backbone of the SCION Internet, while the EDGE product enables the use of the SCION Internet by transparently encapsulating IP-based traffic into SCION-based traffic (IP-in-SCION Tunneling) and ensuring optimal usage of the SCION Internet by choosing optimal network paths through SCION's path control and multipath capabilities.

On a technical level, the CORE and the EDGE product are implemented as network appliances - both physical and virtual - that can be deployed in existing network infrastructures. One or multiple CORE and/or EDGE appliances together implement a SCION Autonomous System (SCION AS) - the basic unit of the SCION Internet. Each appliance contains a SCION control plane module (control service) and a SCION data plane module (router). The control service takes part in the SCION control plane that creates and disseminates SCION path information, and the router processes and forwards SCION packets according to the path information included in the packet header. Additionally, an EDGE appliance contains an IP-in-SCION tunneling module that encapsulates IP-based traffic into SCION-based traffic.

Given that the Anapaya CORE and EDGE network appliances are similar in their functionality and building blocks, the following sections will commonly refer to them as the **appliance** or **Anapaya appliance**. Where appropriate, the text indicates the differences between the two versions.

1.1 Interacting with an Anapaya Appliance

The Anapaya appliance offers a feature-rich HTTP REST API supporting the following interactions (more will be added in the future):

- Manipulating the configuration of the appliance.
- Inspecting the status of the appliance.
- Provisioning cryptographic material, such as TRCs and certificates.
- Using basic SCION tools such as `scion ping` and `scion showpaths` that help with exploring the network basic troubleshooting.

The OpenAPI 3 specification of the appliance management API is available *here*.

The entire appliance configuration is described in a single JSON file. This facilitates backup and restore of the appliance configuration - an appliance can be restored to a previous state by enacting a backup of the configuration or even freshly provisioned by installing the appliance base image and applying the configuration.

To interact with the management API, there is a range of frontends available:

- curl (or similar) to interact with the appliance via HTTP from the command line.

- A web-UI directly accessible on the appliance.
- Anapaya Organization Manager (OrgMan). OrgMan enables central orchestration of multiple EDGE appliances.

It is also possible to use tools like Ansible to orchestrate a fleet of appliances, however, this is out of scope of this knowledge base.

Section *Appliance Configuration* describes in detail the configuration of an Anapaya appliance.

1.2 Telemetry

Each appliance exports rich telemetry data, logs, and events to a central collector based on Prometheus. Details on the exported metrics and their semantics are described in section *Telemetry*.

1.3 Next Steps

If you are just starting out with the Anapaya appliance, we recommend that you begin with the *Getting Started* section.

If you are already familiar with the Anapaya appliance, we invite you to check out the *Appliance Configuration* section or read one of our User Guides.

CHAPTER

TWO

GETTING STARTED WITH THE ANAPAYA APPLIANCE

This section will guide you through the steps to get the Anapaya Appliance (henceforth only called **appliance**) up and running - from the initial power-on all the way to establishing SCION connectivity.

Note: This guide assumes you are using the Anapaya S-Type physical appliance that already comes with a pre-installed appliance base image. If you need instructions for how to install the appliance base image on your target platform, please refer to *Installing the Appliance Base Image*.

Note: If you require support with provisioning the appliance, do not hesitate to contact us at customer-support@anapaya.net.

2.1 Connecting to the Appliance

Most interactions with the appliance are done via its management API and the web interface. However, on a fresh installation, these components first need to be configured and activated from **within the appliance host**.

To connect to the appliance, you can use SSH. To do that, you first need to know the IP address of the appliance. The appliance attempts to lease an IP via DHCP on each of its interfaces.

```
ssh anapaya@<ip address>
```

Then, you can authenticate with the default password that has been given to you alongside the installation instructions.

If SSH is not an option, you can also use the serial console to connect to the appliance. Please check your platform documentation for instructions on how to connect to the serial console (or GUI). You can use the same default credentials as above to log in.

2.1.1 Configuring the Management Interface

If you connected to the appliance via the serial console or want to configure a different interface for the management connection you can use the standard Linux `ip` tool set. For example, to configure the IP address `192.168.1.1` on interface `eno1`, you can run the following command:

```
sudo ip address add 192.168.1.1/24 dev eno1
sudo ip link set eno1 up
```

2.1.2 Changing User Authentication

If you want to change the default password of the `anapaya` user you can simply run

```
passwd
```

enter the current password and then enter a new password.

2.2 Updating a SCION release

Your appliance comes with a release of the `anapaya-scion` and `anapaya-system` packages installed. However, we recommend upgrading to the latest release containing all the latest features, bug fixes, and improvements. Ask your Anapaya Customer Success Engineer for the most suitable release for your platform.

Note: The `${mgmt_address}` in the `curl` commands below needs to be replaced with the actual management address of the appliance. This can either be an IP address or a DNS name.

Note: The `-k` option on the `curl` commands in the following steps is required to skip verification of the TLS certificate. The appliance management API uses a self-signed certificate. This will be configurable in the future.

Note: The `-u <user>:<password>` option on the `curl` commands in the following steps is required to pass the user and password to the API. The `anapaya` user is a default user that is configured in the base image. The password should have been provided to you. If that is not the case, ask your Anapaya Customer Success Engineer for it.

To update the `anapaya-scion` or `anapaya-system` release, follow the instructions in *Updating an Appliance*.

2.3 Provisioning an Initial Configuration

Now that the initial installation is complete, you can use the appliance management API to provision an initial configuration.

The entire appliance configuration can be stored in a single JSON file - we will refer to this file as `appliance.json` in the following. It contains all the configuration - the networking interfaces on the appliance host, the SCION and IP-in-SCION tunneling configuration, authentication, and other system configuration. In this guide, we assume that you already have an initial appliance configuration, e.g., from your service provider or from the Anapaya Customer Success Engineering team, that contains the main network interface and SCION configurations.

Tip: For more information on the contents of the appliance configuration, please refer to *Appliance Configuration* section and the *Management API specification*.

You can enact the configuration by using the `/config` endpoint of the management API:

```
curl -X PUT -k -u <user>:<password> https://${mgmt_address}/api/v1/config -d @appliance.
↪json
```

Similarly, you can always retrieve the active configuration and store it to a file:

```
curl -k -u <user>:<password> https://${mgmt_address}/api/v1/config > appliance.json
```

2.4 Configuring Authentication of the Management API

Now, that you have provisioned the initial configuration, we will look at how to configure authentication for the management API.

To do that, open the `appliance.json` file in your favorite text editor and add the following configuration (for more information refer to *Appliance Management Configuration*):

```
{
   "management": {
      "api": {
         "listeners": [
            {
               "description": "Management API",
               "address": "192.168.1.1:443"
            }
         ],
         "basic_auth": {
            "enabled": true,
            "users": [
               {
                  "username": "admin",
                  "password_hashed": "$2y$05$9j6jB0DORgxVir0QerURDO669uiuMlbZSfz8F0t/Ty/
↪GsjanBZlfW"
               }
            ]
         }
      }
   }
}
```

This configures the management API to be available on `192.168.1.1:443` and enables HTTP basic auth for user `admin` and password `password`. The `password_hashed` value can be derived using the interactive `appliance-cli` command on an appliance or by using the Apache `htpasswd` command:

```
appliance-cli crypto kdf hash

htpasswd -nB admin
```

Note: Adjust the `address` value to the address of the appliance host in your setup. Here we assume that the appliance host is reachable on `192.168.1.1`.

If you need to add more users, you can simply extend the `users` list in the `basic_auth` section.

Save the changes to `appliance.json` and apply the configuration by using the `/config` endpoint of the management API:

```
curl -X PUT -k -u <user>:<password> https://${mgmt_address}/api/v1/config -d @appliance.
↪json
```

You should now be able to access the management API from outside the appliance and only if you authenticated with the correct username and password. You can verify this by running the following command from outside the appliance host:

```
curl -k https://192.168.1.1/api/v1/config -u admin:<your-password>
```

Note: The initial `anapaya` user is now no longer usable because the configuration only contains the `admin` user.

Tip: The appliance comes with a web-ui that can be used to view and update the appliance configuration, inspect other system information, and use a range of tools to do network troubleshooting. The web-ui can be accessed by navigating your browser to `https://192.168.1.1/`.

2.5 Provisioning the TRC and AS Certificate

As the final step before being able to establish SCION connectivity, you need to provision at least the Trust Root Configuration (TRC) of your SCION ISD as well as a SCION AS certificate issued by a certificate authority (CA) of your SCION ISD.

Tip: This guide only covers the basics of provisioning the TRC and AS certificate. For more in-depth information please refer to *Certificate/TRC Provisioning*.

2.5.1 TRC Provisioning

Assume that you have a TRC called `ISD1-B1-S1.trc` which would be the TRC of SCION ISD 1. To install this TRC, you need to run the following command:

```
curl -k https://192.68.1.1/api/v1/cppki/trcs -X POST --data-binary "@ISD1-B1-S1.trc" -u␣
↪admin:<your-password>
```

The given TRC is first validated before it is added to the trust store. Only valid TRCs are added to the trust store.

To verify that the TRC has been added, you can run the following command:

```
curl -k https://192.68.1.1/api/v1/cppki/trcs -u admin:<your-password>
```

Tip: When your appliance does not have any TRCs installed, you can use the Add TRC bundle endpoint to install the full chain of TRCs at once. Download the TRC bundles from *Isolation domains* overview.

2.5.2 AS Certificate Provisioning

To create an initial AS certificate, you need to instruct the appliance to create a Certificate Signing Request (CSR). This CSR then needs to be signed by a local CA AS. The result is a certificate chain consisting of the issued SCION CPPKI AS certificate and the issuing SCION CPPKI CA certificate. This certificate chain then needs to be installed on the appliance.

To generate a CSR, you need to first create a file `cp-as.json` with the following content:

```
{
   "subject": {
      "isd_as": "1-ff00:0:110",
      "common_name": "Anapaya Switzerland AS",
      "country": "CH",
      "locality": "Zurich",
      "organization": "Anapaya Systems AG",
      "organizational_unit": "Anapaya Systems Engineering Department",
      "postal_code": "8005",
      "province": "Zurich",
      "serial_number": "CHE 123.456.789",
      "street_address": "Hardturmstrasse 253, 8005 Zurich"
   }
}
```

Note: You should adapt the values to your needs.

Note: Generally, only the `isd_as` value is required and everything else is optional. We do recommend that you at least specify a `common_name`. Some CAs might have different requirements on these fields. Please refer to the CA of your ISD for more information.

To generate a fresh CSR and store it in a file `cp-as.csr` you can run

```
curl -k https://192.68.1.1/api/v1/cppki/csrs -X POST --data '@cp-as.json' -u admin:<your-
↪password> > cp-as.csr
```

The resulting CSR is a PEM-encoded file that needs to be signed by a CA. This is an out-of-band mechanism and is defined by the CA that is responsible for issuing your initial AS certificate.

After you have received back the SCION CPPKI AS certificate as part of a full SCION CPPKI certificate chain, you can install it on the appliance:

```
curl -k https://192.68.1.1/api/v1/cppki/certificates -X POST --data-binary "@cp-as.pem" -
↪u admin:<your-password>
```

This first verifies the certificate chain against the active TRC of the local ISD before it is added. Only verifiable certificate chains are added.

To verify that the certificate has been added, you can run:

```
curl -k https://192.68.1.1/api/v1/cppki/certificates -u admin:<your-password>
```

From this point on, the AS certificate gets automatically renewed by the appliance and you do not have to manually provision it anymore.

Note: This only works assuming that the appliance has SCION connectivity to the CA. If you are not able to establish SCION connectivity within the validity period of the provisioned AS certificate, you will have to repeat the process above to request and provision a new AS certificate.

2.6 Verifying SCION Connectivity

You are now at the stage where you can verify that SCION connectivity is established. Showpaths is the primary tool available to do this.

Tip: The `scion` tool is installed on your appliance. Alternatively, you can also access it from the web-ui that is available on your appliance by navigating your browser to `https://192.168.1.1/` or use the management API directly (see here).

Showpaths will show all paths that are available to a given remote AS. Furthermore, it also automatically probes all paths to verify their health. Having showpaths report a set of available paths that are healthy, indicates that the SCION control plane and data plane are working properly.

The basic command to show all the available paths to a remote AS is

```
scion showpaths <remote_isd_as>
```

where `<remote_isd_as>` is the ISD-AS of the remote AS you want to probe. In a first step, it makes sense to probe a direct neighbor, e.g., your upstream AS or a peering AS. You should already know the ISD-AS identifier from the configuration file, where the links to neighboring ASes are configured.

A further sensible choice is to probe all paths to the issuing AS for your AS certificates. You can get its ISD-AS identifier by either inspecting the appliance configuration (`cppki.issuers` section) or the AS certificate chain that you installed on the appliance in a previous step.

Finally, if you are connected to the public SCION Internet, you can use one of Anapaya's SCION ASes `64-2:0:1a` for probing.

```
scion showpaths 64-2:0:1a
```

2.7 Next Steps

You now have a fully configured appliance that is connected to a SCION network. A likely next step is to configure *IP-in-SCION tunneling* to enable legacy IP applications to use the SCION network for communication. You might also want to upgrade the appliance operating system to the latest version. Refer to *Software Updates* for documentation on how to do this.

CHAPTER

THREE

CONFIGURATION

The configuration of an Anapaya appliance is defined by a JSON object. In this section, we describe the contents and their configuration responsibilities at a higher level. For a detailed description of the configuration object, refer to the Appliance Management API specification.

3.1 SCION

The SCION section contains the configuration of the SCION protocol and AS. It consists of a list of SCION AS configurations and the appliance cluster configuration. An Anapaya appliance can have configurations for multiple SCION ASes, however, in most cases there will only be a single SCION AS configured.

Each SCION AS configuration is further divided into multiple sections - *general AS configuration*, *data plane configuration*, the *control plane configuration*, and - for ASes that also act as a certificate authority - the *certificate authority configuration*.

3.1.1 General AS Configuration

Each SCION AS has several general AS configuration options, such as the ISD-AS identifier, the AS forwarding secret key, or a human-readable description of the AS. The following configuration options are available:

`isd_as`
: The ISD-AS identifier is the tuple that uniquely identifies the AS within a SCION isolation domain (ISD). This number is usually assigned by a numbering authority.

`forwarding_key`
: The forwarding secret key is used to authenticate the hop field of the SCION path which corresponds to this AS. It is a local secret that only needs to be known to the AS. The forwarding key MUST be identical across all appliances in an AS. It is specified as a base64-encoded string in the configuration file.

`core`
: The `core` flag must only be set if the AS is a core AS in its ISD. A core AS provides connectivity to other SCION ISDs and operates the main path directories within its ISD. It also regularly initiates path construction beacons that create path segments to other SCION ASes in the same ISD or to other core ASes in different ISDs. For most of the ASes this is not the case and the flag must be set to `false` or not specified (defaults to `false`).

`shard_id`
: The control and the data plane of a SCION AS is split into multiple shards. Each shard is responsible for processing and disseminating path information only for a subset of links. Shards are units that operate and fail independently from other shards. Usually, each appliance operates as a separate shard to increase the resiliency of the network. This field is the ID of the shard to which the control service and the router on this appliance belong.

`scion_mtu`
: The maximum transmission unit in bytes for SCION packets on the internal network of the AS. This represents the protocol data unit (PDU) of the SCION layer on this interface and is usually calculated as the maximum Ethernet payload - IP Header - UDP Header.

`default`
: Default indicates whether the respective SCION AS should be used by default as the source AS by SCION applications, e.g., **scion ping** or **scion showpaths**. The configurations with more than one default ASes will be rejected because there can only be one default AS. If there is only a single AS configured, it will be the default. Therefore, this setting is only necessary if multiple ASes are configured on the appliance.

`details`
: User-defined details about the SCION AS for informational purposes.

General SCION AS configuration

This example shows how to configure two SCION ASes on the appliance. It only includes the general AS configuration.

```
{
  "scion": {
    "ases": [
      {
        "isd_as": "1-ff00:1:1",
        "forwarding_key": "secret1",
        "core": true,
        "shard_id": 1,
        "scion_mtu": 1472,
        "default": true,
        "details": "This is a test AS in ISD 1"
      },
      {
        "isd_as": "2-ff00:1:1",
        "forwarding_key": "secret2",
        "core": false,
        "shard_id": 2,
        "scion_mtu": 1472,
        "default": false,
        "details": "This is a test AS in ISD 2"
      }
    ]
  }
}
```

3.1.2 Data Plane Configuration

Even though SCION is a layer 3 network protocol, SCION packets are encapsulated in UDP/IP packets between individual SCION routers and SCION endhosts (generally referred to as SCION nodes). In other terms, SCION is using a UDP/IP underlay to transport SCION packets between SCION nodes. These UDP/IP packets are only valid between two SCION nodes and change after every SCION hop.

The SCION data plane configuration consists of the `router` and the `neighbors` sections. The `router` section contains the configuration of the *internal* SCION interface `internal_interface`, while the `neighbors` section defines all the SCION links to other SCION ASes, i.e., the *external* SCION interfaces.

Internal SCION Interface

The internal SCION interface is where the appliance receives SCION packets from AS internal hosts for forwarding. It is defined as a UDP/IP network endpoint on which the SCION packets are received from the internal network. The endpoint needs to be specified as a string in the format `<ip>:<port>`. If the port is set to 0 it will be automatically assigned.

Internal SCION interface configuration

```
{
  "scion": {
    "ases": [
      {
        "router": {
          "enabled": true,
          "internal_interface": "192.168.1.1:31000"
        }
      }
    ]
  }
}
```

Neighbors

Each entry in the `neighbors` section contains a remote SCION AS to which one or multiple links are established. It contains the following entries:

`neighbor_isd_as`
: The ISD-AS of the neighbor.

`relationship`
: The relationship with the neighbor. Possible values for the `relationship` are `CORE`, `CHILD`, `PARENT`, and `PEER`. A `CHILD`/`PARENT` relationship indicates that the remote AS is a child/parent, i.e., downstream/upstream of the local AS. A `PEER` relationship indicates a (usually) settlement-free peering relationship with the neighboring AS. A `CORE` relationship is specific to a link between two SCION Core ASes. These are the ASes that are at the core of a SCION ISD. The `relationship` applies to all the links between two ASes, i.e., it is not possible to have a `CORE` and at the same time a `PARENT` link to the same remote AS.

`interfaces`
: The `interfaces` section is used to configure a list of SCION links. The local end of the link is an *external* SCION interface. Each interface definition has the following entries:

  `interface_id`
  : The unique ID of the external SCION interface. This is the identifier that is also contained in a SCION path that uses this interface.

  `address`
  : The local UDP/IP endpoint on which SCION packets from the remote side of the link are received. `address` is specified as a `<ip>:<port>`, where both `<ip>` and `<port>` must be explicitly specified.

  `remote`
  : The configuration for the remote end of the link. It contains the UDP/IP `address` and the `interface_id` of the remote side of the link. These parameters need to be agreed upon by the two ASes.

`administrative_state`
: The administrative state of the interface with one of the following values:

  - `UP` The interface is up and ready to send/receive SCION packets as well as being advertised during beaconing.
  - `DATAPLANE_ONLY` The interface is up and ready to send/receive SCION packets, but is not advertised during beaconing.
  - `ADMIN_DOWN` The interface is down and neither sends/receives SCION packets nor is it advertised during beaconing.

`scion_mtu`
: The maximum transmission unit in bytes for SCION packets on the external interface. This represents the protocol data unit (PDU) of the SCION layer on this interface and is usually calculated as maximum Ethernet payload - IP Header - UDP Header. Note that the SCION MTU can differ between the various links and also from SCION MTU supported in the internal network of the local AS.

`enable_scion_rss`
: Option to activate SCION RSS for this link. If activated, the router utilizes UDP source port entropy on the underlay such that the remote router can leverage RSS for SCION traffic. This must only be set to true if the neighbor supports the SCION RSS feature.

Single PARENT link to upstream AS

This example shows a single PARENT link to an upstream AS using an IPv6 underlay network.

```
{
  "scion": {
    "ases": [
      {
        "neighbors": [
          {
            "neighbor_isd_as": "1-ff00:1:1",
            "relationship": "PARENT",
            "interfaces": [
              {
                "interface_id": 1,
                "address": "[fd02:e8a2:c9e2:03e6::2]:30100",
                "administrative_state": "UP",
                "scion_mtu": 1452,
                "remote": {
                  "address": "[fd02:e8a2:c9e2:03e6::1]:30100",
                  "interface_id": 201
                }
              }
            ]
          }
        ]
      }
    ]
  }
}
```

Two CORE links between two SCION Core ASes

This example shows two CORE links to a neighboring SCION Core AS.

```
{
  "scion": {
    "ases": [
      {
        "neighbors": [
          {
            "neighbor_isd_as": "1-ff00:1:1",
            "relationship": "CORE",
            "interfaces": [
              {
                "interface_id": 1,
                "address": "169.254.1.1:30100",
                "administrative_state": "UP",
                "scion_mtu": 1472,
                "remote": {
                  "address": "169.254.1.2:30101",
                  "interface_id": 3
                }
              },
              {
                "interface_id": 200,
                "address": "169.254.100.1:30100",
                "administrative_state": "ADMIN_DOWN",
                "scion_mtu": 1472,
                "remote": {
                  "address": "169.254.100.2:30101",
                  "interface_id": 1
                }
              }
            ]
          }
        ]
      }
    ]
  }
}
```

Bidirectional Forwarding Detection

Bidirectional Forwarding Detection (BFD) is used to track the healthiness of a SCION link and can be configured for each external interface.

Note: In most cases, the default values work well and the BFD configuration does not need to be explicitly specified.

enabled
: If set to true, then the BFD session is enabled on the SCION interface - if it is set to false, BFD is disabled on that SCION interface. When disabled, the health of the interface is not tracked and it is assumed to be healthy. Note that the remote side of this SCION interface should have the same setting for enabled. If not set this defaults to `true`.

desired_minimum_tx_interval
: The minimum interval between transmission of BFD control packets that the operator desires. This value is advertised to the peer, however, the actual interval used is specified by taking the maximum of desired-minimum-tx-interval and the value of the remote required-minimum-receive interval value. This value is specified as an integer number of microseconds. Defaults to 200000 (200ms).

detection_multiplier
: The number of packets that must be missed to declare this session as down. The detection interval for the BFD session is calculated by multiplying the value of the negotiated transmission interval by this value. Defaults to 3.

required_minimum_receive
: The minimum interval between received BFD control packets that this system should support. This value is advertised to the remote peer to indicate the maximum frequency (i.e., minimum inter-packet interval) between BFD control packets that is acceptable to the local system. Defaults to 200000 (200ms).

Custom BFD configuration

This example shows how to configure BFD for an external interface that runs over an unreliable network. To account for that, the detection multiplier is increased to 10 and timers decreased to 50ms (50000us), i.e., BFD packets are sent more frequently and more missing packets are tolerated before declaring the link as down.

```
{
  "bfd": {
    "enabled": true,
    "desired_minimum_tx_interval": 50000,
    "detection_multiplier": 10,
    "required_minimum_receive": 50000
  }
}
```

3.1.3 Control Plane Configuration

The SCION control plane configuration consists of the `control` and `cppki` sections that configure the network endpoint of the control service and the AS certificate renewal parameters.

control
: Section `control` is the configuration for the SCION control service.

  address
  : This configures where the control service is exposed. Clients connect to this address to request control plane data, e.g., SCION path segments. The address needs to be specified as a string in the format `<ip>:<port>`. If the port is set to 0 it will be automatically assigned.

  enabled
  : If set to true, the control service is enabled. If set to false, the control service is disabled.

cppki
: The `cppki` section contains the configuration of the SCION control plane PKI.

  issuers
  : A list of SCION certificate authorities that should be used to renew the SCION AS certificate of this appliance. `isd_as` is the ISD-AS identifier of the AS that runs the CA. `priority` indicates the priority of the issuing AS. The appliance attempts to get certificates issued from the AS with the highest priority. The value 0 indicates the highest priority, higher numbers are lower priority.

`disable_auto_renewal`
: Whether automatic renewal of AS certificates should be disabled. Usually, this value should not be set. By disabling certificate renewal, the appliance is set into a manual mode where new AS certificates must be provisioned manually and periodically.

Control Plane configuration

This example shows how to configure the control plane including a primary and a secondary certificate issuing CA. The appliance will try to contact 1-ff00:1:100 first, and if that fails, fallback to 1-ff00:1:200.

```
{
  "scion": {
    "ases": [
      {
        "control": {
          "enabled": true,
          "address": "192.168.1.1:40100"
        },
        "cppki": {
          "issuers": [
            {
              "isd_as": "1-ff00:1:100",
              "priority": 0,
            },
            {
              "isd_as": "1-ff00:1:200",
              "priority": 1,
            }
          ]
        }
      }
    ]
  }
}
```

3.1.4 Certificate Authority Configuration

For ASes that act as a CA in one or multiple SCION ISDs, the CA configuration section contains the necessary parameters to interface with SCION CA backend.

Note: The certificate authority (CA) configuration is only needed for SCION ASes that act as a certificate authority in one or multiple SCION ISDs. For all other ASes this section must not be configured.

`service_type`
: The `service_type` indicates what type of CA service is used. Currently, the Anapaya appliance supports three types of CA services:

  - `ANAPAYA_VAULT` is the SCION CA backend provided by Anapaya. It is based on Hashicorp Vault with a suitable frontend. If this service type is used, the `anapaya_vault` section must be configured.
  - `EXTERNAL` is a generic CA backend not implemented by Anapaya. Note that the external CA backend must implement the SCION CA API to be compatible with the Anapaya appliance. If this service type is used,

the `external` section must be configured.

- `IN_PROCESS` is a CA service that is implemented in the Anapaya appliance. It should only be used for testing purposes and is not suitable for productive use. If this service type is used, no other section needs to be configured.

Anapaya Vault CA

The `anapaya_vault` section contains the configuration for the Anapaya Vault based CA service. It must be provided if the `service_type` is set to `ANAPAYA_VAULT`.

`addresses`
: This field contains a list of addresses where Anapaya Vault backend can be reached. Addresses are of the form `https://<host>:<port>`, where `<host>` can either be a hostname or an IP address. The list of addresses must not be empty.

`role_id` and `secret_id`
: These are the credentials used to authenticate with the Anapaya Vault backend. Refer to the *Anapaya Vault setup guide* for how these credentials are created.

Anapaya Vault CA backend configuration

```
{
  "ca_service": {
    "service_type": "ANAPAYA_VAULT",
    "anapaya_vault": {
      "addresses": [
        "https://vault.example.com:8200"
      ],
      "role_id": "role-id",
      "secret_id": "secret-id"
    }
  }
}
```

External CA

The `external` section contains the configuration for a generic external CA backend that implements the SCION CA API.

`address`
: The address of the external CA backend. It is of the form `https://<host>:<port>`, where `<host>` can either be a hostname or an IP address.

`client_id` and `shared_secret`
: These are the credentials used to authenticate with the external CA backend. They must be agreed with the external CA backend operator. Starting with release v0.31, the shared secret can be configured directly. Prior releases require that the shared secret is PEM encoded with the type `SYMMETRIC KEY`.

External CA backend configuration

```
{
  "ca_service": {
    "service_type": "EXTERNAL",
    "external": {
      "address": "https://ca.example.com:5000",
      "client_id": "client-id",
      "shared_secret": "-----BEGIN SYMMETRIC KEY-----\
↪nMIIB6jCCAZCgAwIBAgIUO1CvSfi8Kueey1+f\n-----END SYMMETRIC KEY-----"
    }
  }
}
```

3.2 Cluster

A SCION AS consists of one or more appliances that form a cluster to synchronize state. The appliances are the shards of the AS cluster and are referred to with a shard identifier. Each shard is an independent unit of the SCION AS and can function and fail independently of the other shards.

Note that this is an Anapaya-specific extension of how a SCION AS is internally implemented while fully adhering to the specification of the SCION protocol.

Note: An appliance can be configured to take part in several ASes for instance to connect to multiple isolation domains. Hence, an appliance can be a member of several AS clusters. In each of the clusters it is identified by a shard id that is unique within the cluster.

The shard ids of the appliance are configured in the `scion.ases` section and referenced in the peer's cluster section.

Appliances synchronize two kinds of state:

- **SCION control plane data**
  Consists of path segments and beacons that are used to construct the ASes path database. The synchronization of control data happens only within an AS and is performed by the SCION control service. The control service synchronizes its path segments and beacons with the control services of all peers whose addresses it learns from the topology information. This happens regardless of whether the topology information is statically configured or synchronized.
- **Topology information**
  Consists of addresses of SCION control services, IP-in-SCION tunneling endpoints and information about SCION links to neighbors. The synchronization of topology information is configured in the `cluster` section. Topology information can be automatically synchronized or configured statically. In both cases, each cluster member is configured with a list of all of its peers.

3.2.1 SCION Control Plane Data Synchronization

SCION control plane data is synchronized by the control services. The local control service addresses are configured in the `scion.ases` section. Refer to the documentation of the *SCION section* for a detailed description of this section. The addresses of remote control services are configured or synchronized through topology synchronization and no additional configuration is required.

3.2.2 Topology Synchronization

The topology synchronization is configured in the `cluster` section which has the following fields:

`synchronization`
: Configures the synchronization endpoint of the appliance where topology information is served. If all peers in the cluster are configured statically this section does not need to be configured.

  `address`
  : The `<host>:<port>` TCP address where topology information is served.

  `node_synchronization_interval`
  : Configures the interval between two consecutive topology synchronization attempts to the cluster peers.

`peers`
: For every peer in the cluster, this section either statically configures the topology information or configures the endpoint for automatic synchronization. In case there is only a single appliance in the cluster, the list of peers is empty.

`features`
: Configures features that are announced to the peers.

  `scion_rss`
  : Configures whether the appliance announces support for SCION RSS to its peers. If the local host supports the SCION RSS feature, per default it will announce this to the peers.

The appliance uses the peer's `scion` and `synchronization` fields to decide whether to initiate topology synchronization with them. The `scion` and `scion_tunneling` sections of the peer are mutually exclusive with its `synchronization` section. If the synchronization section is configured then the topology is dynamically synchronized from the peer. If either the `scion` section or the `scion_tunneling` section is configured then no topology synchronization is initiated with this peer.

Note: Even if topology synchronization is disabled the control service will still dynamically synchronize its SCION control plane data (beacons and path segments) with all control services that are configured as peers in the same AS.

Note: The decision to configure topology information statically or to enable synchronization should be the same for all peers.

Static Configuration

Note: Statically configuring the topology information is recommended for edge deployments where the configuration overhead is small and the topology does not change frequently.

For each statically configured peer the following fields need to be configured:

`cluster.peers.scion.ases`
: Configures the peer's ASes. Note, that this object should match the `scion.ases` section in the peer's appliance configuration. Refer to the *SCION section* for an explanation of the analogous fields in the `scion.ases` section. For each AS, configure the following fields:

  - `isd_as`
  - `control.address`
  - `shard_id`
  - `neighbors`
    - `neighbor_isd_as`
    - `relationship`
    - `interfaces`
      - `interface_id`
      - `next_hop`
      - `scion_mtu`

`cluster.peers.scion_tunneling`
: Configures how to reach the peer's IP-in-SCION tunneling endpoint. This section should match the `scion_tunneling` section in the peer's appliance configuration. Refer to the documentation of the documentation on the *IP-in-SCION tunneling section* for a description of the corresponding fields in the SCION tunneling section. In the section configure the following fields:

  - `ip`
  - `data_port`
  - `control_port`
  - `probe_port`
  - `allowed_interfaces`

`cluster.peers.features`
: Configures the features of the peer. The following fields can be configured:

  - `scion_rss`
    Option to statically enable or disable SCION RSS for traffic forwarded to the peer. This must only be set to true if the peer supports the SCION RSS feature.

Static peer configuration

This example shows how to statically configure a peer's topology information:

```
{
    "cluster": {
        "peers": [
            {
                "name": "peer 1",
                "scion": {
                    "ases": [
                        {
                            "control": {
                                "address": "10.2.0.2:40001"
                            },
                            "isd_as": "1-ff00:1:2",
                            "neighbors": [
                                {
                                    "interfaces": [
                                        {
                                            "interface_id": 2,
                                            "next_hop": "10.2.0.2:31000",
                                        }
                                    ],
                                    "neighbor_isd_as": "1-ff00:1:1"
                                }
                            ],
                            "shard_id": 2
                        }
                    ]
                },
                "scion_tunneling": {
                    "endpoint": {
                        "allowed_interfaces": [
                            {
                                "isd_as": "1-ff00:1:2"
                            }
                        ],
                        "control_port": 40201,
                        "data_port": 40200,
                        "ip": "10.2.0.2",
                        "probe_port": 40202,
                    }
                },
                "features": {
                    "scion_rss": true
                }
            }
        ]
    }
}
```

Automatic Synchronization

Note: Automatic topology synchronization is recommended for appliances in core ASes. This way the configuration does not need to be updated when the topology changes for instance if a new SCION link is added.

To configure a peer for topology synchronization only the endpoint needs to be configured through the `synchronization.address`. This field needs to match the `cluster.synchronization.address` configured in the peer's appliance configuration.

Automatic topology synchronization

This example shows the configuration for automatic topology synchronization with two peers.

```
{
    "cluster": {
        "peers": [
            {
                "name": "peer 1",
                "synchronization": {
                    "address": "10.1.0.3:42003"
                }
            },
            {
                "name": "peer 2",
                "synchronization": {
                    "address": "10.1.0.2:42003"
                }
            }
        ],
        "synchronization": {
            "address": "10.1.0.1:42003",
            "node_synchronization_interval": "10m"
        }
    }
}
```

3.3 Firewall

3.3.1 Overview

The firewall section contains the configuration for the appliance firewall to filter traffic. The appliance firewall makes use of the Linux kernel packet classification framework nftables, thus it is highly recommended that you familiarize yourself with nftables by reading the following short overview: nftables in 10 minutes.

The following documentation assumes that you are familiar with the basic nftables concepts of tables, chains, and rules.

Note: The appliance firewall only affects traffic to and from the appliance host itself. Traffic that is forwarded by the appliance as part of the SCION dataplane and IP-in-SCION tunneling cannot be filtered by the appliance firewall.

3.3.2 Configuration

There are two main methodologies for access control using a firewall. Blacklisting allows all traffic and drops only specific packets, whitelisting on the other hand drops all the traffic by default and only accepts what you specify. The appliance's firewall configuration contains tables of chains, which in turn contain the actual rules for enforcing the desired filtering. Four different firewall modes are available to the user to configure the desired behavior:

- `AUTO` mode is the default mode and generates set of rules based on the rest of the appliance configuration. `AUTO` mode mostly locks down the appliance to only allow traffic required for the appliance to function properly. If you do not have any special requirements, this is the recommended mode.
- `PREPEND` mode uses the rules generated by the `AUTO` mode and allows prepending additional rules to the generated ones. This mode is useful if you want to add custom rules to the appliance firewall without having to reconfigure the whole firewall.
- `CUSTOM` mode does not generate any rules and allows the user to configure the appliance firewall from scratch. This mode is useful if you want to have full control over the appliance firewall.
- `UNMANAGED` mode disables the appliance firewall and does not interfere with existing firewall configurations. This mode should only be used when transitioning from iptables to the appliance-managed firewall.

The generated rules for the `AUTO` and `PREPEND` modes are a set of firewall rules based on the other sections of the appliance configuration. Check the *auto rule generation* section for more information. The following describes the available configuration options independent of the selected mode, followed by the configuration options for each mode.

`mode`
: The mode declares how the appliance firewall is operating. Possible values are `AUTO`, `PREPEND`, `CUSTOM`, and `UNMANAGED`.

Tip: The `AUTO` mode is a good way to secure your appliance while `CUSTOM` mode offers more flexibility. We recommend that you use `AUTO` or `PREPEND` unless you have very specific requirements.

Tables

This section describes the table configuration.

`tables`
: The tables section is used to configure a list of tables using the nftables table syntax. A table contains a list of chains, but has otherwise no effect on the packet filtering.

  `name`
  : The name of the table. This can be any string and has no effect on the processing of packets.

  `family`
  : The family type of the table. The packets seen by the table depends on the family type. Available family types are `INET` for IPv4 and IPv6 packets, `IP` for IPv4 packets, and `IP6` for IPv6 packets.

  `chains`
  : Chains are a list of chains using the nftables chains syntax. Each chain contains rules for the desired packet filtering behavior. A chain can be either a base chain (registered into the Netfilter hooks) or a regular chain (reachable via `jump` statements).

  `counters`
  : A list of named counters which counts the number of packets and the total bytes. Counters need to be attached to rules to count the number of packets and total bytes that were matched by a given rule.

Chains

This section describes the chain configuration. The required fields depend on the type of chain to configure. Base chains attached to the netfilter hooks require the fields `chaintype`, `hook`, `policy`, and `priority` (default 0). Regular chains can only be reached using a jump statement and therefore only include the `name` and the `rules`.

In `PREPEND` mode, only a single base chain per hook can be defined. Furthermore, the base chains for the `INPUT` and `FORWARD` hooks - `default_input` and `default_forward` - are already defined by the appliance and cannot be redefined. Only rules can be prepended to these chains.

`name`
: The name of the chain. When using `PREPEND` mode the names for the base chains of the `INPUT` and `FORWARD` hooks must be `default_input` respectively `default_forward`. Other base chain names can be arbitrary. In `CUSTOM` mode, all names can be arbitrary. For regular chains, any name can be used regardless of the mode.

`chaintype`
: Specify the chain type. Possible values are `FILTER`, `ROUTE` and `NAT`. Only used for base chains.

`hook`
: Specify the netfilter hook for the base chain. Possible values are `INPUT`, `FORWARD`, `OUTPUT`, `PREROUTING` and `POSTROUTING`. Only used for base chains.

`policy`
: Specify the default verdict if the packet reaches the end of the chain. Possible values are `ACCEPT` or `DROP`. Only used for base chains.

`priority`
: Specify the priority of the chain (default 0). Lower numbers have higher priority, e.g., a priority of -100 has higher priority than 0, which in turn has higher priority than 10. Only used for base chains.

`rules`
: The rules define actions on the packet if they match the specified criteria.

Rules

This section describes the rule configuration.

`rule`
: The rule string can contain zero or more expressions followed by one or more statements as documented by nftables. The expressions are evaluated from left to right and if a packet matches all expressions, the statements are executed. Please refer to the nftable documentation for further information (examples for expressions and statements).

`comment`
: Informational comment for the rule.

`sequence_id`
: The sequence ID determines the order of the rules within the chain.

Firewall Modes

AUTO

This mode does not allow any customization. The appliance uses the default table called `appliance` of type `INET` with the *generated rules*. This mode is the default mode and does not need to be specified explicitly. If you want to specify the mode explicitly the tables section must be empty.

AUTO appliance firewall configuration

Simple configuration to use the firewall `AUTO` mode. You can also omit the firewall section.

```
{
  "firewall": {
    "mode": "AUTO",
  }
}
```

PREPEND

The appliance still uses the default table but allows prepending additional rules. The configuration only allows a single table named `appliance` of type `INET`.

In `PREPEND` mode, only a single base chain per hook can be defined. Furthermore, the base chains for the `INPUT` and `FORWARD` hooks - `default_input` and `default_forward` - are already defined by the appliance and cannot be redefined. Only rules can be prepended to these chains.

The rules you define in these chains are then prepended to the generated ones of the same hook by the appliance. For the `OUTPUT`, `POSTROUTING`, and `PREROUTING` hooks you can define at most one base chain per hook. You can still define arbitrary regular chains which can be reached via `jump` statements.

PREPEND appliance firewall configuration

This example shows how to configure the appliance firewall in `PREPEND` mode by defining a custom rule that gets executed before the generated rules for the `default_input` and `default_forward` chains. Furthermore, it defines a base chain for the `POSTROUTING` hook and a regular chain that can be reached via a `jump` statement.

```
{
  "firewall": {
    "mode": "PREPEND",
    "tables": [
      {
        "name": "appliance",
        "family": "INET",
        "chains": [
          {
            "name": "default_input",
            "rules": [
              {
                "comment": "custom input rule",
                "rule": "ip saddr 30.0.0.0/24 accept",
                "sequence_id": 0
```

(continues on next page)

(continued from previous page)

```
            }
          ]
        },
        {
          "name": "default_forward",
          "rules": [
            {
              "comment": "custom forward rule",
              "rule": "ip saddr 10.0.0.0/24 jump my_regular_chain",
              "sequence_id": 0
            }
          ]
        },
        {
          "name": "my_postrouting_chain",
          "hook": "POSTROUTING",
          "chaintype": "NAT",
          "policy": "ACCEPT",
          "priority": 0,
          "rules": [
            {
              "comment": "source nat",
              "rule": "ip saddr 192.168.2.0/24 snat to 1.2.3.4",
              "sequence_id": 0
            }
          ]
        },
        {
          "name": "my_regular_chain",
          "rules": [
            {
              "comment": "accept if you reach this chain",
              "rule": "counter name \"accept_custom\" accept",
              "sequence_id": 0
            }
          ]
        }
      ],
      "counters": [
        {
          "name": "accept_custom"
        }
      ]
    }
  ]
}
```

CUSTOM

This mode provides full control. The tables specified in the configuration are applied by the appliance without any modifications.

CUSTOM appliance firewall configuration

This example shows how to configure the appliance firewall in `CUSTOM` mode.

```
{
  "firewall": {
    "mode": "CUSTOM",
    "tables": [
      {
        "name": "my_custom_table",
        "family": "INET",
        "chains": [
          {
            "chaintype": "FILTER",
            "hook": "INPUT",
            "policy": "DROP",
            "priority": 0,
            "name": "my_custom_chain",
            "rules": [
              {
                "comment": "Connection initiated by the appliance",
                "rule": "ct state established,related counter name \"accept_conntrack\"␣
↪accept",
                "sequence_id": 0
              },
              {
                "comment": "Appliance management API",
                "rule": "tcp dport 443 counter name \"accept_API\" accept",
                "sequence_id": 1
              },
              {
                "comment": "SSH connection",
                "rule": "tcp dport 22 counter name \"accept_ssh\" accept",
                "sequence_id": 2
              }
            ]
          }
        ],
        "counters": [
          {
            "name": "accept_conntrack"
          },
          {
            "name": "accept_API"
          },
          {
            "name": "accept_ssh"
          }
```

(continues on next page)

(continued from previous page)

```
        ]
      }
    ]
  }
}
```

Named Counters

Each rule can include a non-verdict counter statement. Nftables distinguishes between anonymous and named counters. Named counters are defined on the table level and can be referenced in rules by their name (e.g. *counter name "my_counter"*). Anonymous counters are defined within the rule and can only be used in this rule.

3.3.3 Automatic Firewall Rule Generation

In `AUTO` and `PREPEND` mode, the appliance generates a set of firewall rules based on the other sections of the appliance configuration. The following traffic is allowed by default:

- All established and related connections, i.e., connections that are initiated from the appliance itself.
- Traffic on the default loopback interface `lo`.
- Traffic on wireguard interfaces.
- Traffic for port 22 (SSH) and the port of the management API (443 by default) on the management addresses.
- Traffic for the configured telemetry endpoints.
- SCION control plane and appliance cluster synchronization traffic.
- BGP traffic for the configured BGP peers.
- ICMP echo and neighbor discovery traffic.

Any other traffic that is not explicitly allowed by the rules above is dropped.

3.3.4 Limitations

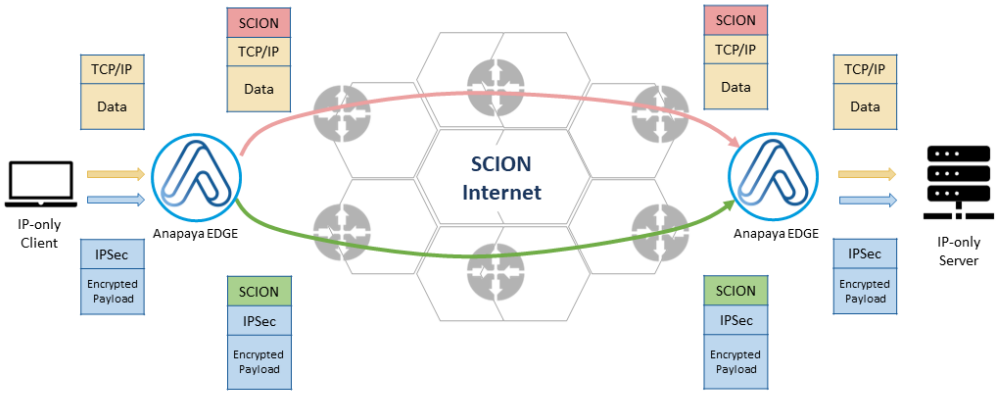
As the appliance firewall is tightly coupled to nftables, the following limitations apply:

- Similar to the counters, nftables supports sets. However, the appliance firewall does not support named sets but only anonymous sets. So if you want to use sets (e.g. *tcp dport { 22, 23 }*), you have to define them within the rule.

3.4 IP-in-SCION Tunneling

3.4.1 Overview

IP-in-SCION tunneling is a mechanism that allows to tunnel IP packets between two IP-in-SCION tunneling endpoints, i.e., Anapaya EDGE appliances. This mechanism enables any type of application and communication to take advantage of SCION-based networking without the need to update any application, client, or server.



On a high-level, IP-in-SCION tunneling involves the following steps:

1. A sender sends an IP packet towards an IP destination.
2. The IP packet reaches an IP-in-SCION tunneling endpoint, such as an Anapaya EDGE appliance, in the sender's network via standard local IP routing.
3. Based on the destination IP address, the Anapaya EDGE appliance determines the destination SCION AS and possible IP-in-SCION tunneling endpoints within the destination SCION AS. To achieve this, the Anapaya EDGE appliance uses the *SCION Gateway Routing Protocol (SGRP)*.
4. Next, the Anapaya EDGE appliance determines the optimal SCION path to reach the destination SCION AS. The choice of the path depends on the *Routing Domain Configuration* and the real-time network performance characteristics, such as latency, jitter, drop rate, etc.
5. Then, the original IP packet is encapsulated within a SCION packet by the Anapaya EDGE appliance and sent to the remote IP-in-SCION tunneling endpoint in the destination SCION AS. The SCION path used is the one chosen in the previous step.
6. The remote IP-in-SCION tunneling endpoint receives the SCION packet and decapsulates the original IP packet. It then forwards the packet to the final IP destination using standard local IP routing.

Routing Domains

Routing Domains are the primary mechanism to configure the *traffic policies* used when communicating with remote IP destinations. Conceptually, a routing domain is defined by a set of IP prefixes to which a common set of traffic policies should be applied as shown in the following figure.

These IP prefixes could be announced by one or multiple SCION ASes. The Anapaya EDGE appliance can optimize path selection across all possible destination SCION ASes (while respecting the configured traffic policy). This flexibility enables operators to capture a wide range of routing use-cases - from simple site-to-site tunnels, to complex setups involving remotes that are part of multiple SCION isolation domains, or destinations that are reachable via many SCION ASes.

A routing domain can be further specified by a set of remote SCION ASes. This is useful for example when a domain should only include remote ASes of a specific ISD or a specific set of remote ASes, e.g., the sites of the company's wide area network. Furthermore, it is also possible to limit the routing domain to a set of *source* ASes. This is used by Anapaya EDGE appliances that are part of multiple ISDs and need to separate the traffic and use different policies per ISD.

Finally, a domain also includes the ability to limit the IP prefixes that are announced by a local Anapaya EDGE appliance to remote IP-in-SCION tunneling endpoints.

Traffic Policies

Traffic policies allow users to influence and control the path selection process of the Anapaya EDGE appliance. For example, an operator can define policies to keep traffic within Swiss jurisdiction or avoid certain network providers. A traffic policy uses *path filters* to specify the set of SCION paths that should be considered for the path selection. Each path filter defines a subset of all available SCION paths.

Path filters are combined into a *failover sequence* to define the order in which the path filters are considered. The path filters in a failover sequence are applied separately to the set of possible paths until one of them returns a non-empty path set.

Finally, the traffic policy includes a *traffic matcher* that defines the set of IP packets for which the failover sequence should be applied.

SCION Gateway Routing Protocol (SGRP)

The SCION Gateway Routing Protocol (SGRP) is a routing protocol that enables IP-in-SCION tunneling endpoints, i.e., Anapaya EDGE appliances, to map IP prefixes to SCION ASes.

At a high level, a tunneling endpoint participating in SGRP between two SCION ASes does the following steps:

1. It discovers the tunneling endpoints in the remote SCION AS. To that end, it periodically sends a discovery message to the control plane of the remote AS which replies with a list of local IP-in-SCION tunneling endpoints.
2. It periodically queries each discovered endpoint in the remote AS to learn the IP prefixes it announces. From that, the local endpoint builds a mapping from IP prefix to remote tunneling endpoints.
3. When queried by a remote tunneling endpoint about its prefixes, the local endpoint replies with the set of IP prefixes it wants to announce to the remote.

For the Anapaya EDGE appliance, the set of announced IP prefixes can either be statically configured or is dynamically learned via BGP. Please refer to the *BGP section* for information on how to configure a BGP session to a peering BGP router.

3.4.2 Domain Configuration

As explained in *Routing Domains*, domains define a common set of traffic policies towards a distinct range of IP destinations. Thus, there are two main aspects to configuring a domain:

1. *What* entities are part of the domain?
2. *How* is traffic routed within the domain?

The *what* includes the IP prefixes that are accepted and announced by the local endpoint, the remote SCION ASes, and possibly the local SCION AS identities (in case the Anapaya EDGE appliance is part of multiple SCION ISDs).

The *how* includes the traffic policies that are applied to outgoing traffic towards any remote destination that is part of the domain.

Domain Entities

This sections explains how to define *what* entities are part of a domain.

IP Prefixes

The `prefixes` section of the domain configuration defines the set of IP prefixes that are *accepted* and *announced* by the local endpoint. To define these sets the `accept_filter` and `announce_filter` subsections are used. Each filter is a list of IP prefix matchers combined with an `action`. The `action` is either `ACCEPT` or `REJECT` to implement allow- and block lists.

Note: The accept and announce filters are what the name implies - filters. This means that not everything they contain is actually accepted or announced. The set of announced IP prefixes are all the IP prefixes configured in *Static Announcements* or received from BGP peers configured in *Border Gateway Protocol (BGP)* and filtered by the announce filter. Similarly, the set of accepted IP prefixes is the set of all IP prefixes received from remote IP-in-SCION tunneling endpoints filtered by the accept filter.

Remote ASes

The `remote_isd_ases` section contains a list of SCION AS matchers that - together with an `action` (`ACCEPT` or `REJECT`) - define the set of SCION ASes that are part of the domain. This has the effect that IP prefix announcements will be accepted from these remote ASes and all IP traffic will be tunneled over paths that end in one of these remote ASes. A SCION AS matcher has the form `<isd>-<as>` where `<isd>` is the ISD ID and `<as>` is the AS ID. Both `<isd>` and `<as>` can be `0` to match all ISDs and all ASes respectively. E.g., the matcher `1-ff00:0:1` matches exactly this single AS, `1-0` matches all ASes that are part of ISD 1, and `0-ff00:0:1` matches the specific AS in all ISDs. A single `0` (or `0-0`) matches all ASes in all ISDs.

Local AS Identities

The `local_isd_ases` section defines a list of local SCION AS identities that are part of the domain. Setting this field is only necessary if the Anapaya EDGE appliance is part of multiple SCION ISDs and an operator wants to restrict the domain to a subset of the local SCION ISDs. This has the effect that traffic towards remote tunneling endpoints in this domain only uses paths that start in one of the listed local ASes. Note that if the Anapaya EDGE appliance is only part of a single ISD or if all available local AS identities are part of the domain, then `local_isd_ases` does not need to be set.

Default Domain

There can be a single `default` domain. The default domain is used when no other domain matches a packet's destination, i.e., the default domain implicitly accepts the entire IP space that is not covered by other domains. For that reason, the default domain can not include any `accept_filter`. Specifying a default domain is optional.

Examples

Allow-all domain

This example shows a domain that accepts any IP prefix from any remote AS and announces any learned or configured IP prefixes to any remote AS.

```
{
   "remote_isd_ases": [
      {
         "isd_as": "0-0",
         "action": "ACCEPT",
         "description": "Include all remote ASes",
         "sequence_id": 0
      },
   ],
   "prefixes": {
      "accept_filter": [
         {
            "prefixes": [ "0.0.0.0/0, ::/0" ],
            "action": "ACCEPT",
            "sequence_id": 0
         },
      ],
      "announce_filter": [
         {
            "prefixes": [ "0.0.0.0/0, ::/0" ],
            "action": "ACCEPT",
            "sequence_id": 0
         },
      ],
   },
}
```

ISD-specific domain

This example shows a domain that only accepts `1.0.0.0/8` and `2.0.0.0/8` from ASes in ISD `1` and announces only prefixes in `1.100.0.0/16`. Furthermore, the domain only includes the local AS identity that is part of ISD `1`.

```
{
   "remote_isd_ases": [
      {
         "isd_as": "1-0"
         "action": "ACCEPT"
         "description": "Only ASes in ISD 1 are part of the domain"
         "sequence_id": 0
      },
   ],
   "local_isd_ases": [ "1-ff00:0:1" ],
   "prefixes": {
      "accept_filter": [
```

(continues on next page)

(continued from previous page)

```
            {
               "prefixes": [ "1.0.0.0/8", "2.0.0.0/8" ],
               "action": "ACCEPT"
               "sequence_id": 0
            }
         ],
         "announce_filter": [
            {
               "prefixes": [ "1.100.0.0/16" ],
               "action": "ACCEPT",
               "sequence_id": 0
            }
         ]
      }
}
```

ISD-specific default Domain

This example shows the definition of a default domain. The default domain can be thought of as the default route in classical IP routing and is entirely optional. While the default domain cannot limit the set of accepted IP prefixes, it is perfectly possible to limit the set of announced IP prefixes, as well as the set of remote ASes.

```
{
   "default": true,
   "remote_isd_ases": [
      {
         "isd_as": "1-0"
         "action": "ACCEPT"
         "description": "Only ASes in ISD 1 are part of the domain"
         "sequence_id": 0
      },
   ],
   "prefixes": {
      "announce_filter": [
         {
            "prefixes": [ "1.100.0.0/16" ],
            "action": "ACCEPT",
            "sequence_id": 0
         }
      ]
   }
}
```

Traffic Policies

Traffic policies let an operator influence how traffic is routed within a domain. Each domain can have one or multiple traffic policies. A traffic policy contains a traffic matcher that defines the set of packets to which the policy applies and a set of path filters (grouped into a failover sequence) that defines what paths are allowed to be used by the matched packets.

A path filter defines a set of rules that a path must satisfy to be accepted by the filter. To select a path for the tunnel, the following steps are taken:

1. The path filters are traversed in order of their sequence ID within the failover sequence.
2. Each path filter is applied to the candidate set of paths.
3. The best path out of the acceptable set is chosen according to path health and performance metrics. If no such path exists, i.e., no candidate path is allowed by the filter or no path is healthy, selection continues to the next path filter in the failover sequence.
4. If all filters have been considered and no healthy path was selected, traffic will be dropped.

Failover Sequence

To illustrate the failover sequence consider the example below. First, the path filter `provider_1_outgoing` is applied to the set of all available paths leading to an acceptable set of paths. If one or more paths of the acceptable set are healthy, the best path is chosen according to its performance metrics. If no path of the acceptable set is healthy, the path filter `allow_all` is applied to the set of all available paths. This leads to a (potentially) different acceptable set. Again, the best healthy path is chosen and if no such path exists, the traffic is dropped.

```
{
   "failover_sequence": [
      {
         "sequence_id": 0,
         "path_filter": "provider_1_outgoing"
      },
      {
         "sequence_id": 1,
         "path_filter": "allow_all"
      },
   ]
}
```

Traffic policies are applied in the order of their sequence ID (ascending). For each IP packet, only a single traffic policy will be applied even if it matches multiple policies. A traffic policy matches if the traffic matcher defined in the policy matches the incoming IP packet.

While only a single traffic policy applies to an incoming IP packet, within the traffic policy the failover sequence can be used to configure "failover" behavior. If a path filter in the failover sequence has no healthy path to the remote AS, the next filter will be tried in the order defined by the failover sequence.

Traffic matchers and path filters are referenced by name. This allows to easily reuse these objects in different traffic policies. The traffic matchers are defined in *Traffic Matchers* and the path filters are defined in *Path Filters*.

Examples

Traffic policy limited to a single ISD

This example shows a configuration with a single traffic policy that configures all traffic to only use paths within the Swiss isolation domain. Refer to the *path filters* and *traffic matchers* examples to see how the corresponding filters and matchers are defined.

```
{
   "traffic_policies": [
      {
         "sequence_id": 0,
         "name": "CH ISD only",
         "description": "No traffic leaves the Swiss ISD.",
         "traffic_matcher": "all",
         "failover_sequence": [
            {
               "sequence_id": 0,
               "path_filter": "ch_isd_only"
            }
         ]
      }
   ]
}
```

Traffic engineering for priority traffic

This example shows a configuration with two traffic policies. The first policy chooses paths via provider 1 for priority traffic (defined as traffic tagged with the EF DSCP flag) and falls back to paths via provider 2 if no paths via provider 1 are available. The second policy chooses paths via provider 2 for all non-priority traffic. If no paths are available via provider 2 no fallback happens and traffic is dropped. Refer to the *path filters* and *traffic matchers* examples to see how the corresponding filters and matchers are defined.

```
{
   "traffic_policies": [
      {
         "sequence_id": 0,
         "name": "Priority traffic"
         "description": "Priority traffic is sent preferred via provider 1",
         "traffic_matcher": "ef_tagged",
         "failover_sequence": [
            {
               "sequence_id": 0,
               "path_filter": "provider_1_outgoing"
            },
            {
               "sequence_id": 1,
               "path_filter": "allow_all"
            },
         ]
      },
```

(continues on next page)

(continued from previous page)

```
      {
         "sequence_id": 1,
         "name": "Non-priority traffic"
         "description": "Non-priority traffic is sent preferred via provider 2",
         "traffic_matcher": "all",
         "failover_sequence": [
            {
               "sequence_id": 0,
               "path_filter": "provider_2_outgoing"
            },
         ]
      },
   ]
}
```

3.4.3 Path Filters

A path filter defines a set of paths by applying the filter to all available paths. The filter is identified by a unique name and can contain an *Access Control List* (ACL) and a *hop pattern* both of which operate on *hop predicates*.

A hop predicate defines a hop of a SCION path and has the form `<isd>-<as>#<interface>`, where `<isd>` is the SCION ISD identifier, `<as>` is the SCION AS identifier, and `<interface>` is the interface identifier of a SCION path hop, i.e., the SCION link in the given SCION AS. The hop predicate can be fully or partially qualified, i.e., all entries of the hop predicate are optional or can include wildcards (`0`). A single `0` matches every SCION path hop.

An *ACL* consists of a list of ACL entries. Each ACL entry has the form `<action> <hop-predicate>`, where the action can either be *accept* (+) or *deny* (-). The hop predicate is optional. If no hop predicate is specified, the action matches every hop, i.e., a single + is the default accept action and a single - is the default deny action. The ACL is applied by sequentially applying all ACL entries to paths. If the ACL is empty, it defaults to accepting all paths.

A *hop pattern* defines a sequence of hop predicates that a path has to match to be accepted. It can be thought of as a regular expression on SCION paths. Each hop predicate can optionally be extended with a modifier * or +. The * modifier means 0 or more occurrences. The + means one or more occurrences.

Path Filters

```
{
   "path_filters": [
      {
         "name": "allow_all",
         "description": "Allow all paths",
         "acl": [
            "+ 0"
         ]
      },
      {
         "name": "ch_isd_only",
         "description": "Only allow paths within the Swiss ISD (64)",
         "acl": [
            "+ 64-0",
            "-"
```

(continues on next page)

(continued from previous page)

```
        ]
      },
      {
         "name": "provider_1_outgoing",
         "description": "Paths that leave the local AS through SCION interface 1",
         "hop_pattern": "1-ff00:1:111#1 0*"
      },
      {
         "name": "provider_1_incoming",
         "description": "Paths that enter the local AS through SCION interface 1",
         "hop_pattern": "0* 1-ff00:1:111#1"
      },
      {
         "name": "provider_2_outgoing",
         "description": "Paths that leave the local AS through SCION interface 2",
         "hop_pattern": "1-ff00:1:111#2 0*"
      }
      {
         "name": "provider_2_incoming",
         "description": "Paths that enter the local AS through SCION interface 2",
         "hop_pattern": "0* 1-ff00:1:111#2"
      }
   ]
}
```

3.4.4 Traffic Matchers

A traffic matcher is used to classify traffic in traffic policies. Each packet is classified based on configured traffic matchers and put in a traffic class. A traffic class is used in a traffic policy to map a failover sequence to a traffic class.

A traffic matcher consists of a unique name (`name`) and a condition (`condition`). The condition defines the matching criteria for the traffic matcher. It is expressed as a boolean expression that then evaluates to either true - the IP packet matches the traffic matcher - or false - the IP packet does not match the traffic matcher. The expression consists of atoms and combinators. The following atoms are supported:

BOOL=<true|false>
: Evaluates to the specified boolean value. This can be used to define a traffic matcher that always matches: `BOOL=true` matches all packets. On the other hand, `BOOL=false` matches no packets.

SRC=<IP-prefix>
: Evaluates to true if `IP-prefix` contains the source IP address of the packet, otherwise false.

DST=<IP-prefix>
: Evaluates to true if `IP-prefix` contains the destination IP address of the packet, otherwise false.

SRCPORT=<port-range>
: Evaluates to true if the source port of the packet is in the range `port-range`, otherwise false. This applies to TCP and UDP packets.

DSTPORT=<port-range>
: Evaluates to true if the destination port of the packet is in the range `port-range`, otherwise false. This applies to TCP and UDP packets.

PROTOCOL=<protocol>
: Evaluates to true if the L4 protocol of the packet is `protocol`, otherwise false. Valid strings for `protocol` are `tcp`, `udp`, and `icmp`.

DSCP=<dscp-value>
: Evaluates to true if the DSCP value of the packet is `dscp-value`, otherwise false. The DSCP value must be specified as a hexadecimal number of the form `0xYY`.

The following combinators are supported:

ANY(cond1, cond2, ...)
: Evaluates to true if any of the `cond` conditions evaluates to true, otherwise false.

ALL(cond1, cond2, ...)
: Evaluates to true if all of the `cond` conditions evaluate to true, otherwise false.

NOT(cond)
: Evaluates to true if `cond` evaluates to false and vice versa.

Traffic Matchers

```
{
   "traffic_matchers": [
      {
         "name": "all",
         "description": "All traffic",
         "condition": "BOOL=true"
      },
      {
         "name": "ef_tagged",
         "description": "Traffic tagged as Expedited Forwarding",
         "condition": "DSCP=0x2e"
      },
      {
         "name": "site1_to_site2",
         "description": "Traffic from Site 1 to Site 2.",
         "condition": "ALL(SRC=10.1.0.0/24, DST=10.2.0.0/24)"
      }
   ]
}
```

3.4.5 Local Endpoint

The local `endpoint` configuration defines the local IP-in-SCION tunneling endpoint. The local endpoint configuration contains the local network endpoint on which IP packets are received and encapsulated. In addition to the local `ip` (host) address, a set of `ports` for data, control, and probing purposes can be configured. However, these ports do not need to be set explicitly as the Anapaya EDGE appliance will automatically assign them if left empty. Both, IPv4 and IPv6 addresses can be configured. Furthermore, SCION RSS can be activated by setting `enable_scion_rss` if the local routers support it. With SCION RSS activated, the gateway utilizes UDP source port entropy on the underlay to enable RSS for SCION traffic, which can drastically increase forwarding performance.

Additionally, the local endpoint configuration contains a set of `allowed_interfaces` for each ISD-AS the appliance is part of, to be used by the IP-in-SCION tunnels involving this appliance. This can be used to control incoming

traffic. For example, if an Anapaya EDGE appliance should only be reachable via links that enter the local site for ISD-AS `1-ff00:1:1` and those links correspond to SCION interfaces 1 and 2, `allowed_interfaces` should list them explicitly. Remote tunnel endpoints will then only choose paths entering the local AS via SCION interface 1 or 2. An empty list means all interfaces can be used by remote IP-in-SCION tunneling endpoints.

Note: Please note that if you specify a set of allowed interfaces for an appliance, you need to ensure that the allowed interfaces are also included in the *Cluster* section of the peer appliances' configuration.

Local Endpoint Configuration

Local endpoint configuration including the host IP address to bind to and limiting the set of interfaces that can be used for the incoming IP-in-SCION tunnels.

```
{
   "endpoint": {
      "enabled": true,
      "ip": "10.8.0.1",
      "allowed_interfaces": [
         {
            "interfaces": [
                  1,
                  2
            ],
            "isd_as": "1-ff00:0:1"
         }
      ],
      "enable_scion_rss": true
   }
}
```

3.4.6 Remote Endpoints

Remote IP-in-SCION tunneling endpoints are automatically discovered within their respective SCION AS. However, the set of remote ASes which should be considered for remote endpoint discovery needs to be statically provided. The `remotes` section contains a list of all remote SCION ASes towards which the local endpoint should establish IP-in-SCION tunnels.

Note: *Routing Domains* also include `remote_isd_ases`, however, that section is used to limit the set of remote ASes to which a domain applies and can include wildcards as well as allow and block lists. If a remote SCION AS is not listed as part of the `remotes` section, it will not be considered during remote endpoint discovery.

Remotes

Consider ASes `1-ff00:0:1` and `2-ff00:0:2` during remote endpoint discovery.

```
{
   "remotes": [
      {
```

(continues on next page)

(continued from previous page)

```
            "isd_as": "1-ff00:0:1",
            "description": "Branch office in Zurich"
        },
        {
            "isd_as": "2-ff00:0:2",
            "description": "Branch office in London"
        }
    ]
}
```

3.4.7 Static Announcements

The IP prefixes that will be statically announced are defined in the `static_announcements` section. Each entry defines a list of `prefixes` to announce.

Note: The set of prefixes that are announced to any given remote tunnel endpoint is not necessarily the statically defined `prefixes` list. Configured announce-filters in each domain can limit the set of announced prefixes in a given domain.

Optionally, next-hop tracking can be enabled for a set of `prefixes`. If next-hop tracking is enabled, the `prefixes` are only distributed if the specified `target` responds to ICMP ECHO requests. This can be used to implement dynamically retractable routes without having to resort to a dynamic routing protocol.

Static IP Prefixes

Statically announce `10.8.0.2/32` and `10.8.0.5/32` if `10.8.0.2` is reachable and unconditionally announce `10.8.0.1/32`.

```
{
    "static_announcements": [
        {
            "next_hop_tracking": {
                "enabled": true,
                "target": "10.8.0.2"
            },
            "prefixes": [
                "10.8.0.2/32",
                "10.8.0.5/32"
            ],
            "sequence_id": 0
        },
        {
            "prefixes": [
                "10.8.0.1/32"
            ],
            "sequence_id": 1
        },
    ]
}
```

3.4.8 Use Cases

This section contains common IP-in-SCION tunneling use cases and how they can be implemented using the Anapaya EDGE appliance's domain-based routing configuration. There are of course many more use cases and implementation options. The examples below are intended to illustrate common use cases and should serve the reader as a basis for implementation and adaptation.

Company WAN and Public Cloud SaaS Access

In this use case, the operator has two Anapaya EDGE appliances (**edge1** and **edge2**) in two local data center sites, each with one SCION access link to an upstream SCION ISP. This is a typical deployment for a regional or global headquarter site (**HQ Zurich**). The operator has to consider the company's remote sites (**Branch Paris** and **Branch London**) that are part of the company's wide area network (WAN). Furthermore, the operator wants to configure access to two Software-as-a-Service (SaaS) applications that run in a public cloud. As the public cloud is not directly SCION-enabled, the company makes use of two service providers that provide SCION-enabled access to the public cloud (**Cloud Access Provider**). Please refer to the figure when working through the examples.

This use case can be naturally implemented with two domains: one domain for the company's WAN and a separate domain to configure public cloud access.

Domains Configuration Skeleton

```
"domains": [
    {
        "name": "company_wan",
        "description": "The company's wide area network.",
    },
    {
        "name": "public_cloud",
        "description": "Access to SaaS applications in a public cloud.",
    }
]
```

Domain Entities

The first step is to define *what* entities are part of the domains, i.e., announced and accepted IP prefixes and remote SCION ASes.

Let us first focus on the `company_wan` domain and assume the following requirements:

- Possible IP ranges for the company's WAN: `10.0.0.0/8` and `192.168.0.0/16`.
- The local sites should only announce prefixes in `10.1.0.0/16` and `10.2.0.0/16`
- Only the remote sites `2-ff00:0:2` (Branch Paris) and `2-ff00:0:3` (Branch London) should be part of the `company_wan` domain.
- IP-in-SCION tunnels should be established to both these branches from **HQ Zurich**.

Putting all this together, we get the following configuration for the `company_wan` domain:

Company WAN Domain with Configured Remotes

```
"scion_tunneling": {
    "domains": [
        {
            "name": "company_wan",
            "description": "The company's wide area network.",
            "prefixes": {
                "accept_filter": [
                    {
                        "prefixes": ["10.0.0.0/8", "192.168.0.0/16"],
                        "action": "ACCEPT",
                        "sequence_id": 0
                    }
                ],
                "announce_filter": [
                    {
                        "prefixes": ["10.1.0.0/16", "10.2.0.0/16"],
                        "action": "ACCEPT",
                        "sequence_id": 0
                    }
                ]
            },
            "remote_isd_ases": [
                {
                    "isd_as": "2-ff00:0:2",
                    "action": "ACCEPT",
                    "sequence_id": 0
                },
                {
                    "isd_as": "2-ff00:0:3",
                    "action": "ACCEPT",
                    "sequence_id": 1
                }
            ]
        }
    ],
    "remotes": [
        {
            "isd_as": "2-ff00:0:2",
            "description": "Branch office in Paris"
        },
        {
            "isd_as": "2-ff00:0:3",
            "description": "Branch office in London"
        }
    ]
}
```

Next, the operator needs to define the *what* entities that are part of the `public_cloud` domain. The following requirements are assumed:

- The public cloud is accessible via the Cloud Access Providers (CAP) `2-ff00:0:100`, `1-ff00:0:200`, and `2-ff00:0:200`. Note that the latter two ASes are distinct on the SCION level, but obviously they belong to the same CAP.
- The IP prefix assigned **app1** is `1.0.1.0/24` and the one assigned to **app2** is `1.0.2.0/24`, therefore these prefixes must be accepted.
- All internal hosts are NATed to an IP in `2.0.0.0/24`, i.e., this is the IP prefix that needs to be announced in the `public_cloud` domain, such that the return traffic can be correctly routed from the public cloud to the local site.

Putting all this together, we get the following configuration for the `public_cloud` domain:

Public Cloud Domain Entities

```
"scion_tunneling": {
    "domains": [
        {
            "name": "public_cloud",
            "description": "Access to SaaS applications in a public cloud.",
            "prefixes": {
                "accept_filter": [
                    {
                        "prefixes": ["1.0.1.0/24", "1.0.2.0/24"],
                        "action": "ACCEPT",
                        "sequence_id": 0
                    }
                ],
                "announce_filter": [
                    {
                        "prefixes": ["2.0.0.0/24"],
                        "action": "ACCEPT",
                        "sequence_id": 0
                    }
                ]
            },
            "remote_isd_ases": [
                {
                    "isd_as": "1-ff00:0:100",
                    "action": "ACCEPT",
                    "sequence_id": 0
                },
                {
                    "isd_as": "0-ff00:0:200",
                    "action": "ACCEPT",
                    "sequence_id": 1
                }
            ]
        }
    ],
    "remotes": [
        {
            "isd_as": "1-ff00:0:100",
            "description": "Access provider 1 to public cloud"
```

(continues on next page)

(continued from previous page)

```
        },
        {
            "isd_as": "1-ff00:0:200",
            "description": "Access provider 2 to public cloud (in ISD 1)"
        },
        {
            "isd_as": "2-ff00:0:200",
            "description": "Access provider 2 to public cloud (in ISD 2)"
        }
    ]
}
```

Note the use of the ISD wildcard `0-ff00:0:200` in the `remote_isd_ases` section. This matches the AS identifier `ff00:0:200` in every ISD, however, since the operator only explicitly adds the ASes `1-ff00:0:200` and `2-ff00:0:200` in the `remotes` section, only those will be used to establish tunnels with.

Announced Prefixes

The prefixes defined in the `announce_filter` of the domains are not the prefixes that are necessarily announced. Instead, these is a filter applied on statically configured and dynamically learned prefixes (via BGP). In this example, both Anapaya EDGE appliances are connected internally to a BGP peer that announces prefixes to and accepts prefixes from the Anapaya EDGE appliances. To learn how to configure BGP peers on an Anapaya EDGE appliance, please refer to the *Border Gateway Protocol (BGP)* section.

Traffic Policies

Besides defining the entities that are part of the domains, the operator also needs to configure at least one traffic policy per domain.

Note: Often, a permissive traffic policy, i.e., a traffic policy that does not limit the set of acceptable SCION paths too much, is advantageous. The Anapaya EDGE appliance has a powerful path optimization engine that can ensure optimal path selection amongst the set of acceptable paths.

The requirements are as follows:

- Generally, reliability and performance should be maximized.
- **app1** must only be accessible via paths within ISD 1 as it deals with highly sensitive and regulated data.
- There are no specific requirements for **app2**.

Note: The operator has little influence over the traffic policies used by the remote endpoints. To ensure that the traffic for **app1** also only uses paths within ISD 1 on the return path, the operator needs to coordinate that with the remote endpoint. If the operator also controls the remote endpoint then this is not an issue.

Given these requirements, the operator first needs to define three traffic matchers: one that matches traffic belonging to app1, one that matches traffic belonging to app2, and a third that matches any traffic.

Traffic Matchers

```
"traffic_matchers": [
    {
        "name": "app1",
        "description": "Traffic belonging to app1",
        "condition": "DST=1.0.1.0/24"
    },
    {
        "name": "app2",
        "description": "Traffic belonging to app2",
        "condition": "DST=1.0.2.0/24"
    },
    {
        "name": "all",
        "description": "All traffic",
        "condition": "BOOL=true"
    }
]
```

Furthermore, two path filters need to be defined - one that only allows paths within ISD 1 and one that allows all paths.

Path Filters

```
"path_filters": [
    {
        "name": "isd_1_only",
        "description": "Only allow paths within ISD 1",
        "acl": ["+ 1-0", "-"]
    },
    {
        "name": "allow_all",
        "description": "Allow all paths",
        "acl": ["+ 0"]
    }
]
```

Having defined the traffic matchers and path filters, the operator can now assemble the traffic policies for the domains.

Traffic Policies

```
"domains": [
    {
        "name": "company_wan",
        "traffic_policies": [
            {
                "sequence_id": 0,
                "description": "Allow all traffic to use all available paths.",
                "traffic_matcher": "all",
                "failover_sequence": [
                    {
                        "sequence_id": 0,
```

(continues on next page)

(continued from previous page)

```
                        "path_filter": "allow_all"
                    }
                ]
            }
        ]
    },
    {
        "name": "public_cloud",
        "traffic_policies": [
            {
                "sequence_id": 0,
                "description": "Traffic belonging to app1 can only use paths within ISD␣
↪1.",
                "traffic_matcher": "app1",
                "failover_sequence": [
                    {
                        "sequence_id": 0,
                        "path_filter": "isd_1_only"
                    }
                ]
            },
            {
                "sequence_id": 1,
                "description": "Traffic belonging to app2 can use all paths",
                "traffic_matcher": "app2",
                "failover_sequence": [
                    {
                        "sequence_id": 0,
                        "path_filter": "allow_all"
                    }
                ]
            }
        ]
    }
]
```

Local Endpoint

To finalize the IP-in-SCION tunneling configuration, the operator needs to configure the local tunneling endpoint for each Anapaya EDGE appliance. This is the part of the configuration that will be different for each appliance. Besides the host address to which the tunneling endpoint is bound, the operator also wants to control incoming traffic such that traffic coming via SCION interface 1 is always handled by `edge1`, and traffic coming via SCION interface 2 is always handled by `edge2`. To that end, the operator needs to specify `allowed_interfaces` accordingly.

Local Endpoint Configuration

edge1:

```
"endpoint": {
    "ip": "10.1.0.1",
```

(continues on next page)

(continued from previous page)

```
    "allowed_interfaces": [
        {
           "interfaces": [
                1
           ],
           "isd_as": "1-ff00:1:1"
        }
    ],
    "enabled": true
},
```

edge2:

```
"endpoint": {
   "ip": "10.1.0.2",
   "allowed_interfaces": [
      {
         "interfaces": [
              2
         ],
         "isd_as": "1-ff00:1:1"
      }
   ],
   "enabled": true
}
```

Complete Configuration

For completeness the entire `scion_tunneling` configurations for `edge1` is shown below. The one for `edge2` only differs in the `endpoint` section.

IP-in-SCION tunneling configuration for edge1

```
{
    "scion_tunneling": {
        "endpoint": {
            "ip": "10.1.0.1",
            "allowed_interfaces": [
                {
                   "interfaces": [
                        1
                   ],
                   "isd_as": "1-ff00:1:1"
                }
            ],
            "enabled": true
        },
        "domains": [
            {
```

(continues on next page)

(continued from previous page)

```
                "name": "company_wan",
                "description": "The company's wide area network.",
                "prefixes": {
                    "accept_filter": [
                        {
                            "prefixes": ["10.0.0.0/8", "192.168.0.0/16"],
                            "action": "ACCEPT",
                            "sequence_id": 0
                        }
                    ],
                    "announce_filter": [
                        {
                            "prefixes": ["10.1.0.0/16", "10.2.0.0/16"],
                            "action": "ACCEPT",
                            "sequence_id": 0
                        }
                    ]
                },
                "remote_isd_ases": [
                    {
                        "isd_as": "2-ff00:0:2",
                        "action": "ACCEPT",
                        "sequence_id": 0
                    },
                    {
                        "isd_as": "2-ff00:0:3",
                        "action": "ACCEPT",
                        "sequence_id": 1
                    }
                ],
                "traffic_policies": [
                    {
                        "sequence_id": 0,
                        "description": "Allow all traffic to use all available␣
↪paths.",
                        "traffic_matcher": "all",
                        "failover_sequence": [
                            {
                                "sequence_id": 0,
                                "path_filter": "allow_all"
                            }
                        ]
                    }
                ]
            },
            {
                "name": "public_cloud",
                "description": "Access to SaaS applications in a public cloud.
↪",
                "prefixes": {
                    "accept_filter": [
                        {
```

(continues on next page)

(continued from previous page)

```
                            "prefixes": ["1.0.1.0/24", "1.0.2.0/24"],
                            "action": "ACCEPT",
                            "sequence_id": 0
                        }
                    ],
                    "announce_filter": [
                        {
                            "prefixes": ["2.0.0.0/24"],
                            "action": "ACCEPT",
                            "sequence_id": 0
                        }
                    ]
                },
                "remote_isd_ases": [
                    {
                        "isd_as": "1-ff00:0:100",
                        "action": "ACCEPT",
                        "sequence_id": 0
                    },
                    {
                        "isd_as": "0-ff00:0:200",
                        "action": "ACCEPT",
                        "sequence_id": 1
                    }
                ],
                "traffic_policies": [
                    {
                        "sequence_id": 0,
                        "description": "Traffic belonging to app1 can only use␣
↪paths within ISD 1.",
                        "traffic_matcher": "app1",
                        "failover_sequence": [
                            {
                                "sequence_id": 0,
                                "path_filter": "isd_1_only"
                            }
                        ]
                    },
                    {
                        "sequence_id": 1,
                        "description": "Traffic belonging to app2 can use all␣
↪paths",
                        "traffic_matcher": "app2",
                        "failover_sequence": [
                            {
                                "sequence_id": 0,
                                "path_filter": "allow_all"
                            }
                        ]
                    }
                ]
            }
```

(continues on next page)

(continued from previous page)

```
        ],
        "remotes": [
            {
                "isd_as": "2-ff00:0:2",
                "description": "Branch office in Paris"
            },
            {
                "isd_as": "2-ff00:0:3",
                "description": "Branch office in London"
            },
            {
                "isd_as": "1-ff00:0:100",
                "description": "Access provider 1 to public cloud"
            },
            {
                "isd_as": "1-ff00:0:200",
                "description": "Access provider 2 to public cloud (in ISD 1)"
            },
            {
                "isd_as": "2-ff00:0:200",
                "description": "Access provider 2 to public cloud (in ISD 2)"
            }
        ],
        "traffic_matchers": [
            {
                "name": "app1",
                "description": "Traffic belonging to app1",
                "condition": "DST=1.0.1.0/24"
            },
            {
                "name": "app2",
                "description": "Traffic belonging to app2",
                "condition": "DST=1.0.2.0/24"
            },
            {
                "name": "all",
                "description": "All traffic",
                "condition": "BOOL=true"
            }
        ],
        "path_filters": [
            {
                "name": "isd_1_only",
                "description": "Only allow paths within ISD 1",
                "acl": ["+ 1-0", "-"]
            },
            {
                "name": "allow_all",
                "description": "Allow all paths",
                "acl": ["+ 0"]
            }
        ]
```

(continues on next page)

(continued from previous page)

```
    }
}
```

Multi-ISD Anapaya EDGE Setup

This example shows a single Anapaya EDGE appliance (**edge**) that is part of two ISDs. One use case for this setup is having an appliance that is part of an ISD that is completely disconnected from the rest of the public SCION Internet, but needs to be remotely managed. In this case, a second ISD that is connected to the public SCION Internet needs to be configured on the appliance, such that it can be remotely managed. Another use case is if the Anapaya EDGE appliance has two upstream SCION ISPs that are in different ISDs themselves. This is only possible if the Anapaya EDGE appliance is also part of both those ISDs.

The following example shows an Anapaya EDGE appliance that is part of two separate and isolated ISDs (**ISD 1** and **ISD 2**). The requirements are as follows:

- Only IP prefixes in `1.0.0.0/8` are announced and accepted to and from ISD 1.
- Only IP prefixes in `2.0.0.0/8` are announced and accepted to and from ISD 2.
- Only remote ASes that are part of the corresponding ISD can announce these prefixes to the local Anapaya EDGE appliance.
- The Anapaya EDGE appliance should use its local AS identity that corresponds the ISD when communicating with other IP-in-SCION tunneling endpoints in the same ISD.
- IP prefix `1.0.1.0/24` should be announced to ISD 1 and `2.0.1.0/24` to ISD 2 if (and only if) internally the **firewall** at `192.168.2.100` is reachable.

Given these requirements, the operator defines two routing domains: one for ISD 1 and one for ISD 2.

Domains Configuration Skeleton

```
"domains": [
    {
        "name": "isd_1",
        "description": "Domain for communication in ISD 1.",
    },
    {
        "name": "isd_2",
        "description": "Domain for communication in ISD 2.",
    }
]
```

Domain Entities

The first step is to define *what* entities are part of the domains, i.e., announced and accepted IP prefixes and remote SCION ASes.

Let us first focus on the `isd_1` domain. As described above, the operator wants to ensure that only IP prefixes in `1.0.0.0/8` are announced to and accepted from remote IP-in-SCION tunneling endpoints in ISD 1. Furthermore, the operator wants to limit the set of remote ASes to those that are part of ISD 1 and use the local AS identity in ISD 1 for communication in this domain. Finally, the remote site **Remote 1** (`1-ff00:0:2`) needs to be explicitly added to the `remotes` section such that the Anapaya EDGE appliance will establish IP-in-SCION tunnels to endpoints in **Remote 1**.

ISD1 Domain with Configured Remotes

```
"scion_tunneling": {
    "domains": [
        {
            "name": "isd_1",
            "description": "Domain for communication in ISD 1.",
            "prefixes": {
                "accept_filter": [
                    {
                        "prefixes": ["1.0.0.0/8"],
                        "action": "ACCEPT",
                        "sequence_id": 0
                    }
                ],
                "announce_filter": [
                    {
                        "prefixes": ["1.0.0.0/8"],
                        "action": "ACCEPT",
                        "sequence_id": 0
                    }
                ]
            },
            "local_isd_ases": ["1-ff00:0:1"],
            "remote_isd_ases": [
                {
                    "isd_as": "1-0",
                    "action": "ACCEPT",
                    "sequence_id": 0
                }
            ]
        }
    ],
    "remotes": [
        {
            "isd_as": "1-ff00:0:2",
            "description": "Remote AS in ISD 1"
        }
    ]
}
```

The `isd_2` domain is configured similarly, with the straightforward adaption of the configuration to the requirements of ISD 2.

ISD2 Domain with Configured Remotes

```
"scion_tunneling": {
    "domains": [
        {
            "name": "isd_2",
            "description": "Domain for communication in ISD 2.",
            "prefixes": {
                "accept_filter": [
                    {
                        "prefixes": ["2.0.0.0/8"],
                        "action": "ACCEPT",
                        "sequence_id": 0
                    }
                ],
                "announce_filter": [
                    {
                        "prefixes": ["2.0.0.0/8"],
                        "action": "ACCEPT",
                        "sequence_id": 0
                    }
                ]
            },
            "local_isd_ases": ["2-ff00:0:1"],
            "remote_isd_ases": [
                {
                    "isd_as": "2-0",
                    "action": "ACCEPT",
                    "sequence_id": 0
                }
            ]
        }
    ],
    "remotes": [
        {
            "isd_as": "2-ff00:0:3",
            "description": "Remote AS in ISD 2"
        }
    ]
}
```

Announced Prefixes

As described in the introduction, the Anapaya EDGE appliance should announce the local prefixes `1.0.1.0/24` in ISD 1 and `2.0.1.0/24` in ISD 2 if internally the firewall at `192.168.2.100` is reachable. This can be accomplished by configuring the `static_announcements` section with `next_hop_tracking` activated. Note that the `static_anouncements` section contains both IP prefixes. The `announce_filter` in the corresponding domain configurations ensure that `1.0.1.0/24` is announce in ISD 1 and `2.0.1.0/24` in ISD 2.

Static Announcements

```
"static_announcements": [
    {
        "next_hop_tracking": {
            "enabled": true,
            "target": "192.168.2.100"
        },
        "prefixes": ["1.0.1.0/24", "2.0.1.0/24"],
        "sequence_id": 0
    }
]
```

Traffic Policies

Besides defining the entities that are part of the domains, the operator also needs to configure at least one traffic policy per domain.

Note: Often, a permissive traffic policy, i.e., a traffic policy that does not limit the set of acceptable SCION paths too much, is advantageous. The Anapaya EDGE appliance has a powerful path optimization engine that can ensure optimal path selection amongst the set of acceptable paths.

In this example, the operator wants to ensure that traffic for destinations in ISD 1 only uses paths in ISD 1 and similarly for destinations in ISD 2. Besides that, the operator does not want to further limit the set of acceptable paths to give the Anapaya EDGE appliance the biggest possible path selection scope.

Note: If the two ISDs are truly isolated from the rest of the SCION Internet, limiting the set of acceptable paths to those in the same ISD is not required as no other paths exist anyway. Nevertheless, it is good practice to be specific about the set of acceptable paths.

To implement these requirements, the operator first needs to define one traffic matcher to match all traffic and two path filters to match paths in ISD 1 and ISD 2 respectively.

Traffic Matchers and Path Filters

```
"traffic_matchers": [
    {
        "name": "all",
        "description": "All traffic",
        "condition": "BOOL=true"
```

(continues on next page)

(continued from previous page)

```
    }
],
"path_filters": [
    {
        "name": "isd_1_only",
        "description": "Only allow paths within ISD 1",
        "acl": ["+ 1-0", "-"]
    },
    {
        "name": "isd_2_only",
        "description": "Only allow paths within ISD 2",
        "acl": ["+ 2-0", "-"]
    }
]
```

Using the defined traffic matcher and path filters, the operator can define the traffic policies for each domain:

Traffic Policies

```
"domains": [
    {
        "name": "isd_1",
        "traffic_policies": [
            {
                "sequence_id": 0,
                "description": "All traffic must only use paths within ISD 1.",
                "traffic_matcher": "all",
                "failover_sequence": [
                    {
                        "sequence_id": 0,
                        "path_filter": "isd_1_only"
                    }
                ]
            }
        ]
    },
    {
        "name": "isd_2",
        "traffic_policies": [
            {
                "sequence_id": 0,
                "description": "All traffic must only use paths within ISD 2.",
                "traffic_matcher": "all",
                "failover_sequence": [
                    {
                        "sequence_id": 0,
                        "path_filter": "isd_2_only"
                    }
                ]
            }
        ]
```

(continues on next page)

(continued from previous page)

```
    }
]
```

Local Endpoint

To finalize the IP-in-SCION tunneling configuration, the operator needs to configure the local tunneling endpoint for the Anapaya EDGE appliance. Since the operator does not need to control incoming traffic, the only required configuration is to specify the host address to bind to and to enable the IP-in-SCION tunneling endpoint.

Local Endpoint Configuration

```
"endpoint": {
    "ip": "192.168.2.1",
    "enabled": true
},
```

Complete Configuration

For completeness the entire `scion_tunneling` configurations for `edge` is shown below.

Complete IP-in-SCION tunneling configuration

```
{
    "scion_tunneling": {
        "endpoint": {
            "ip": "192.168.2.1",
            "enabled": true
        },
        "domains": [
            {
                "name": "isd_1",
                "description": "Domain for communication in ISD 1.",
                "prefixes": {
                    "accept_filter": [
                        {
                            "prefixes": ["1.0.0.0/8"],
                            "action": "ACCEPT",
                            "sequence_id": 0
                        }
                    ],
                    "announce_filter": [
                        {
                            "prefixes": ["1.0.0.0/8"],
                            "action": "ACCEPT",
                            "sequence_id": 0
                        }
```

(continues on next page)

(continued from previous page)

```
                    ]
                },
                "local_isd_ases": ["1-ff00:0:1"],
                "remote_isd_ases": [
                    {
                        "isd_as": "1-0",
                        "action": "ACCEPT",
                        "sequence_id": 0
                    }
                ],
                "traffic_policies": [
                    {
                        "sequence_id": 0,
                        "description": "All traffic must only use paths within␣
↪ISD 1.",
                        "traffic_matcher": "all",
                        "failover_sequence": [
                            {
                                "sequence_id": 0,
                                "path_filter": "isd_1_only"
                            }
                        ]
                    }
                ]
            },
            {
                "name": "isd_2",
                "description": "Domain for communication in ISD 2.",
                "prefixes": {
                    "accept_filter": [
                        {
                            "prefixes": ["2.0.0.0/8"],
                            "action": "ACCEPT",
                            "sequence_id": 0
                        }
                    ],
                    "announce_filter": [
                        {
                            "prefixes": ["2.0.0.0/8"],
                            "action": "ACCEPT",
                            "sequence_id": 0
                        }
                    ]
                },
                "local_isd_ases": ["2-ff00:0:1"],
                "remote_isd_ases": [
                    {
                        "isd_as": "2-0",
                        "action": "ACCEPT",
                        "sequence_id": 0
                    }
                ],
```

(continues on next page)

(continued from previous page)

```
                "traffic_policies": [
                    {
                        "sequence_id": 0,
                        "description": "All traffic must only use paths within␣
↪ISD 2.",
                        "traffic_matcher": "all",
                        "failover_sequence": [
                            {
                                "sequence_id": 0,
                                "path_filter": "isd_2_only"
                            }
                        ]
                    }
                ]
            }
        ],
        "remotes": [
            {
                "isd_as": "1-ff00:0:2",
                "description": "Remote AS in ISD 1"
            },
            {
                "isd_as": "2-ff00:0:3",
                "description": "Remote AS in ISD 2"
            }
        ],
        "traffic_matchers": [
            {
                "name": "all",
                "description": "All traffic",
                "condition": "BOOL=true"
            }
        ],
        "path_filters": [
            {
                "name": "isd_1_only",
                "description": "Only allow paths within ISD 1",
                "acl": ["+ 1-0", "-"]
            },
            {
                "name": "isd_2_only",
                "description": "Only allow paths within ISD 2",
                "acl": ["+ 2-0", "-"]
            }
        ],
        "static_announcements": [
            {
                "next_hop_tracking": {
                    "enabled": true,
                    "target": "192.168.2.100"
                },
                "prefixes": ["1.0.1.0/24", "2.0.1.0/24"],
```

(continues on next page)

(continued from previous page)

```
                "sequence_id": 0
            }
        ]
    }
}
```

Configuring Source NAT to collect Users from the Anapaya GATE

In this use case, the operator wants to ensure that traffic streams coming from an IP-in-SCION tunnel from a remote Anapaya GATE are returned via the local Anapaya EDGE appliance, even though the remote IP prefixes are also reachable via the legacy Internet.

To address this use case, the EDGE appliance can do source NAT on the incoming traffic. Source addresses in packets arriving from the tunnel are rewritten by one of the addresses from the NAT's address pool.

The operator is responsible for setting up the static routes so that replies to the addresses in the NAT's address pool are routed back to the appliance. This way the operator can ensure that replies to the packets arriving via the tunnel are routed back to the tunnel, even if the remote IP prefixes are also reachable via the legacy Internet.

If, on the other hand, a packet arrives through the Internet it keeps its original source address and the reply is routed back through the legacy Internet. Again, it is up to the operator to set up the routes in the local network accordingly.

Source NAT Configuration

```
"nat": {
   "snat": {
      "address_pool": ["192.168.0.1/32"],
      "exclude": [],
      "interfaces": ["eth0"]
   }
}
```

Note: Source NAT does not work with BGP enabled in the appliance.

Warning: When using source NAT it is not possible by default to create connections from the local AS to the remote AS that would go through the IP-in-SCION tunnel. Using NAT means that there are no fixed IP addresses to connect to. If this is needed, a specific IP address, or a range of addresses, can be excluded from the NAT. The packet will then arrive from the tunnel with its original source address unmodified. In this case it's up to the operator to route the packets destined to that address to the SCION tunnel rather than routing them through the legacy Internet.

3.5 Network Interfaces

The physical and/or virtual network interfaces of the appliance are configured in the `interfaces` section of the appliance configuration. Note that these network interfaces are different from the SCION interfaces - a SCION interface is a purely logical interface that always binds to a physical or virtual network interface, usually a WAN interface. In addition to WAN interfaces, an appliance can have multiple LAN interfaces.

All interface types have standard configuration properties, such as the `addresses` assigned to the interface, `routes` specific to the interface, `mtu`, etc. The `gateway` field must only be set on one interface. It is a shortcut to configure the default routes `0.0.0.0/0` and/or `::/0` on a particular interface. If a custom `metric` or `from` is necessary for the default route, configure it using the `routes` section instead.

Additionally, most interface types can be configured to be part of a virtual router using the Virtual Router Redundancy Protocol (VRRP). For more details about the exact configuration entries for each interface type, please refer to the Appliance Management API specification.

There is a wide range of network interface types supported:

3.5.1 Ethernet Interfaces

An Ethernet interface is a physical network interface. Availability depends on the platform the appliance runs on. Please consult the platform documentation to find out what physical interfaces are available. Alternatively, the /api/v1/network/physical-interfaces API endpoint can be used to retrieve the available physical interfaces.

The interface `driver` can be explicitly configured. The following options are available:

- `LINUX`: The interface is bound by Linux with the native driver. This driver should be selected for management interfaces, or if there is no DPDK driver support in VPP for the physical interface (e.g., old hardware or WIFI interfaces). Using this driver for management interfaces is strongly recommended to guarantee management access even if the performance optimized dataplane is misbehaving.
- `VPP_DPDK`: The interface driver is bound by VPP with the bundled DPDK driver. This driver is optimized for performance and should be selected for interfaces that service dataplane traffic whenever possible. With this driver, you might also need to configure the `vlan-strip-offload` mode based your configuration. See the VPP documentation for more information.
- `VPP_VMXNET3`: Same as `VPP_DPDK` but for VMXNET3 interfaces. This driver is only available on VMware ESXi hosts.
- `VPP_MEMIF`: This driver can be used to communicate over shared memory with another client using memif. This driver is only available on Linux hosts and unless you know what you are doing, you should not use this driver.

If no driver is configured, the default driver is `VPP_DPDK`.

Separate LAN, WAN and management interfaces

```
{
  "ethernets": [
    {
      "addresses": [
        "10.10.0.2/24"
      ],
      "name": "lan",
      "gateway": {
        "ipv4_gateway": "10.10.0.1"
      },
```

(continues on next page)

(continued from previous page)

```
      "driver": "VPP_DPDK"
    },
    {
      "addresses": [
        "169.254.10.2/30"
      ],
      "name": "wan",
      "driver": "VPP_DPDK"
    },
    {
      "addresses": [
        "192.168.1.2/24"
      ],
      "name": "mgmt",
      "driver": "LINUX"
    }
  ]
}
```

This examples shows a configuration for an appliance with three physical interfaces:

- Interface `lan` configures the LAN access with the address 10.10.0.2. We also configure the IPv4 gateway on this interface via 10.10.0.1. This means that a default route 0.0.0.0/0 via 10.10.0.1 is installed. The interface driver is `VPP` for performance, and we will use this interface to deliver traffic towards the AS internal network.
- Interface `wan` configures the WAN access with the link-local address 169.254.10.2. The interface driver is `VPP` for performance, and we will use this interface to deliver traffic towards the neighboring AS.
- Interface `mgmt` configures the management access with the address 192.168.1.2. Because we use this interface for management, we do not bind it in VPP and use the `LINUX` driver. We will use this interface for management traffic, e.g., to interact with the management API.

3.5.2 Virtual Functions

A Virtual Function (VF) is a virtual network interface that can be configured on top of a physical interface supporting SR-IOV. Please consult the platform documentation to find out if and which interfaces support the creation of VFs. A VF is always bound to a physical interface that is configured in the `interfaces.ethernets` section (specified by the `link` property). This physical interface must use the `LINUX` driver.

Virtual Function

```
{
  "ethernets": [
    {
      "name": "ens1",
      "driver": "VPP_DPDK"
    }
  ],
  "virtual_functions": [
```

(continues on next page)

(continued from previous page)

```
    {
      "addresses": [
        "169.254.10.2/30"
      ],
      "link": "ens1",
      "name": "ens1vf0"
    }
  ]
}
```

This example shows a configuration for an appliance with a physical interface that supports SR-IOV. The `ens1vf0` VF on top of this interface.

3.5.3 VLAN Interfaces

A VLAN interface is a logical interface that is bound to an underlying network interface. A VLAN interface can be configured on top of `ethernets`, `virtual_functions` and `bonds`. The underlying network interface is specified by the `link` property. The name of the VLAN interface must be of the form `<link>.<id>`, e.g., `eth0.42` where `eth0` is the name of the physical interface, and `42` is the VLAN ID.

VLAN

```
{
  "ethernets": [
    {
      "name": "eth0",
      "driver": "VPP_DPDK"
    }
  ],
  "vlans": [
    {
      "addresses": [
        "169.254.10.2/30"
      ],
      "link": "eth0",
      "name": "eth0.42"
    }
  ]
}
```

This example shows a configuration for an appliance with a VLAN interface on the ethernet interface `eth0` with the VLAN ID 42.

Warning: When changing the MAC address of a VLAN's link interface, the VLAN interface will be recreated. This means that the VLAN interface will disappear for a short period.

3.5.4 Bond Interfaces

A bond interface groups multiple network interfaces together to a single logical network interface. Bonding can offer performance improvements and increased redundancy. A bond can only consist of `ethernet` interfaces. Additionally, all `ethernet` interfaces that are part of a bond must use the `VPP` driver.

Bond

```
{
  "bonds": [
    {
      "addresses": [
        "169.254.10.2/30"
      ],
      "interfaces": [
        "eno5",
        "eno6"
      ],
      "name": "bond0"
    }
  ],
  "ethernets": [
    {
      "name": "eno5",
      "driver": "VPP_DPDK"
    },
    {
      "name": "eno6",
      "driver": "VPP_DPDK"
    }
  ]
}
```

This example shows a configuration of an appliance that bonds two interfaces together. The bond is created over two virtual functions.

3.5.5 Loopback Interfaces

A loopback interface is a logical network interface that a host can use to communicate with itself. Each loopback interface must be configured with a list of IP prefixes and the name of the interface must start with `loop`, e.g., `loop0`. Note that a default loopback interface with IP prefixes `127.0.0.0/8` and `::1/128` is automatically created for each appliance host and must not be configured.

Loopback interfaces can be useful in various scenarios. We will mention two examples here:

- In a single appliance setup where you do not need the SCION services to be reachable from the internal network, you can configure a loopback and use it for the SCION control and router addresses. Even if the SCION services are not reachable from the outside, you can still configure the appliance to provide IP-in-SCION tunneling between your internal network to the remote SCION ASes.
- You might want the SCION services of an appliance to be reachable via multiple interfaces for redundancy on the network interface level. In that case, you can configure a loopback and bind the SCION services to that address.

This can be paired with BGP such that the chosen loopback address is announced to the internal network. If no dynamic routing protocol is used, you will need to install the routes appropriately in the internal network.

Loopback

```
{
  "loopbacks": [
    {
      "addresses": [
        "10.10.0.1/32"
      ],
      "name": "loop0"
    }
  ]
}
```

3.5.6 Wireguard Interfaces

A wireguard interface is a virtual network interface using the WireGuard protocol to establish a secure tunnel to other wireguard peers. When configuring a wireguard interface, the appliance automatically generates a private key and a public key. The public key is then exposed in the configuration as the `public_key` property of wireguard interface entry, which you can request via the Appliance Management API.

To add a new wireguard peer, a `peer` entry needs to be configured containing the network endpoint of the form `host:port` as well as the public key of the peer.

Wireguard

```
{
  "wireguards": [
    {
      "addresses": [
        "10.10.0.1/32"
      ],
      "name": "wg0",
      "port": 51021,
      "peers": [
        {
          "endpoint": "10.10.0.1:51021",
          "public_key": "<public key>"
        }
      ]
    }
  ]
}
```

3.6 Network Address Translation (NAT)

The `nat` section exposes the configuration of NAT.

3.6.1 Source NAT

Source NAT is useful when replies to the packets coming out from a SCION tunnel are supposed to be routed back to the tunnel while other packets can still be routed in an arbitrary user-defined way.

Using source NAT assumes that the routing of packets to the appliance is done using static routes. Combining source NAT with BGP is not supported.

Note: The supported source NAT feature is not for outgoing traffic. Instead, source NAT is supported for incoming traffic. The motivation for this is so that return traffic can be sent via the EDGE if there is also an Internet router available for it.

`nat.snat.address_pool`
: A list of IPv4 prefixes to specify which addresses can be used for the NAT. An incoming packet's source address will be replaced by one of these addresses. It is up to the user to route the reply packets sent to these addresses back to the appliance.

`nat.snat.exclude`
: A list of IPv4 prefixes to exclude from the NAT. A packet with source IP address covered by one of these prefixes will be passed as is without rewriting its source address. The number of addresses to exclude is limited to 1000000.

`nat.snat.interfaces`
: A list of network interfaces to apply source NAT to. These are typically the interfaces connected to the local network.

3.7 Border Gateway Protocol (BGP)

The Anapaya appliance supports integration with the Border Gateway Protocol (BGP). There are two use cases for this:

1. **LAN connectivity:** The local network is administered using BGP and the appliance needs to learn about reachable IP destinations in the local network via BGP.
2. **Route redistribution:** The IP prefixes learned from local BGP peers are redistributed via *SGRP* to remote SCION ASes. Conversely, IP prefixes learned from remote SCION AS via *SGRP* are redistributed to local BGP peers. Accept- and announce-filters can be used to control which IP prefixes are redistributed.

For both use cases, the appliance needs to establish a BGP session with one or more local BGP peers. The BGP configuration is specified in the `bgp` section of the appliance configuration. It is split into the following two subsections:

`global`
: Global configures the configures global attributes that apply to communication with all BGP peers. This includes the following fields:

  - `as`
    The local BGP autonomous system number of the appliance as 32-bit number from RFC 6991.
  - `networks`
    A list of static network prefixes advertised to all BGP peers.

- **router_id**
  Id of the router - an unsigned 32-bit integer expressed in dotted quad notation. Usually, this is an IP address.
- **src_address**
  The preferred source IP address that is used when entering received routes in the Linux routing table.

neighbors
Configures the BGP neighbors. For each neighbor, the following can be specified:

- **auth_password**
  An MD5 authentication password for use with the neighboring device.
- **description**
  An optional user-provided description of the neighbor.
- **enabled**
  Whether the BGP peer is enabled. If this is set to false, the local BGP daemon will not initiate connections to the neighbor, and will not respond to TCP connection attempts from the neighbor. If the BGP session is established at the time that this property is set to false, it will be ceased.
- **local_as**
  The local BGP autonomous system number is used to establish sessions with the peer. If this is not provided the global BGP autonomous system number is used.
- **neighbor_address**
  Address of the BGP peer, either IPv4 or IPv6.
- **peer_as**
  BGP autonomous system number of the peer.
- **timers.connect_retry**
  Time interval in seconds between attempts to establish a session with the peer.
- **timers.hold_time**
  Time interval in seconds that a BGP session will be considered active in the absence of keepalive or other messages from the peer. This is typically set to 3x the keepalive-interval.
- **timers.keepalive_interval**
  Time interval in seconds between transmission of keepalive messages to the neighbor.
- **timers.minimum_advertisement_interval**
  The minimum time in seconds which must elapse between subsequent UPDATE messages relating to a common set of prefixes being transmitted to a peer. This timer is referred to as MinRouteAdvertisementIntervalTimer by RFC 4271 and serves to reduce the number of UPDATE messages transmitted when a particular set of prefixes exhibit instability.
- **transport.local_address**
  Set the local IPv4 address to be used for the BGP session. This may be expressed as either an IP address or the name of an interface.
- **ttl_security**
  BGP Time To Live (TTL) security check. See RFC 5082 for details.
- **ebgp_multihop**
  Allow eBGP sessions to be established with peers that are not directly connected. This setting is mutually exclusive with `ttl_security`.

The BGP daemon configured in the `bgp` section receives and announces IP prefixes from and to peers on the internal network and peers connected via legacy IP networks.

Prefixes can also be exchanged by the IP-in-SCION tunneling endpoints using SGRP. Please refer to the *IP-in-SCION tunneling section* documentation section for information how to configure this. IP prefixes learned and accepted from SCION remotes are entered into the local routing table with a metric of 15 (the Administrative Distance of SGRP). From there they will be picked up by the BGP service and propagated to the BGP peers. Similarly, routes learned via BGP are picked up by the IP-in-SCION tunneling endpoint and propagated to remote ASes while respecting the policies configured in the SCION tunneling section.

BGP Configuration Example

The example shows a BGP configuration with one neighbor.

```
{
   "bgp": {
      "global": {
        "as": 64496,
        "router_id": "10.0.0.1",
        "networks": ["10.31.206.120/32"]
      },
      "neighbors": [
        {
          "enabled": true,
          "local_as": 64510,
          "neighbor_address": "10.0.0.2",
          "peer_as": 64510
        }
      ]
   }
}
```

3.8 System

The `system` section exposes some configuration settings of the operating system and the VPP dataplane that are relevant for the operation of the appliance.

3.8.1 Operating system

The following fields expose operating system settings:

`dns.servers`
: A list of DNS server IP addresses that the appliance uses for domain name resolution.

`ntp.servers`
: A list of NTP time server IP addresses that the appliance uses for time synchronization.

`kernel`
: A group of Linux kernel settings that are exposed.

> **Warning:** Be advised that you should only change these values if you understand the implications. In case you have any questions, reach out to customer-support@anapaya.net.

hugepage_size
The size of hugepages the kernel should allocate at boot time.

hugepages
The number of hugepages the kernel should allocate at boot time.

iommu_enabled
A flag configuring whether the IOMMU subsystem in the Linux kernel is enabled. To improve performance, this should be set to true on all systems that support it.

Hint: Use the following command to find out if the appliance host supports IOMMU:

```
find /sys | grep dmar
```

It should report a non-empty list of devices.

Note: After changing this option the appliance needs to be rebooted.

---

system

The following example configures enables IOMMU and configures custom DNS and NTP servers.

```
"system": {
  "dns": {
    "servers": [
      {
        "address": "2.2.2.2"
      }
    ]
  },
  "kernel": {
    "iommu_enabled": true
  },
  "ntp": {
    "servers": [
      {
        "address": "time3.example.ch"
      }
    ]
  },
}
```

3.8.2 Vector Packet Processor (VPP)

Warning: Be advised that you should only change these values if you understand the implications. In case you have any questions, reach out to customer-support@anapaya.net.

The appliance router and IP-in-SCION tunneling processes use a VPP dataplane to provide fast packet processing.

The following fields configure relevant settings of and related to the VPP dataplane:

buffers.data_size
: The default size of the buffer VPP allocates for a packet, in bytes (VPP Reference).

connection
: Configures the appliance's connection to the VPP dataplane. The VPP control service is the part of the appliance that configures the underlying network devices for the VPP dataplane and regularly performs health checks to make sure it works as expected.

  - **health_check.probe_interval**
    The interval in which a health check probe is sent to the VPP dataplane.
  - **health_check.reply_timeout**
    The time in which the VPP control service expects a reply from the VPP dataplane.
  - **health_check.threshold**
    The number of health checks, from VPP control service to the VPP dataplane, that is allowed to time out before the connection is considered dead.
  - **reconnect_attempts**
    The number of connection attempts from VPP control service to the VPP dataplane after starting up.
  - **reconnect_interval**
    The interval at which a connection is attempted from VPP control service to the VPP dataplane after starting up.

cpu.main_core
: The logical CPU core the main VPP dataplane thread runs on (VPP Reference).

cpu.workers
: The number of workers that are created for the VPP dataplane. The workers are automatically pinned to consecutive CPU cores. (VPP Reference)

VPP creates a TUN device to accept packets from and route packets to Linux. The device gets created automatically and can be configured with the following fields:

tun.mtu
: The MTU (Maximum Transmission Unit) on the TUN device that the VPP dataplane creates in Linux.

tun.prefixes
: A list of prefixes that should not be forwarded by VPP but routed to Linux.

---

system.vpp

The following example configures the VPP dataplane to use a larger than default buffer size and only one CPU.

```
"system": {
  "vpp": {
    "buffers": {
      "data_size": 9000
```

(continues on next page)

(continued from previous page)

```
    },
    "cpu": {
      "main_core": 1,
      "workers": 1
    },
  }
}
```

3.9 Management

The `management` section of the configuration contains the appliance management-related configuration items.

3.9.1 Management API

The appliance management API is configured in the `api` section.

`listeners`
: A list of listeners where each listener specifies a management API endpoint. A listener consists of an address and a description. The address is a combination of an IP address and a fixed port. The address must be specified as `ip:port` for IPv4 and `[ip]:port` for IPv6 addresses. Both IP and port must be set and cannot be left empty. To expose the management API on port 443 on all interfaces, bind it to `0.0.0.0:443`. There must be at least one listener specified.

Note: By default, the management API is configured to listen on `127.0.0.1:443` for initial provisioning.

Note: Appliances prior to version v0.33 only specify a single management API address `address` of the same format as the listener address e.g. `"address": "127.0.0.1:443"`. This field is has been removed and `listeners` must be used.

Connections to the management API are secured using HTTPS. The certificate used is self-signed and generated on the first boot-up of the appliance. You will need to manually trust the certificate to use the management API.

`basic_auth`
: The management API supports basic user authentication using HTTP Basic Auth.

  `enabled`
  : To enable authentication, set the `enabled` property to `true` By default, authentication is disabled.

  `users`
  : The list of users and their hashed passwords that are authorized to access the management API. The `password_hashed` value can be derived using the interactive `appliance-cli` or the `htpasswd` command::

```
appliance-cli crypto kdf hash

htpasswd -nB admin
```

oauth

The management API supports user authentication via OAuth2. It supports the authorization code flow to authenticate users and the client credentials flow to authenticate machines, for example, a periodic job. The management API distinguishes between two roles:

- `appliance/reader` a reader that has read-only access.
- `appliance/writer` a writer that has read-write access.

The roles must be specified in the JSON Web Token (JWT) provided to the API. The roles can be specified either in the `roles` field or in the `https://anapaya.net/roles` field.

enabled

To enable OAuth2 authentication, set the `enabled` property to `true` By default, OAuth2 authentication is disabled.

identity_providers

The list of identity providers. This is needed for the authorization code flow that provides a seamless experience for users of the appliance UI. Each identity provider has the following properties:

id

An identifier to identify the provider. This ID must be unique amongst the other providers.

type

The type of identity provider. The following two types are supported:

- `GENERIC`: A generic identity provider. This should be used if no specific type exist for the identity provider of your choice. In this case the `base_auth_url` and the `metadata_url` field must be set.
- `AZURE_AD`: The Azure active directory (AD) identity provider. When selecting this you likely need to specify the `tenant_id` field.

client_id and client_secret

The credentials to authenticate the appliance at the identity provider. These two fields always need to be set. The values should be retrieved from the configuration in the identity provider.

base_auth_url

The base URL for the identity provider. For example `https://anapaya.eu.auth0.com/` for Auth0. This field only needs to be set for the `GENERIC` provider type.

metadata_url

The URL where the well-known OpenID configuration is hosted. This usually has the following path in the URL: `/.well-known/openid-configuration` .

tenant_id

The tenant ID of the Azure AD. This field only needs to be set if the `type` is `AZURE_AD`, otherwise it is ignored.

token_verification_keys

The token verification keys list specifies the keys that are used to verify the JWTs passed to the API. This is needed for the authorization code flow, where the client provides the JWT directly to the management API.

id

The unique identifier for this key set. This can be a descriptive name, like: `anapaya-auth0`.

jwks_url

The URL where the key set can be fetched. The appliance will periodically get the keys from this URL and update the API Gateway to accept tokens which can be verified with the given key set.

Management API Configuration

This example configures the management API to be available on `192.168.1.1:443` and enables HTTP basic auth for user `admin` and password `password`. The `password_hashed` value is created with the `appliance-cli crypto kdf hash` command.

```
{
   "management": {
      "api": {
         "listeners": [
            {
               "description": "LAN",
               "address": "192.168.1.1:443"
            }
         ],
         "basic_auth": {
            "enabled": true,
            "users": [
               {
                  "username": "admin",
                  "password_hashed": "$2y$05$9j6jB0DORgxVir0QerURDO669uiuMlbZSfz8F0t/Ty/
↪GsjanBZlfW"
               }
            ]
         }
      }
   }
}
```

3.9.2 Telemetry

To enable telemetry of the appliance, the telemetry scraping endpoint needs to be configured in the `telemetry` section.

`address`
: The endpoint on which the telemetry is exposed. The address is a combination of an IP address and a fixed port. The address must be specified as `host:port`, where the host can be empty. An empty address indicates a wildcard address. If the port is 0, it will be automatically allocated.

`labels` List of static labels that are added to all the telemetry data.

`logging`
: Besides telemetry, appliance logs can be centrally collected.

  `logging_type`
  : The type of log export. Currently, only Grafana Loki (`"LOKI"`) is supported.

    `loki`
    : The configuration for the Loki log export.

      `url`
      : The URL of the Loki server. This is where the log records are being pushed to.

      `basic_auth`
      : The basic auth configuration for the Loki log export. To configure it `username` and a `password` need to be provided.

`tls_config`
To disable server certificate verification, set the `insecure_skip_verify` to true.

`flow_metrics`
Configuration for the flow-metrics feature. The gateway collects information about outgoing flows, such as the source and destination ISD-AS and IP address, in order to export the number of IP-in-SCION tunneling users. The flow information is sent to a collector for storage and processing.

`enabled`
To enable the flow-metrics feature, set the `enabled` property to `true`. By default, the feature is disabled.

`collector_url`
The URL of the collector. Supports HTTP, HTTPS, and gRPC transports.

`proxy_url`
URL of the optional HTTP(s) proxy. If set, the flow metric information is sent to the collector via the proxy.

`export_task_interval`
Time interval at which flow metrics are exported to the collector.

`flow_expiration_interval`
Time interval after which inactive flows are considered expired and are marked for cleanup.

`cleanup_task_interval`
Time interval at which expired flows are cleaned up.

Telemetry Configuration

This example configures telemetry to be exported on `192.168.1.1:42001`. Furthermore, appliance logs are exported to a Loki server at `https://loki.example.org:3100/api/v1/push`, using `admin:password` to authenticate to the server and add a label `appliance: s01.chzrh.anapaya` to each log record.

```
{
   "telemetry": {
      "address": "192.168.1.1:42001",
      "labels": [
          {
          "label": "appliance",
          "value": "s01.chzrh.anapaya"
          }
      ]
      "logging": {
         "logging_type": "LOKI",
         "loki": {
            "url": "https://loki.example.org:3100/api/v1/push",
            "basic_auth": {
               "username": "admin",
               "password": "password"
            },
         }
      }
```

3.9.3 Hostname

The appliance hostname can be configured using the `hostname` property. The hostname is used to identify the host in the telemetry data, thus each appliance should have a unique hostname.

hostname
: The hostname of the appliance.

Hostname Configuration

This example configures the hostname to be `s01.chzrh.anapaya`.

```
{
   "management": {
      "hostname": "s01.chzrh.anapaya"
   }
}
```

3.9.4 Remote Software Repository

The appliance can be configured to use a remote software repository to fetch new `anapaya-scion` and `anapaya-system` software releases.

repository_type
: The type of repository. Currently, only `"CLOUDSMITH"` is supported.

cloudsmith
: The configuration for the Cloudsmith repository.

  url
  : The URL of the Cloudsmith repository.

  access_token
  : The access token for the Cloudsmith repository. You receive this as part of your Anapaya software license. Reach out to customer-support@anapaya.net if you need help with the access token.

Remote Repository Configuration

This example configures the appliance to use the Cloudsmith repository at `https://cloudsmith.io/anapaya` and the access token `secret-token`.

```
{
   "management": {
      "repository_type": "CLOUDSMITH",
      "cloudsmith": {
         "url": "https://dl.cloudsmith.io/anapaya",
         "access_token": "secret-token"
      }
   }
}
```

The appliance configuration is divided into the following sections:

1. *SCION*: The configuration of the SCION protocol and AS.

2. *Cluster*: The configuration of the cluster of Anapaya appliances that form a SCION AS.
3. *Cluster*: The configuration of the firewall of the Anapaya appliance.
4. *IP-in-SCION Tunneling*: The configuration of the IP-in-SCION tunneling functionality.
5. *Network Interfaces*: The configuration of the physical and virtual network interfaces of the appliance.
6. *NAT*: The configuration of the NAT.
7. *Border Gateway Protocol*: The configuration of Border Gateway Protocol (BGP).
8. *System*: The configuration of the Linux-based operating system and the VPP dataplane of the appliance.
9. *Appliance Management*: The configuration of the management API, telemetry, and authentication.

3.10 Working with the Configuration

3.10.1 Inspecting the Configuration

To inspect the configuration of an appliance, the GET /api/v1/config API endpoint can be used.

Fetching an appliance configuration

```
curl -k https://${mgmt_address}/api/v1/config
```

3.10.2 Applying a new Configuration

To apply a new configuration, the updated JSON description needs to be pushed to the appliance via the PUT /api/v1/config endpoint. The new configuration first gets validated and then applied. If validation fails, the API responds with an error including a detailed description of the validation error. Once the configuration passed validation, it gets written to the internal configuration store and applied to all the subsystems of the appliance.

Putting a new appliance configuration

```
curl -k https://${mgmt_address}/api/v1/config -X PUT -d @appliance.json
```

CHAPTER

FOUR

MANAGEMENT API SPECIFICATION

CHAPTER

FIVE

SOFTWARE UPDATES

The Anapaya appliance features an installer that can be used to upgrade the software of an Anapaya appliance. The purpose here is to explain how the upgrade process works. For a step by step guide refer to to *Updating an Appliance*.

Note: The assumption here is that the Anapaya appliance with installer (i.e., version *v0.28.0* or higher) is up and running. Refer to *Getting started with the Anapaya Appliance* to install an Anapaya appliance initially.

Anapaya ships the appliance software in two packages: `anapaya-scion` and `anapaya-system`. The packages and their versioning are described in *SCION and System Packages*. *Appliance Management API* gives an overview of the relevant API endpoints for software upgrades. Building on these, the process of upgrading is explained in *Upgrade Process*.

Note: In all commands below *${mgmt_address}* needs to be replaced with the management API address which is configured for the Anapaya appliance. If the commands are run directly via the machine itself, then it can be set to *localhost*.

Similarly, *${version}* needs to be replaced with the version of the `anapaya-scion` or `anapaya-system` package that is to be installed.

5.1 SCION and System Packages

There are two types of packages, `anapaya-scion` packages and `anapaya-system` packages. An `anapaya-scion` package contains Anapaya's own software and tightly integrated third-party software. An `anapaya-system` package contains the system packages present on the appliance. (For example, currently, these are the packages included in the ubuntu-minimal 18.04 image and a set of packages that are installed on top.)

The `anapaya-scion` package is a compressed archive containing the images and binaries that are necessary to run the SCION services. It also contains some files with meta information that can be used for validation of the package or during the installation process.

The `anapaya-system` package is a compressed archive that contains the desired system packages including all their dependencies in a package format that can be fed into a package manager. Currently, this is the APT package manager.

One of the main reasons for the split into the aforementioned package types are that these packages are on different release cycles.

For both types of packages, semantic versioning in the form of <*major*>.<*minor*>.<*patch*> is used. While for the `anapaya-scion` packages, any two arbitrary versions *v1* and *v2* can be installed on top of each other (as far as, *v1* and *v2* are *v0.28* or higher, i.e., have Anapaya installer), this is not entirely true for the `anapaya-system` packages, due to

restrictions imposed by how the system packages are handled. In the following, the semantics behind the versioning schema for `anapaya-system` packages is explained.

The <*major*> version indicates the underlying system release version against which the corresponding system package can be installed.

The <*minor*> version of the system package includes the service pack version. The service packs contain all available updates for every package installed in a major release. The service packs are self-contained with respect to the major release, i.e., it is possible to install service pack 2 and skip service pack 1 for a given major release.

The <*patch*> releases contain individual updates to packages for a given service pack. The patch releases are on-demand and are usually triggered by security updates that become available for critical security vulnerabilities. The patch releases are self-contained with respect to the currently installed service pack, i.e., it is possible to install patch 2 and skip patch 1 for a given service pack and a given major release.

5.2 Appliance Management API

The Anapaya appliance offers an HTTP REST API supporting various operations. In particular, it provides a set of endpoints that can be used to manage and install software packages.

You can refer to *Management API Specification*, for the complete list of software endpoints that are supported. However, let us have a closer look at some of the endpoints, in particular, the ones needed to upgrade to a new version, as will be explained in *Upgrade Process*. Additionally, further information regarding signed releases, keys and signatures can be found at *Signed Releases*.

5.2.1 Authentication

To access the Appliance management API you need to authenticate yourself. The API supports basic authentication. The username and password are configured during the installation of the appliance. When accessing the API using CURL commands you can use the following command line options to authenticate:

```
curl -u <username>:<password> <url>
```

5.2.2 *scion* vs *system* Endpoints

All endpoints can be classified into either *scion* or *system*. As their names suggest, the *scion* and *system* endpoints are utilized to handle `anapaya-scion` and `anapaya-system` packages, respectively. For example you can use the endpoint *GET /software/scion/installed* (resp. *GET /software/system/installed*) to get the version of `anapaya-scion` (resp. `anapaya-system`) package currently installed.

5.2.3 *keys* Endpoint

The `keys` endpoint allows you to view and update the installed public keys that will be used to verify the integrity and authenticity of the `anapaya-scion` and `anapaya-system` packages.

5.2.4 *signatures* Endpoints

The `signatures` endpoints allows you to view and update the installed signatures of the `anapaya-scion` and `anapaya-system` packages that will be used for package verification. For example, the endpoint *GET /software/signatures/scion/${version}* can be used to list the installed signatures for the `anapaya-scion` package of version ${version}.

5.3 Upgrade Process

To upgrade to a new version, you need to download the package for the desired version from Anapaya's software releases repository and then execute the necessary operations via software endpoints of the appliance management API to complete the installation.

5.3.1 Checking the Installed Version

To check what is the version of the `anapaya-scion` package which is currently installed, run:

```
curl -k -u <user>:<password> https://${mgmt_address}/api/v1/software/scion/
↪installed
```

For the installed system version, run:

```
curl -k -u <user>:<password> https://${mgmt_address}/api/v1/software/system/
↪installed
```

5.3.2 Uploading the Desired Package

Now, you need to ensure that the `anapaya-scion`/anapaya-system package for the version that you want to install is locally available. Download it from the Anapaya software releases repository (ask your Anapaya Customer Success Engineer if you do not know how to access the repository). Then you can upload the package to the appliance via the management API.

To upload an `anapaya-scion` package named `anapaya-scion-${version}.tar.gz`, run:

```
curl -k -u <user>:<password> https://${mgmt_address}/api/v1/software/scion/
↪packages/local -X POST --data-binary "@anapaya-scion-${version}.tar.gz"
```

Similarly, to upload an `anapaya-system` package named `anapaya-system-${version}.tar.gz`, run:

```
curl -k -u <user>:<password> https://${mgmt_address}/api/v1/software/system/
↪packages/local -X POST --data-binary "@anapaya-system-${version}.tar.gz"
```

To list the version of all `anapaya-scion` packages which are available locally, run:

```
curl -k -u <user>:<password> https://${mgmt_address}/api/v1/software/scion/
↪packages/local
```

Similarly, for `anapaya-system` packages, run:

```
curl -k -u <user>:<password> https://${mgmt_address}/api/v1/software/system/
↪packages/local
```

Check that the `anapaya-scion` or `anapaya-system` package that you want to install is stored locally.

5.3.3 Upload the Package Signatures

The appliance installer verifies the package integrity and authenticity using the signatures and public keys that are stored in the installer. See *Signed Releases* for more details.

The installer on an Anapaya appliance already includes the official public keys used by Anapaya to sign the software packages, so there is no further action required to fetch the public keys.

The signatures for the packages can be found on the Anapaya Releases website. Find and download the valid signatures for the package version:

```
curl https://releases.anapaya.net/anapaya-scion-${version}.tar.gz.signatures.
↪json > signatures.json
```

Push the latest signatures for this package version to the host using the API:

```
curl -X POST -k -u <user>:<password> https://${mgmt_address}/api/v1/software/
↪signatures/scion/${version} -d @signatures.json
```

5.3.4 Triggering the Installation

To trigger the installation of the `anapaya-scion` package with version `${version}`, run:

```
curl -k -u <user>:<password> https://${mgmt_address}/api/v1/software/scion/
↪install -X POST -d '{"version": "${version}"}'
```

Note: By default, signature verification is enabled. If you want to disable it, you can pass the `skip_signature_verification` parameter to the request.

```
curl -k  -u <user>:<password> https://${mgmt_address}/api/v1/software/scion/
↪install -X POST -d '{"version": "${version}","skip_signature_verification":␣
↪true}'
```

Note that this should never be used in productive setups.

As a part of the response body of the above API call to trigger the installation, you get an installation ID. You can use this installation ID in the below command to check the progress of the installation:

```
curl -k -u <user>:<password> https://${mgmt_address}/api/v1/software/scion/
↪install/{id}
```

Similarly, to trigger the installation of the `anapaya-system` package with version `${version}`, run:

```
curl -k -u <user>:<password> https://${mgmt_address}/api/v1/software/system/
↪install -X POST -d '{"version": "${version}"}'
```

You can again use the returned installation ID in the below command to check the progress of the installation:

```
curl -k -u <user>:<password> https://${mgmt_address}/api/v1/software/system/
↪install/{id}
```

The installation status will be "in_progress", "completed", or "failed". In the case of a "failed" installation of an `anapaya-scion` package, a roll-back to the original version is triggered automatically. For the `anapaya-system` packages, a roll-back is not supported and manual intervention is required. However, this is very unlikely, should it happen please contact Anapaya Customer Support.

5.3.5 Verifying the Installed Version

Check that the desired version has been successfully installed. For `anapaya-scion`, run:

```
curl -k -u <user>:<password> https://${mgmt_address}/api/v1/software/scion/
↪installed
```

And for `anapaya-system`, run:

```
curl -k -u <user>:<password> https://${mgmt_address}/api/v1/software/system/
↪installed
```

5.4 Rolling Back to a Previous Version

In case of a failed installation of an `anapaya-scion` package, a rollback to the original version is triggered automatically and no interaction is required. In case you need to manually roll back, the process is the same as the process to install a new version of the package (see ref:*upgrade_version*). In most cases, the package and signatures should already be available on the appliance from a previous installation. If that is not the case, you need to upload the desired package and its signature to the host, and then trigger the installation using the API.

In general, rollbacks are possible for all versions greater than v0.28.0 of the `anapaya-scion` package. Please refer to the release notes of the newer version to see if any known issues prevent a rollback to the previous version.

For the `anapaya-system` packages, a roll-back is not supported and manual intervention is required. However, this is very unlikely, should it happen please contact Anapaya Customer Support.

5.5 Signed Releases

To verify that the Anapaya software has not been tampered with from the time it is released until the operator installs it on a device, we cryptographically sign our software releases of the `anapaya-scion` and `anapaya-system` packages. This enables our own installer as well as third parties to verify the authenticity and integrity of our software releases against a root of trust.

5.5.1 Signing and verification process

At a high level, a signature is a cryptographically verifiable statement that a certain piece of data - in our case, a software package - (1) has been signed by the owner of the private key corresponding to the public key that is used to verify the signature (authenticity) and (2) that the data has not been modified since it has been signed (integrity).

To sign our releases, we generate an ECDSA-P256 key pair. Using the private key, our signing tool computes the signature over the SHA256 hash of an Anapaya software package, which will serve as the signature of the corresponding package. The private key is stored in a secure, access-controlled location to ensure that it is not compromised.

5.5.2 Retrieving public keys and package signatures

To ensure that the verification process is valid, the verifier (operator) needs to have trust in the public key used for verification. To ensure that operators have access to Anapaya's untampered, valid public keys, we publish them on our website. Specifically, the public keys and the signatures of the Anapaya packages can be found at releases.anapaya.net. In this way, as long as Anapaya manages its own releases, operators can trust that the published keys are valid.

The releases.anapaya.net page contains two sections:

- The `Signing keys` section contains the keys.json file. This is a JSON file that lists all the valid public keys and metadata (e.g. fingerprint and creation time) corresponding to the private keys that Anapaya uses to sign a release version.
- The `Signatures` section is split into two subsections for the `anapaya-scion` and the `anapaya-system`. For each new release of the `anapaya-scion` or `anapaya-system` package, we provide a JSON file that contains the name and checksum of the package, as well as a list of package signatures along with the signature creation time and the public key that can be used to verify the package.

With every release, signatures are added to releases.anapaya.net to reflect the latest state of affairs.

5.5.3 Updating installed public keys

The software installer provided by Anapaya already includes the official public keys used by Anapaya to sign the software packages. Usually, there is no further action required to fetch the public keys. Still, there can be cases where it is required to update the list of public keys on an appliance, for instance in case of a key compromise or periodic key update. To achieve this, you can follow these steps:

1. Fetch the latest public keys from Anapaya's website.

```
curl https://releases.anapaya.net/keys.json > keys.json
```

2. Update the installed keys using the Appliance Installer's API.

```
curl -X POST -k -u <user>:<password> https://${mgmt_address}/api/v1/software/keys -
↪d @keys.json
```

3. (Optional): Verify which public keys are currently installed.

```
curl -k -u <user>:<password> https://${mgmt_address}/api/v1/software/keys
```

5.5.4 Verifying the signature manually

To manually verify that the signature of the software package is valid, you can use Cosign. Cosign is part of the larger sigstore project - a new standard for signing, verifying and protecting software that is owned by the Linux Foundation and has been developed by a collaboration between Redhat and the Google Security team.

After installing cosign following the official instructions, you can verify a signature by running:

```
cosign verify-blob --key <path-to-public-key> --signature <signature> <path-to-package>
```

where:

- `<path-to-public-key>` is the path to the public key that should be used for signature verification. The public key must be in PEM format.
- `<signature>` is the signature of the software package in string format

- <path-to-package> is the path to the software package for which you are doing the verification.

  For example, the command could look similar to the following:

```
cosign verify-blob --key public_keys/public_key1.json --signature␣
↪MEQCIBS8KnIP07fRD2Hi1L0HvvZ2Rc4jkCifx8pjR+/
↪0NxTXAiACzQ+f4piwhVisUdCnJj0bpQgi0CsDlCCJd5MgdPznwA== anapaya-scion-${version}.
↪tar.gz
```

5.5.5 In case of key compromise

While Anapaya goes to great lengths to ensure the private signing keys are secure, we have defined a process for the unlikely case of key compromise. If a private key is compromised, Anapaya will inform all affected parties via email. The compromised key will be removed from our webpage (releases.anapaya.net), and our webpage will explicitly state which key was tainted.

You must then follow this process:

1. Update the verification public keys that are installed in all appliances, following the steps in *Updating installed public keys*.
2. Reinstall the `anapaya-scion` and `anapaya-system` packages following the steps in *Upgrade Process*.

With this process, you can ensure that the installed public keys are up-to-date and that the Anapaya software packages can successfully be verified.

5.6 Custom package installation

Warning: The installation of custom packages might interfere with the Anapaya-managed packages. The following configuration changes might lead to issues with the appliance installer when upgrading to a newer software version and cause the system to break. We highly recommend avoiding custom package installation, unless absolutely necessary.

By default, based on the latest software versions, it is not possible to install packages that are not part of the `anapaya-system` package. However, there might be special cases that require custom package repositories and additional packages to be available. In such cases, you need to follow the next steps:

5.6.1 If the appliance has Internet access

Note: For the following commands, the operator needs to be connected to the appliance as the `root` user or use `sudo`.

1. Copy the following section to the file `/etc/apt/sources.list.d/jammy.list` and save the file.

```
deb http://archive.ubuntu.com/ubuntu jammy main restricted
deb http://archive.ubuntu.com/ubuntu jammy-updates main restricted

deb http://archive.ubuntu.com/ubuntu jammy universe
deb http://archive.ubuntu.com/ubuntu jammy-updates universe

```

(continues on next page)

(continued from previous page)

```
deb http://archive.ubuntu.com/ubuntu jammy multiverse
deb http://archive.ubuntu.com/ubuntu jammy-updates multiverse

deb http://security.ubuntu.com/ubuntu jammy-security main restricted
deb http://security.ubuntu.com/ubuntu jammy-security universe
deb http://security.ubuntu.com/ubuntu jammy-security multiverse
```

2. Update the package lists on your appliance by running `apt update`.
3. Install the package running `apt install <package>`.
4. After installing the custom package, you need to remove the apt sources that were added above. To do this, delete the file `/etc/apt/sources.list.d/jammy.list` that was added in step 2.

5.6.2 If the appliance does not have Internet access

1. Find the desired package in the official list of Ubuntu Jammy packages.
2. Download the `amd64` package version to your local system.
3. Copy it to the appliance device (you can use scp).
4. Install the debian package using `dpkg -i <package>.deb`

CHAPTER

SIX

CERTIFICATE/TRC PROVISIONING

The SCION Control Plane PKI (CPPKI) is built around the concept of federated trust. There is no single entity that defines whom to trust in the SCION network. Each ISD announces their own Trust Root Configuration (TRC) that anchors all of the trust inside that ISD. The TRCs are needed to verify the authenticity of the SCION control plane data.

In order for the appliance to join the SCION network and communicate with other nodes, it has to be configured with a set of trusted TRCs. Furthermore, for the appliance to participate in the SCION control plane, a SCION CPPKI AS certificate and the associated private key are necessary.

We first give a brief introduction to the basic concepts of SCION Control Plane PKI, namely TRC, CA certificate, and AS certificate. Then, we explain how you can provision and inspect TRCs, as well as CPPKI certificates.

6.1 Trust Root Configuration

The SCION TRC is a signed collection of X.509v3 certificates and additional ISD policy information. A TRC acts as a distribution mechanism for the trust anchors of the local ISD. It enables securing the control plane interactions and thus is an integral part of the SCION infrastructure.

A set of trusted TRCs are required for the appliance to join the SCION network and communicate with other nodes. These TRCs build the trust bases for verifying all of the control plane data that is exchanged in the SCION protocol. At the very least, the TRC of the local ISD of the appliance needs to be configured to communicate with the direct neighbors and all the ASes in the ISD.

To communicate with remote SCION ASes in other ISDs, you will need to install the TRC of their respective ISDs and all the ISDs on the path. This ensures that the appliance can verify all the necessary control plane data to reach these remote ASes.

6.2 CPPKI Certificates

Trust within an ISD is rooted in the TRC. The TRC contains one or more Root Certificates, which are used to sign new CA Certificates, which are in turn used to sign one or more AS Certificates.

SCION certificates are x.509v3 certificates. Every certificate has a subject (the entity that owns the certificate) and an issuer (the entity that issued the certificate). Like in Web PKI, in SCION sometimes both the subject and the issuer can be the same entity.

6.3 Installing TRCs

A TRC is created in a TRC ceremony where the voters of the ISD meet in person to sign an agreed-upon TRC. As part of the ceremony, the public keys of all voters are exchanged. More details on this can be found in TRC Signing Ceremony.

The appliance exposes an endpoint to install a TRC. For further information, check out the documentation for this endpoint: Add a TRC file.

Assume that you have a TRC called `ISD1-B1-S1.trc`. To install this TRC, you just need to run the following command:

```
curl -k https://${mgmt_address}/api/v1/cppki/trcs -X POST --data-binary "@ISD1-B1-S1.trc"
```

Hint:

- The base URL for the API on the host is `https://${mgmt_address}/api/v1`.
- The `--data-binary "@<path/to/file>"` flag attaches the TRC as the HTTP request body payload.

Note: `${mgmt_address}` needs to be replaced with the actual management address of the appliance.

The given TRC is first validated before it is added to the trust store. Only valid TRCs are added to the trust store. You can use the `force` query parameter to force the addition of the TRC regardless of its validity.

Warning: Adding a TRC forcefully is a dangerous operation and you should only perform it if you are completely aware of the implications.

When working with cryptographic materials, the **scion-pki** tool comes in handy very often. In particular, it can be used to verify TRC updates or inspect the TRC contents in a human-readable form. For example, you can inspect the contents of the TRC by running:

```
scion-pki trc inspect ISD1-B1-S1.trc
```

6.4 Inspecting TRCs

You can see the list of TRCs which are configured on the appliance by running the command:

```
curl -k https://${mgmt_address}/api/v1/cppki/trcs
```

Alternatively, to inspect the list in a browser, you can navigate to https://\protect\T1\textdollar\{mgmt_address\}/api/v1/cppki/trcs.

For further information, check out the documentation for this endpoint: List the TRC files. There are two related endpoints that are useful when working with TRCs: Get the TRC and List the TRC blob.

6.5 Generating a Certificate Signing Request

For the appliance to participate in the SCION control plane, e.g., to propagate SCION beacons, or transform beacons into proper SCION path segments, a SCION CPPKI AS certificate and the associated private key are necessary. The SCION CPPKI AS certificate is used to verify the authenticity of the control plane material that is produced and signed by this appliance.

A SCION CPPKI AS certificate needs to be requested from one of the CA ASes of the local ISD. The initial certificate is requested with an out-of-band mechanism. Afterward, the SCION CPPKI AS certificates are periodically renewed by the appliance in a fully automatic fashion.

To request the initial certificate, you will need to instruct the appliance to create a Certificate Signing Request (CSR). This CSR then needs to be signed by a local CA AS. The result is a certificate chain consisting of the issued SCION CPPKI AS certificate and the issuing SCION CPPKI CA certificate. This certificate chain then needs to be installed on the appliance.

To learn about generating a new CSR, check out the documentation of the endpoint Create an AS certificate signing request. As you can see, the endpoint documentation explains what entries the request and response body include and what they would look like by providing concrete examples. To make a request, you need to understand and choose a suitable value for each entry.

Assume that you have created a file named `cp-as.json` which matches the request structure of the endpoint. This, in particular, must include the ISD-AS number. The remaining entries are optional in the SCION CPPKI on the technical level. However, their format and presence might be required by a policy that is defined by your local ISD.

To generate a fresh CSR and store it in a file, say `cp-as.csr`, you can run:

```
curl -k https://${mgmt_address}/api/v1/cppki/csrs -X POST --data '@cp-as.json' > cp-as.
↪csr
```

To check the contents of the freshly created CSR, you can use openssl or scion-pki:

```
openssl req -in cp-as.csr -noout -text
scion-pki certificate inspect cp-as.csr
```

6.6 Installing AS Certificates

Assume that you have generated a fresh CSR. Building on that, we need to create a proper SCION CPPKI AS certificate. This is an out-of-band mechanism and is defined by the CA that is responsible for issuing your initial AS certificate. After you have received back the SCION CPPKI AS certificate as part of a full SCION CPPKI certificate chain, you can install it on the appliance.

You can inspect the contents of the certificate chain, which is written in a file called `cp-as.pem`, with openssl or scion-pki by running the command:

```
openssl x509 -in cp-as.pem -noout -text
scion-pki certificate inspect cp-as.pem
```

For further information, check out the documentation of the endpoint Add an AS certificate chain. This explains what the request and response bodies look like.

You can install the AS certificate `cp-as.pem` on the appliance by running:

```
curl -k https://${mgmt_address}/api/v1/cppki/certificates -X POST --data-binary "@cp-as.
↪pem"
```

This first verifies the certificate chain against the active TRC of the local ISD before it is added. Only verifiable certificate chains are added. However, the endpoint provides the option to forcefully use a certificate chain regardless of its validity or verifiability.

Note: If the certificate chain fails to verify, it might be that a TRC is missing which needs to be installed first.

Warning: Adding a certificate chain forcefully is a dangerous operation and you should only perform it if you are completely aware of the implications.

6.7 Inspecting AS Certificates

You can run the following command to list all the installed certificate chains:

```
curl -k https://${mgmt_address}/api/v1/cppki/certificates
```

Alternatively, you can inspect the list of installed certificate chains in a browser by navigating to https://\protect\T1\textdollar\{mgmt_address\}/api/v1/cppki/certificates

For further information, check out the documentation of the endpoint List the certificate chains.

6.8 Manually Renewing AS Certificates

The appliance periodically renews the SCION CPPKI AS certificate. No manual interaction is necessary during normal operation. During the periodic renewal, the distinguished name of the predecessor SCION CPPKI AS certificate is used.

Our appliance management API allows manually triggering a certificate renewal that runs in-band and does not need the manual CSR creation. This is useful, for example, to change the distinguished name or force the use of a new certificate.

For further information, check out the documentation for the endpoint Manually renew an AS certificate chain. As you can see, the request body is very similar to the one from Create an AS certificate signing request. You can actually modify the `cp-as.json` file that we mentioned earlier to reflect the updates that you want to make. Then, by running the following command you can trigger a manual certificate renewal:

```
curl -k https://${mgmt_address}/api/v1/cppki/certificates/renew -X POST --data '@cp-as.
↪json'
```

CHAPTER

SEVEN

TELEMETRY

7.1 Overview

Each Anapaya appliance exposes a telemetry endpoint that can be used to retrieve telemetry data from the appliance.

Tip: To enable telemetry of the appliance, the telemetry endpoint needs to be configured in the *Management* section of the appliance configuration.

The telemetry data is exported in the form of Prometheus metrics. Prometheus is an open-source systems monitoring and alerting tool. It collects and stores metrics as time series data alongside optional key-value pairs called labels. A metric is a numeric measurement of a specific event or condition, e.g., the number of packets sent on a specific interface. Recording metrics in time series provides then higher-level insights such as the rate of change of the sent packet counter to calculate the throughput of the interface. Labels add additional dimensions to a metric, e.g., the name of the interface for which the packet count is collected is added as a label.

Each Anapaya appliance internally has several modules that expose some of their internal states as metrics. Each module manages a particular part of the system, such as the SCION control plane, the SCION data plane, or the IP-in-SCION tunneling service. For each module, we list the exposed metrics, their names, the type of the metric, a brief description, and the attached labels. Please refer to the individual sections below for more information.

To access these metrics, a Prometheus server is needed that ingests the metrics from each appliance. How to set up a Prometheus server to collect appliance metrics is outside the scope of this document. Please refer to the Prometheus Getting Started guide for more information. Should you require assistance with integrating appliance metrics in your monitoring setup, please contact Anapaya's customer support team (customer-support@anapaya.net).

7.2 Control Plane Metrics

control_beaconing_originated_beacons_total

Description	Total number of beacons originated.
Type	counter
Labels	`egress_interface` `result`

control_beaconing_propagated_beacons_total

Description	Total number of beacons propagated.
Type	counter
Labels	`start_isd_as` `ingress_interface` `egress_interface` `result`

control_beaconing_received_beacons_total

Description	Total number of beacons received.
Type	counter
Labels	`ingress_interface` `neighbor_isd_as` `result`

control_beaconing_registered_segments_total

Description	Total number of segments registered.
Type	counter
Labels	`start_isd_as` `ingress_interface` `seg_type` `result`

control_segment_expiration_deficient

Description	Indicates whether the expiration time of the segment is below the configured maximum. This happens when the signer expiration time is lower than the maximum segment expiration time.
Type	gauge
Labels	None

control_segment_lookup_requests_total

Description	Total number of path segments requests received.
Type	counter
Labels	`dst_isd` `seg_type` `result`

control_segment_registry_segments_received_total

Description	Total number of path segments received through registrations.
Type	counter
Labels	`src` `seg_type` `result`

renewal_ca_health_status

Description	Exposes the status of the CA (available, unavailable, starting, stopping), if the host acts as CA and is delegating certificate renewal to the CA service.
Type	gauge
Labels	`status`

renewal_handled_requests_total

Description	Total number of renewal requests served by each handler type (legacy, in-process, delegating).
Type	counter
Labels	`result` `type`

renewal_received_requests_total

Description	Total number of renewal requests served.
Type	counter
Labels	`result`

renewal_registered_handlers

Description	Exposes which handler type (legacy, in-process, delegating) is registered.
Type	gauge
Labels	`type`

trustengine_latest_trc_not_after_time_seconds

Description	The not_after time of the latest TRC for the local ISD in seconds since UNIX epoch.
Type	gauge
Labels	None

trustengine_latest_trc_not_before_time_seconds

Description	The not_before time of the latest TRC for the local ISD in seconds since UNIX epoch.
Type	gauge
Labels	None

trustengine_latest_trc_serial_number

Description	The serial number of the latest TRC for the local ISD.
Type	gauge
Labels	None

7.3 Data Plane Metrics

router_dropped_pkts_total

Description	Total number of packets dropped.
Type	counter
Labels	`interface` `isd_as` `neighbor_isd_as` `type`

router_input_bytes_total

Description	Total number of bytes received
Type	counter
Labels	`interface` `isd_as` `neighbor_isd_as`

router_input_pkts_total

Description	Total number of packets received
Type	counter
Labels	`interface` `isd_as` `neighbor_isd_as`

router_interface_up

Description	1 indicates the interface is up, 0 otherwise.
Type	gauge
Labels	`interface` `isd_as` `neighbor_isd_as`

router_output_bytes_total

Description	Total number of bytes sent.
Type	counter
Labels	`interface` `isd_as` `neighbor_isd_as`

router_output_pkts_total

Description	Total number of packets sent.
Type	counter
Labels	`interface` `isd_as` `neighbor_isd_as`

dataplane_control_dataplane_sync_error

Description	Indicates whether the last dataplane sync had an error (1) or not (0).
Type	gauge
Labels	None

7.4 IP-in-SCION Tunneling Metrics

gateway_flow_exporter_cleanup_run_time

Description	Overall time the flow clean up has been running, in seconds.
Type	gauge
Labels	None

gateway_flow_exporter_export_run_time

Description	Overall time the flow exporting has been running, in seconds.
Type	gauge
Labels	None

gateway_flow_exporter_last_cleanup_time

Description	The timestamp up until which the finished flows were deleted. Seconds since UNIX epoch.
Type	gauge
Labels	None

gateway_flow_exporter_last_export_time

Description	The timestamp of the last time when the flow metrics were exported, successfully. Measured in seconds since UNIX epoch.
Type	gauge
Labels	None

gateway_flow_exporter_lost

Description	The cumulative duration of time (in seconds) for which there has been flow data lost by the flow exporter.
Type	counter
Labels	None

gateway_ippkt_bytes_local_received_total

Description	Total IP packet bytes received from the local network.
Type	counter
Labels	None

gateway_ippkt_bytes_local_sent_total

Description	Total IP packet bytes sent to the local network.
Type	counter
Labels	`isd_as`

gateway_ippkt_bytes_received_total

Description	Total IP packet bytes received from remote gateways.
Type	counter
Labels	`isd_as` `remote_isd_as`

gateway_ippkt_bytes_sent_total

Description	Total IP packet bytes sent to remote gateways.
Type	counter
Labels	`isd_as` `remote_isd_as` `domain` `traffic_class` `path_filter` `remote_address`

gateway_ippkts_discarded_total

Description	Total number of discarded IP packets received from the local network.
Type	counter
Labels	`reason`

gateway_ippkts_local_received_total

Description	Total number of IP packets received from the local network.
Type	counter
Labels	None

gateway_ippkts_local_sent_total

Description	Total number of IP packets sent to the local network.
Type	counter
Labels	`isd_as`

gateway_ippkts_received_total

Description	Total number of IP packets received from remote gateways.
Type	counter
Labels	`isd_as` `remote_isd_as`

gateway_ippkts_sent_total

Description	Total number of IP packets sent to remote gateways.
Type	counter
Labels	`isd_as` `remote_isd_as` `domain` `traffic_class` `path_filter` `remote_address`

gateway_netlink_listener_subscribed

Description	Flag reflecting whether the netlink listener is subscribed route updates.
Type	gauge
Labels	None

gateway_netlink_listener_updates_errors_total

Description	Total number of netlink route updates errors.
Type	counter
Labels	None

gateway_paths_monitored

Description	Total number of paths being monitored by the gateway.
Type	gauge
Labels	`isd_as` `remote_isd_as`

gateway_prefix_fetch_errors_total

Description	Total number of errors fetching prefixes via SGRP.
Type	counter
Labels	`isd_as` `remote_isd_as` `remote_address`

gateway_prefix_fetch_invalid_total

Description	Total number of invalid prefixes received via SGRP.
Type	gauge
Labels	`isd_as` `remote_isd_as` `remote_address`

gateway_prefixes_advertised

Description	Total number of IP prefixes advertised over SGRP.
Type	gauge
Labels	`isd_as` `remote_isd_as` `remote_address`

gateway_prefixes_fetched

Description	Total number of IP prefixes fetched via SGRP.
Type	gauge
Labels	`isd_as` `remote_isd_as` `remote_address`

gateway_remote_discovery_errors_total

Description	Total number of errors discovering remote gateways.
Type	counter
Labels	`isd_as` `remote_isd_as`

gateway_remote_discovery_paths_available

Description	Total number of SCION paths available to the remote gateway discovery.
Type	gauge
Labels	`isd_as` `remote_isd_as` `status`

gateway_remotes

Description	Total number of discovered remote gateways.
Type	gauge
Labels	`isd_as` `remote_isd_as`

gateway_remotes_changes

Description	The number of times the remotes number changed.
Type	counter
Labels	`isd_as` `remote_isd_as`

gateway_session_is_healthy

Description	Flag reflecting session healthiness.
Type	gauge
Labels	`isd_as` `remote_isd_as` `remote_address` `path_filter` `domain`

gateway_session_latest_path_expiration

Description	Latest path expiration per session monitor.
Type	gauge
Labels	`isd_as` `remote_isd_as` `remote_address` `path_filter` `domain`

gateway_session_path_changes

Description	Number of path changes per session monitor.
Type	counter
Labels	`isd_as` `remote_isd_as` `remote_address` `path_filter` `domain`

gateway_session_paths_available

Description	Total number of paths available per session.
Type	gauge
Labels	`isd_as` `remote_isd_as` `remote_address` `path_filter` `domain` `status`

gateway_session_state_changes

Description	Number of state changes per session monitor.
Type	counter
Labels	`isd_as` `remote_isd_as` `remote_address` `path_filter` `domain`

gateway_sgrp_paths_available

Description	Total number of paths available for SGRP per remote gateway.
Type	gauge
Labels	`remote_isd_as` `remote_address` `status`

gateway_state_reconfiguration_duration_seconds_total

Description	Overall Duration of all the state reconfigurations. Component label can be either ‘planning’, ‘dataplane’ or ‘controlplane’.
Type	counter
Labels	`component`

7.5 Appliance Cluster Metrics

appliance_controller_enforcer_license_expiry

Description	Time when the current license expires or when the current trial/grace period ends.
Type	gauge
Labels	None

nodesync_topology_fetch_errors_total

Description	The number of errors when fetching topology information from a remote node.
Type	counter
Labels	`remote`

nodesync_topology_merge_interface_conflicts_total

Description	The number of topology merge conflicts. This indicates a severe misconfiguration of appliances. It means that multiple appliances have the same interfaces configured.
Type	counter
Labels	`isd_as` `interface`

nodesync_topology_merge_service_conflicts_total

Description	The number of topology merge conflicts. This indicates a severe misconfiguration of appliances. It means that multiple appliances have services configured with the same configuration.
Type	counter
Labels	`service` `isd_as` `shard`

7.6 Installer Metrics

appliance_installer_checksum_consistent

Description	Whether the checksum of the installed package does match the checksum in the package signature file. This may fail if a different package with the same version number was uploaded but it hasn't been installed.
Type	gauge
Labels	`pkgtype`

appliance_installer_controller_watchdog_errors_total

Description	Total number of errors encountered by the appliance controller watchdog. If this counter increases, the installer logs should be inspected for more details.
Type	counter
Labels	None

appliance_installer_installed_package_versions

Description	The version of the installed scion and system package.
Type	gauge
Labels	`pkgtype` `version`

appliance_installer_metastore_inconsistent

Description	Whether the appliance installer's metastore is in an inconsistent state. Value is 1 if the metastore is in an inconsistent state, 0 otherwise.
Type	gauge
Labels	None

appliance_installer_rollback_installations_total

Description	Total number of rollback installations. Result label is the result of the installation.
Type	counter
Labels	`result`

appliance_installer_scion_installations_total

Description	Total number of scion package installations. Result label is the result of the installation.
Type	counter
Labels	`result`

appliance_installer_system_installations_total

Description	Total number of system package installations. Result label is the result of the installation.
Type	counter
Labels	`result`

7.7 BGP Metrics

BGP metrics are metrics from the BGP daemon (FRR).

frr_bgp_peer_groups_count_total

Description	Number of peer groups configured.
Type	gauge
Labels	`vrf` `afi` `safi` `local_as`

frr_bgp_peer_groups_memory_bytes

Description	Memory consumed by peer groups.
Type	gauge
Labels	`vrf` `afi` `safi` `local_as`

frr_bgp_peer_message_received_total

Description	Number of received messages.
Type	counter
Labels	`vrf` `afi` `safi` `local_as` `peer` `peer_as`

frr_bgp_peer_message_sent_total

Description	Number of sent messages.
Type	counter
Labels	`vrf` `afi` `safi` `local_as` `peer` `peer_as`

frr_bgp_peer_prefixes_advertised_count_total

Description	Number of prefixes advertised.
Type	gauge
Labels	`vrf` `afi` `safi` `local_as` `peer` `peer_as`

frr_bgp_peer_prefixes_received_count_total

Description	Number of prefixes received.
Type	gauge
Labels	`vrf` `afi` `safi` `local_as` `peer` `peer_as`

frr_bgp_peer_state

Description	State of the peer (2 = Administratively Down, 1 = Established, 0 = Down).
Type	gauge
Labels	`vrf` `afi` `safi` `local_as` `peer` `peer_as`

frr_bgp_peer_types_up

Description	Total Number of Peer Types that are Up.
Type	gauge
Labels	`type` `afi` `safi`

frr_bgp_peer_uptime_seconds

Description	How long has the peer been up.
Type	gauge
Labels	`vrf` `afi` `safi` `local_as` `peer` `peer_as`

frr_bgp_peers_count_total

Description	Number peers configured.
Type	gauge
Labels	`vrf` `afi` `safi` `local_as`

frr_bgp_peers_memory_bytes

Description	Memory consumed by peers.
Type	gauge
Labels	`vrf` `afi` `safi` `local_as`

frr_bgp_rib_count_total

Description	Number of routes in the RIB.
Type	gauge
Labels	`vrf` `afi` `safi` `local_as`

frr_bgp_rib_memory_bytes

Description	Memory consumbed by the RIB.
Type	gauge
Labels	`vrf` `afi` `safi` `local_as`

7.8 Host Metrics

Host metrics are metrics from the host itself, such as CPU usage, memory consumption or network traffic on the physical network ports.

node_cpu_seconds_total

Description	Seconds the CPU spends in each mode.
Type	counter
Labels	`cpu` `mode`

node_load1

Description	1 minute load average.
Type	gauge
Labels	None

node_load5

Description	5 minute load average.
Type	gauge
Labels	None

node_load15

Description	15 minute load average.
Type	gauge
Labels	None

node_memory_MemTotal_bytes

Description	Total amount of memory in the node.
Type	gauge
Labels	None

node_memory_MemAvailable_bytes

Description	Amount of available memory in the node.
Type	gauge
Labels	None

node_filesystem_size_bytes

Description	Filesystem size in bytes.
Type	gauge
Labels	`device` `fstype` `mountpoint`

node_filesystem_avail_bytes

Description	Filesystem available bytes.
Type	gauge
Labels	`device` `fstype` `mountpoint`

node_network_receive_bytes_total

Description	Number of bytes received from the network.
Type	counter
Labels	`device`

node_network_transmit_bytes_total

Description	Number of bytes transmitted to the network.
Type	counter
Labels	`device`

CHAPTER

EIGHT

TROUBLESHOOTING & RUNBOOKS

This documentation page contains runbooks for the alerts sent out by the Anapaya products as well as common operations that are helpful when troubleshooting.

8.1 Alerts

The names of the alerts follow a uniform structure to provide a quick insight into their meaning:

- **Namespace**: This is the overall part that is having a problem.
- **Subsystem**: Within the namespace what component is affected.
- **Subject**: The subject that is having an issue.
- **State**: The state of the subject, explaining in what way it is not conforming to expectations.

8.1.1 BGPDaemonDisconnected

In general, the gateway is connected to a BGP daemon (FRR) and uses it to publish the routes received via SGRP to the local network. This alert fires when the connection between the gateway and the BGP daemon breaks. Typically, this is caused by the BGP daemon being dead. Given that the appliance automatically restarts the crashed daemon, it is likely that the alert was triggered by the daemon crashlooping on startup.

Alternatively, the daemon may be stuck and thus the connection cannot be established.

When the alert is firing, the routes received via SGRP cannot be published to the local network. This means that the remote IP prefixes are not reachable any more through IP-in-SCION tunneling.

Actions

1. *Check at the logs* of the `frr` service to understand what the state of the daemon is.
2. *Restart the service* if needed.

For more detailed debugging, you can *connect to FRR's interactive console* and refer to the official FRR documentation for further information.

8.1.2 BGPDaemonUnresponsive

In general, the gateway is connected to a BGP daemon (FRR) and uses it to publish the routes received via SGRP to the local network. This alert fires when the connection between the gateway and the BGP daemon exists, but the gateway cannot publish paths to the daemon. Most likely, the BGP daemon is stuck.

When the alert is firing, the routes received via SGRP cannot be published to the local network. This means that the remote IP prefixes are not reachable anymore through IP-in-SCION tunneling.

Actions

1. *Look at the logs* of the `frr` service to understand what the state of the daemon is.
2. *Restart the service* if needed.

For more detailed debugging, you can *connect to FRR's interactive console* and refer to the official FRR documentation.

8.1.3 BGPDaemonDown

This alert fires when the BGP daemon (FRR) on the appliance is down.

Actions

1. *Look at the logs* of the `frr` service to understand why the daemon is not running.
2. *Restart the service* if needed.

For more detailed debugging, you can *connect to FRR's interactive console* and refer to the official FRR documentation.

8.1.4 BGPPeerDown

This alert fires when a BGP peer of the appliance's BGP daemon is down. If the appliance has multiple BGP peers configured, this might not be critical.

Actions

1. Check if the peer IP is reachable from the appliance using the *IP ping command*.

For more detailed debugging, you can *connect to FRR's interactive console* and refer to the official FRR documentation.

8.1.5 ClusterTopologySyncFetchError

This alert fires if there is an error when fetching topology information from a remote node.

Actions

1. Check the network connectivity between the two appliances using the *IP ping command*.
    1. *Check the logs* of the appliance-controller to find information on the other appliance and its address.
    2. If the appliance-controller logs do not have this information, check the cluster section of the appliance configuration for all the necessary information. Please refer to *Cluster* for further information on the cluster configuration fields.

    ```
    appliance-cli get config
    ```

2. In case the connectivity between the two appliances does not work, then there might be an underlying network problem. Please check and troubleshoot the internal network setup to resolve the issue.
3. If the connectivity works, *check the logs* of the appliance-controller to further investigate this issue.

8.1.6 ClusterTopologySyncInterfaceMergeConflict

This alert fires if there are conflicts in the topology merge for the specified interface. This indicates a severe misconfiguration of appliances and means that multiple appliances have the same SCION interfaces configured.

Actions

1. Follow the instructions on how to *fix a topology synchronization misconfiguration*.

8.1.7 ClusterTopologySyncServiceMergeConflict

This alert fires if there is a conflict in the topology merge. It indicates that multiple appliances have the same address for the SCION control service configured.

Actions

1. Follow the instructions on how to *fix a topology synchronization misconfiguration*.

8.1.8 DataplaneControlSyncFailing

This alert fires if the dataplane control process could not apply the desired configuration. This should only happen after a new configuration has been pushed.

Actions:

1. *Check the logs* of the `dataplane-control` process. Identify the problematic configuration and correct it.

8.1.9 ManagementSoftwareChecksumInconsistent

The appliance-controller periodically checks whether the installed software version is consistent with the version on the disk. This alert fires if the checksum of the installed software package does not match the checksum in the signatures files.

Actions

1. Check the checksum of the installed package version: `appliance-cli info`
2. Check if the signatures file for the same version exists (`anapaya-scion-{version}.tar.gz.signatures.json`):

```
ls -l /var/anapaya/installer/packages
```

3. In case the file exists, compare the `sha256sum` inside the file to the one from the first step.
4. If the file does not exist, *upload the package signatures*.
5. Compute the checksum of the package to check if it matches the one from above:

```
sha256sum anapaya-scion-{version}.tar.gz
```

6. If the checksum does not match, *reinstall the package*.

8.1.10 SCIONBeaconOriginationError

The beacon origination fails for a given SCION interface. The alert fires after a prolonged period of beacon origination errors. If beacons can't be originated, connectivity to other ASes via the affected interface will eventually break.

Actions

1. *Check the logs* of the control service for the given ISD-AS. Look for the line `Unable to originate on interface`. It should contain more details about the problem.
2. In case beacon creation failed:
    1. Make sure that the AS certificate is valid, through the /cppki/certificates API endpoint.
    2. Check that *time synchronization* is working properly.
3. In case the creation of the beacon sender or the sending of the beacon failed, make sure that the connection to the neighboring AS is working properly by following the *basic troubleshooting guide*.

8.1.11 SCIONApplicationRegistrationError

SCION applications are having errors while registering with the dispatcher. If this issue persists, SCION control traffic will no longer be sent or received.

Actions

1. *Check the logs* of the dispatcher for the given appliance.
2. *Restart the dispatcher*.

8.1.12 SCIONBeaconPropagationError

The beacon propagation fails for a given SCION interface. The alert fires after a prolonged period of beacon propagation errors. Beacon propagation is needed to keep connectivity to downstream and sibling core ASes. This means that if beacon propagation does not work for an extended period of time, connectivity will eventually break.

Actions

1. *Check the logs* of the control service for the given ISD-AS. Look for the lines `Unable to propagate beacons` and `Error propagating beacons on interface`. These logs should contain more details about the problem.
2. In case the problem is related to beacon extension:
    1. Make sure that the AS certificate is valid, through the /cppki/certificates API endpoint.
    2. Check that *time synchronization* is working properly.
3. In case the creation of the beacon sender or the sending of the beacon failed, make sure that the connection to the neighboring AS is working properly by following the *basic troubleshooting guide*.

8.1.13 SCIONBeaconPropagationInternalError

This alert fires if there is an internal error when attempting to do beacon propagation. Beacon propagation is needed to keep connectivity to downstream and sibling core ASes. This means that if beacon propagation does not work for an extended period of time, connectivity will eventually break.

Since the error is internal, this indicates that there is an issue with the local beacon database.

Actions

1. *Check the logs* of the control service for the given ISD-AS. Look for the `Unable to propagate beacons` line. This log should contain more details about what exactly is the error.
2. The beacon database is in a docker volume. To check the underlying disk,
    1. *Find the name* of the control service container.
    2. Check the underlying disk using the following command (replace the `<control-name>` with the actual control service name):

    ```
    findmnt $(docker inspect <control-name> -f '{{ .GraphDriver.Data.MergedDir}}')
    ```

    In a normally working system, this should list a disk with options `rw` which means the disk is read-writeable. If that isn't the case, the underlying system disk may have an issue.
3. Try to *restart the control service*.

8.1.14 SCIONBeaconReceiveError

There is an increased number of errors regarding the process that receives beacons from neighboring ASes. This can be either because an AS is sending bogus beacons that cannot be verified or because the beacons can not be stored locally.

Actions

1. Check the result label of the metric involved in the alert.
2. If there are a lot of `err_verify` counts, it means that a neighbor is sending bogus beacons that can't be validated or verified:
    1. *Check the logs* of the control service for the given ISD-AS.
    2. Check that *time synchronization* is working properly.
    3. Contact the neighboring AS to inquire why it is sending non-verifiable beacons.
3. If there are a lot of `err_db` or other error labels:
    1. *Enable debug logs* on the control service for the given ISD-AS.
    2. *Check the logs* of the control service for the given ISD-AS for the detailed errors. The errors can either be related to an issue with the disk, or the neighboring AS sending bogus beacons.
    3. If the issue is related to the latter, contact the neighboring AS to inquire why it is sending non-verifiable beacons.

8.1.15 SCIONCAUnavailable

This alert fires if the given ISD-AS is unable to connect to the CA backend.

Actions

For the SCION CA backend provided by Anapaya:

1. Check if the *ca-frontend service is running*.
2. Get the ca-frontend management API address from the start of the log and check if the API is reachable using:

```
curl <management_addr>/api/v1/config
```

3. If the API is reachable, check the connection to the CA backend. Depending on the CA backend used (*Anapaya SCION CA* or the respective third-party CA), try to investigate the connectivity issue using the respective documentation.
4. If you are using an external CA backend, verify the connectivity with the external service and check the configured credentials.

8.1.16 SCIONCryptoASCertificateExpiresSoon

If the AS certificate for the given ISD-AS expires in less than 48 hours, this alert is triggered. This indicates that the automatic renewal process failed to renew the AS certificate.

Renewal happens when the certificate has passed 3/4 of its lifetime. Therefore, if this alert triggers, the automatic renewal process failed to renew the AS certificate for at least six hours.

Actions

1. Check the *Certificate/TRC Provisioning* section to manually renew your AS certificate.
2. If the above step also fails, the error message gives a more detailed insight into the cause of the error.

8.1.17 SCIONCryptoTRCDiskWriteError

This alert fires when writing a TRC to the disk fails.

Actions

1. *Check the logs* of the control service.
2. Look for the logs `Failed to write TRC to disk...` or `Failed to stat TRC file on disk...` indicating that writing the TRC to the disk failed.
3. Check the appended error message to get the root cause of this issue.
4. Check the *disk space* and if the disk is configured as read-only. If that is the case, try to mitigate the cause to avoid any further problems.

8.1.18 SCIONCryptoTRCExpiresSoon

This alert fires if the TRC for the given ISD-AS expires in less than 30 days.

Actions

1. If you are a non-voting party in the ISD, contact the responsible party of the TRC to check if there is an underlying issue.
2. If you are a voting party, follow the actions below.
    1. *Inspect the TRC* to verify that it expires soon.
    2. Initiate the TRC update process with a quorum of voting members. The details of this process depend on the governance rules of your respective ISD.

8.1.19 SCIONInterfaceStateDown

This alert fires when the SCION interface to the neighboring AS is down.

Actions

1. Check if the *administative_state* for the given interface is set as expected in the appliance config:

```
appliance-cli get config -f 'body.config.scion.ases[isd_as == "<ISD-AS>"].
↪neighbors[neighbor_isd_as == "<neighbor ISD-AS>"].interfaces[interface_id ==
↪<interface_id>]'
```

2. Check the actual state of the SCION interface:

```
appliance-cli get debug/scion/interfaces -f 'body.interfaces[local.interface_id ==
↪<interface_id>]'
```

3. If the states do not match, make sure your *network interface configuration* is correct.

```
appliance-cli get config
```

4. Determine whether the underlying IP connectivity is the issue. Get the local and remote interface addresses from the appliance configuration and use the *IP ping command*: `ping <remote> -I <local>`.
5. Check the `Anapaya/router` dashboard to see if BFD packets are being sent and received (`BFD` graphs).

8.1.20 SCIONInterfaceStateFlapping

The Bidirectional Forwarding Detection (BFD) protocol is used to continuously check the health of the links. If the BFD session state on a given interface changes too frequently, this alert is triggered.

Actions

1. Determine whether the underlying connection is the issue. Use the *IP ping command* with the option `-i 0.01` to see if there are any anomalies when the BFD flap happens. You can get the destination address of the relevant neighbor mentioned in the alert from the appliance configuration.

```
appliance-cli get config
```

2. Consider relaxing the BFD timers by updating the `detection_multiplier` in the *Bidirectional Forwarding Detection* section of the appliance configuration.

8.1.21 SCIONNeighborPathsMissing

This alert fires when there is no SCION connectivity from this appliance to one of its direct neighbors.

Actions

1. Follow the *basic troubleshooting guide* to investigate why there is a SCION connectivity issue.

8.1.22 GatewayFlowsNotExported

Flow metrics are not being exported. This means that the reports about gateway usage will be missing data, which, in turn, could lead to the loss of business revenue.

The data is kept for some time (30 minutes by default) so that the operator has the opportunity to resolve the issue without losing any data.

Actions

1. Check *gateway logs* and check for some `exporting flow metrics` error.
2. Depending on the logs, try to resolve the issue.

8.1.23 GatewayNetlinkListenerNotSubscribed

The gateway is not subscribed to netlink route updates and can thus not learn and redistribute routes received from BGP peers. This indicates that something with the underlying Linux operating system is not working as expected.

Actions

1. *Restart the gateway* and check if the problem gets resolved.
2. If the problem persists, reboot the appliance:

```
.. code-block:: bash

   sudo reboot
```

8.1.24 GatewayNetlinkListenerErrors

The gateway is subscribed to netlink route updates, but the netlink listener is missing route updates and might thus not correctly redistribute all route updates to its remotes. This indicates that something with the underlying Linux operating system is not working as expected.

Actions

1. *Restart the gateway* and check if the problem gets resolved.
2. If the problem persists, reboot the appliance:

```
.. code-block:: bash

   sudo reboot
```

8.1.25 SCIONSegmentRegistrationError

Received beacons are converted to segments and registered in the path segment database. Up and core segments are registered in the local database. Down segments are registered at the originating core AS. If this alert fires, then segment registration fails. First, check the segment type from the alert description and depending on it follow the actions below.

Actions for down segments

1. *Check the logs* of the control service for the given ISD-AS. Look for the line `Unable to register segment`. It contains more details about the problem.
2. Make sure that the connection to the originating core AS is working properly by following the *basic troubleshooting guide*.

Actions for up/core segments

1. This alert should not trigger for those segment types. Contact Anapaya Support.

8.1.26 SCIONSegmentRegistrationInternalError

This alert fires if there is an internal error when attempting to register a segment. Segment registration is needed to keep connectivity to other ASes. That means if segment registration does not work for an extended period of time, connectivity will eventually break.

Since the error is internal, this indicates that there is an issue with the local beacon or segment database.

Actions

1. *Check the logs* of the control service for the error to get more insight into what the error could be.
2. The beacon database is in a docker volume. To check the underlying disk:
    1. *Find the name* of the control service container.
    2. Check the underlying disk using the following command (replace the `<control-name>` with the actual control service name):

    ```
    findmnt $(docker inspect <control-name> -f '{{ .GraphDriver.Data.MergedDir }
    ↪}')
    ```

    In a normally working system, this should list a disk with options `rw` which means the disk is read-writeable. If that isn't the case, the underlying system disk may have an issue.
3. Try to *restart the control service*.

8.1.27 SCIONSyncFailing

This alert fires if there has been no successful synchronization of the shard database in the last hour. The synchronization is part of the SCION control service for the given ISD-AS.

Actions

1. *Look at the logs* of the control service. You will find the reason for the failed synchronization in the logs. The process that needs to be followed to resolve the issue depends on the failure case.

8.1.28 SCIONSyncFetchFailing

This alert fires if the SCION control service fails to receive beacons or path segments from some of its peers.

Actions

1. *Check the logs* of the control service and look for the entry `Fetch from network failed...`, which will help to determine the underlying issue.
2. If the issue is network related, make sure the network connectivity is working as expected using the *IP ping command* for the destination mentioned in the logs.
3. In case of a parsing error, investigate why the other party is sending malformed objects.

8.1.29 SCIONSyncStoreFailing

This alert fires if the SCION control service fails to store objects which is equivalent to a database write error.

Actions

1. *Check the logs* of the control service and look for the entry `Write to database failed...`, which will help to determine the underlying issue. The error could be caused due to an issue with the disk.
2. If the error is related to the disk, check the underlying disk using the following command (replace the `<control-name>` with the actual control service name):

   ```
   findmnt $(docker inspect <control-name> -f '{{ .GraphDriver.Data.MergedDir}}')
   ```

   In a normally working system, this should list a disk with options `rw` which means the disk is read-writeable. If that isn't the case, the underlying system disk may have an issue.

8.1.30 ServiceApplianceControllerConfigInvalid

This alert triggers if the appliance successfully validated the configuration, but failed when applying it. This indicates a missing validation in the appliance and therefore that the appliance could not be correctly configured. The reason is not a misconfiguration, but the lack of validation in this case.

Contact Anapaya

In this case, please contact Anapaya Customer Service and provide the following information:

- *General information*
- *Logs* of the appliance-controller

8.1.31 ServiceApplianceControllerPanic

This alert fires when something in the appliance-controller went unexpectedly wrong and resulted in a panic.

The appliance-controller is the centerpiece of the Anapaya appliance. Based on the appliance configuration, the appliance-controller automatically configures all the required SCION infrastructure services.

Actions

1. Make sure the system is up and running. *Check if the appliance-controller is running*. The appliance-installer should have restarted it in case of a failure.
2. If the appliance-controller is not running, *check the logs* of the appliance-controller to get an insight in why it is crashing.
3. Check the release notes, for any known issues with the installed release. If there are known issues, consider *updating the host to the latest version* to resolve the issue.

Contact Anapaya

Please contact Anapaya Customer Support and provide the following information:

- installed scion package version: `appliance-cli version`
- *Logs* of the appliance-controller
- *Logs* of the appliance-installer

8.1.32 SystemClockSyncFailing

The system is not able to synchronize its clock. For the clock synchronization, the system service (`systemd-timesyncd`) is used as an NTP client.

Actions

1. Follow the *timesyncd checks* to check if the corresponding system service is running correctly.

8.1.33 SystemDiskSpaceBootLow

This alert fires when the free disk space for the boot partition (`/boot`) is less than 100MiB.

Actions

1. *Analyze disk space usage* for the `/boot` partition to find what takes up the most memory.
2. Free up space using the *APT related commands*.
3. If the problem persists, consider the possibility that the machine is underprovisioned to meet the resource requirements. Contact Anapaya Customer Support with the *hardware configuration*.

8.1.34 SystemDiskSpaceRootLow

This alert fires when the free disk space for the root partition (`/`) is less than 10%.

Actions

1. *Analyze disk space usage* for the root (`/`) partition to find what takes up the most memory.
2. Free up space using the *APT related commands*.
3. *Remove unused docker images*.
4. Increase size of root partition.
5. If the problem persists, consider the possibility that the machine is underprovisioned to meet the resource requirements. Contact Anapaya Customer Support with the *hardware configuration*.

8.1.35 SystemNTPSyncServiceNotRunning

This alert fires if the `systemd-timesyncd` service is not running. The service is used to syncronize the local system clock with a remote Network Time Protovol (NTP) server.

Actions

1. Follow the *timesyncd checks* to check if the corresponding system service is running correctly and restart it if required.

8.1.36 SystemResourcesAverageLoadHigh

This alert fires if the system is heavily loaded on average in the last 15 minutes. Make sure the system is running properly and try to determine if some processes are using more CPU than usual.

Configuration

1. Check the number of CPU cores:

```
nproc
```

2. If the number is two, make sure you have configured the appliance to not use any workers for the *dataplane*.

Actions

1. Identify the processes causing the high CPU load:

```
ps -eo pcpu,pid,user,args --sort -%cpu | head -10
```

2. Assess if the process should be using the amount of resources. If this is not the case, one option is to restart the process. Note, that this might cause a service interruption.
3. If the problem persists, consider the possibility that the machine is underprovisioned to meet the resource requirements. Contact Anapaya Customer Support with the *hardware configuration*.

8.1.37 SystemResourcesFreeMemoryLow

This alert fires if the system runs low on memory. Make sure the system is running properly and try to free up memory by cleaning caches and buffers.

Actions

1. Check the summary of RAM usage:

```
free -h
```

2. Identify the processes that require the most memory:

```
ps -eo pmem,pid,user,args --sort -%mem | head -10
```

3. Assess if the process should be using the amount of resources. If this is not the case, one option is to restart the process. Note, that this might cause a service interruption.
4. If the problem persists, consider the possibility that the machine is underprovisioned to meet the resource requirements. Contact Anapaya Customer Support with the *hardware configuration*.

8.1.38 SystemServiceDown

This alert fires if the `job` specified in the alert cannot be reached by the monitoring system.

The `appliance` job works as a proxy for all other jobs, so when the alert concerns the `appliance` job it may be that there is overall no network connectivity from the monitoring system to the given host.

Otherwise, this alert means that one of the services on the Anapaya appliance is not working. Services are docker containers managed by the appliance itself. If the service fails, it will be automatically restarted. However, this alert indicates that the service is not available for a prolonged amount of time. Common cases where this alert is fired is (1) when the service is in a crashloop trying to start up and failing repeatedly and (2) when the appliance notifications are disabled and the appliance-controller does not restart the containers.

Actions

1. If the problem is with the connectivity to the appliance you may try to *ping* it from the monitoring host. You can find the address to ping in the `instance` label of the alert.
2. If the address is reachable, you can check *scrape_duration_seconds* metric value, which is helpful to determine whether the scraping failed entirely or whether it is flapping. In the latter case, it may be a latency or a bandwidth problem.
3. If the connectivity is fine, *check the logs* of the failing service. If the service is repeatedly crashing you should be able to find the reason in the logs.
4. If there are no logs from the service it may also be that the appliance-controller is not able to start the service at all. *Check the logs* of the appliance-controller to find out why this is the case.

8.1.39 SystemServiceRestarted

This alert fires if the `job` specified in the alert has *restarted recently*. This may be a legitimate restart (e.g. because a new version of the service was installed or a configuration change required a restart) or it may be that the service crashed.

Note: If the service is functional after the restart, the alert will stop firing in few minutes.

Actions

1. *Check the logs* of the service. You should be able to find the reason why the service exited.

8.1.40 SystemServiceDebugLogsEnabled

This alert fires if the `job` specified in the alert is running in debug log mode for 24 hours. The debug log level should only be used for troubleshooting.

Actions

1. *Revert the log level* to the default info state for the given service.

8.1.41 SystemVPPMemoryUsageHigh

VPP preallocates memory on start up. This alert is triggered when 90% of the preallocated memory is used.

Actions

1. Look at the historic VPP memory usage.
    1. It could have run organically with the increased usage or it may have grown when a new feature was enabled.
    2. If the memory usage slowly grows each time after the dataplane restart, it is probably a bug and Anapaya support should be contacted.
2. To get more information about the VPP memory usage, run:

```
appliance-cli get debug/vpp/memory
```

8.1.42 SystemVPPPacketReceiveErrors

This alert fires if a VPP interface is experiencing errors when receiving packets. This could be due to a faulty NIC, faulty cables, faulty SFP, wrong CRC, L1 configuration mismatch, etc.

Actions

1. *Check the logs* of the dataplane for error entries that highlight the issue.
2. Check the state of the interface in question using

```
appliance-cli get debug/vpp/hardware
```

3. Check the networking hardware itself for issues.

8.1.43 SystemVPPPacketTransmitErrors

This alert fires if a VPP interface is experiencing errors when transmitting packets due to NIC and/or carrier errors such as faulty cable, L1 configuration mismatch, etc.

Actions

1. *Check the logs* of the dataplane for error entries that highlight the issue.
2. Check the state of the interface in question using

```
appliance-cli get debug/vpp/hardware
```

3. Check the networking hardware itself for issues.

8.1.44 SystemVPPReceiveBuffersLow

This alert fires if VPP is running out of buffers. This is probably related to a bug in the software.

Actions

1. Escalate to Anapaya support.
2. To get more information about the VPP buffers, run:

```
appliance-cli get debug/vpp/buffers
```

8.1.45 SystemVPPReceiveQueueFull

This alert fires if a VPP interface is missing packets because the queue is full. This could be caused by VPP worker threads not being able to process the received traffic.

Actions

1. Check the VPP runtime data:

```
appliance-cli get debug/vpp/runtime
```

    1. Look at `Vectors/Call` column. This is the average number of packets processed in one go.The maximum number of packets in one batch is 256. Therefore, if the CPU is not keeping up with the traffic, the numbers will be close to 256.
    2. If this is the case, either configure VPP to use more cores, or, if not possible, consider splitting the traffic between multiple machines.

8.1.46 TunnelingReceivedInvalidPrefixes

This alert fires if a remote AS announced an invalid of non-canonical prefix. The prefix is ignored and the gateway otherwise works normally.

Actions

1. Inform the remote AS that they are trying to announce an invalid prefix.

8.1.47 TunnelingDomainHealthyPathsMissing

This alert fires if there are either no paths for a specific domain or if the paths that are available are not alive (probes don't pass through). Given the above, the packets that match the specified domain cannot be delivered to the destination.

Actions

1. Find the available paths by running:

```
appliance-cli inspect scion-tunneling summary --all-paths \
  --domain <domain>
```

2. Inspect the individual paths. Some of them may be `dead` (no probes are passing through), or `expired`.
    1. If the paths are dead, you should investigate the network connectivity.
    2. If they are expired, you should investigate why the paths are not being updated.

8.1.48 TunnelingTrafficPolicyPathsExpiringSoon

This alert fires if the last remaining path for a specific domain and traffic matcher expires within three hours. After it expires the packets that match the domain and the traffic matcher cannot be delivered to the destination.

The expiration of paths is set to six hours by default, but paths should be updated much more often. If only three hours are remaining, it means there is a problem with refreshing paths.

Actions

1. Check whether there are other firing alerts, which should give more context on what the problem is
2. Ensure that all services and especially the `daemon` or `control` services are *running*.
3. Check if there are problems with certificate renewal, which could have lead to this issue.
    1. Make sure that the AS certificate is valid, through the /cppki/certificates API endpoint.
    2. If it is not,check the *Certificate/TRC Provisioning* section to manually renew your AS certificate.
    3. Check that *time synchronization* is working properly.

8.2 Common Operations

8.2.1 Gather appliance information

To collect appliance-related information to provide it to the Anapaya Customer Support:

1. SSH to the given machine.
2. Collect general information by running:

```
appliance-cli info > appliance.info
```

3. Fetch the appliance configuration by running:

```
appliance-cli get config > config.json
```

Warning: The appliance config contains secrets, so please remove them before sending the information to anyone!

8.2.2 Gather general host information

To collect host-related information to provide it to the Anapaya Customer Support:

1. SSH to the given machine.
2. Run

```
sudo lshw
```

8.2.3 Check docker services

To check whether the services (run as docker containers) are running:

1. SSH to the given machine
2. Use `docker ps -a`:

```
$ docker ps -a
CONTAINER ID  IMAGE                  COMMAND                  CREATED     STATUS     ␣
↪PORTS  NAMES
c718397beaf9  scion-all:v0.32.2      "/app/scion-all netw..."  7 days ago  Up 7␣
↪days         dataplane-control
5beecfb5d081  vpp-dataplane:v0.32.2  "/usr/bin/vpp -c /sh..."  7 days ago  Up 7␣
↪days         dataplane
...
```

The output of the command shows whether the service is up and for how long it has been running. If the service is up for a very short amount of time, there is a chance that it is crashlooping.

For further information please refer to the official Docker documentation.

8.2.4 Change log level

To change the log level to debug to gather more information when investigating an issue:

1. SSH to the given machine
2. Run the following command to change the debug level of a specific service to debug.

```
appliance-cli services log level <service-name> debug
```

Warning: Don't forget to revert your changes after troubleshooting.

8.2.5 Inspect docker service logs

To inspect the logs of services running as docker containers:

1. SSH to the given machine.
2. If needed, you can use the following command to see the list of services:

```
docker ps -a
```

3. Inspect the logs by running the following command:

```
docker logs <service-name>
```

Note: To see only the recent logs use:

```
docker logs <service-name> --since=<time-duration>
```

For example, to check the logs of the last minute, run:

```
docker logs <service-name> --since=1m
```

Note: The logs are printed to `stderr`.

To save the logs in a file use:

```
docker logs <service-name> 2> <filename>
```

To grep through the logs use

```
docker logs <service-name> 2>&1 | grep <query>
```

For further information please refer to the official Docker documentation.

8.2.6 Restart a service

To restart a service you can use the `appliance-cli`:

```
appliance-cli post debug/services/${service_name}/restart
```

where `${service_name}` is the name of the service you want to restart. To get the possible values for the `${service_name}`, use the following command:

```
appliance-cli get debug/services
```

Note: Alternatively, you can restart a service by running the following commands:

1. SSH to the given machine
2. Run `docker restart <service-name>`

Note: The Anapaya appliance restarts failed services automatically, so manual restarting is likely to be useful only when the service is stuck and/or unresponsive.

For further information, please refer to the official Docker documentation.

8.2.7 Clean up docker images

To remove docker images that are no longer used:

1. SSH to the given machine.
2. List all docker images by running:

```
docker image ls
```

3. Remove old unused images by running:

```
docker image prune
```

For further information please refer to the official Docker documentation.

8.2.8 Connect to the BGP daemon's interactive console

To connect to the BGP daemon's shell:

1. SSH to the given machine.
2. Open the interactive console by running:

```
docker exec -it frr vtysh
```

For further information on the console please refer to the official FRR documentation.

8.2.9 Check systemd services

To check if the systemd services are running:

1. SSH to the given machine
2. Run `systemctl list-units '<service-name>'`:

```
$ systemctl list-units 'appliance*'
UNIT                          LOAD   ACTIVE SUB     DESCRIPTION
appliance-host.service        loaded active running Anapaya Appliance Host Service
appliance-installer.service loaded active running Anapaya Appliance Installer
...
```

3. To get a more detailed overview of a specific service, use `systemctl status <service-name>`:

```
$ systemctl status appliance-installer.service
       appliance-installer.service - Anapaya Appliance Installer
   Loaded: loaded (/etc/systemd/system/appliance-installer.service; enabled; vendor␣
↪preset: enabled)
   Active: active (running) since Mon 2023-02-13 08:26:29 UTC; 7min ago
Main PID: 166 (appliance-insta)
   Tasks: 13 (limit: 38262)
   CGroup: /system.slice/appliance-installer.service
           └─166 /usr/bin/appliance-installer --config /etc/anapaya/installer/
↪appliance-installer.toml
...
```

With these commands you can see whether the service is active and running and for how long it has been running.

Note: A systemd service can be restarted using `systemctl restart <service-name>`.

8.2.10 Inspect systemd service logs

To view the systemd service logs:

1. SSH to the given machine.
2. If needed, you can list the appliance-related services by running:

```
systemctl list-units 'appliance*'
```

3. Inspect the logs by running:

```
journalctl -eu <service-name>
```

Note: To see only the recent logs use the `--since` flag. For example, to see only the logs from today use `journalctl -eu <service-name> --since today`.

To show the most recent 20 entries, use the `-n 20` option.

Note: Note that the logs are printed to `stdout`.

To save the logs in a file use:

```
journalctl -u <service-name> > <filename>
```

To grep through the logs use:

```
journalctl -eu <service-name> | grep <query>
```

8.2.11 Check the systemd-timesyncd service

We use the `systemd-timesyncd` service, which acts as an NTP client and connects to a pool of NTP servers for time synchronization. The following actions provide some starting points for troubleshooting the timesyncd service called `systemd-timesyncd.service`. For further information please refer to the official documentation.

1. SSH to the given machine.
2. Check if the system clock is synchronized and if NTP service is active using:

```
timedatectl status
```

3. *Check if the service is running*.
4. *Check the log of the service*.
5. Restart the service:

```
systemctl restart systemd-timesyncd.service
```

6. Find the configured NTP servers:

```
cat /etc/systemd/timesyncd.conf | grep NTP
```

8.2.12 Disk usage analysis

This section contains some helpful commands that you may need when investigating if you run out of disk space.

1. SSH to the given machine.
2. Check the current space:

```
df -h <path>
```

3. Check the list of the current files:

```
ls -l <path>
```

4. The `du` command can be used to get a more detailed overview of which directory consumes how much space. You can vary the `max-depth` option or the starting directory:

```
du -cha --max-depth=1 / | grep -E "M|G"
```

For further information about the `du` command please refer to the official documentation.

8.2.13 Clean up disk space

There are several ways to free up disk space. The options are divided depending on the context.

APT related

1. Remove outdated packages:

```
sudo apt autoclean
```

2. Remove orphaned packages which are no longer needed:

```
sudo apt autoremove
```

3. Clean the entire APT cache:

```
sudo apt clean
```

Systemd journal logs

1. Check systemd journal logs:

```
journalctl --disk-usage
```

2. Clear the logs that are older than 3 days:

```
sudo journalctl --vacuum-time=3d
```

Docker images

- *Remove unused docker images*.

8.2.14 Fix topology synchronization error

Appliances in a cluster share their topology information with each other. This either happens statically through configuration or dynamically through an exchange protocol. For further information on how to configure topology synchronization in the appliance configuration, refer to *Topology Synchronization*. The instructions below should help to identify a misconfiguration.

1. *Check the logs* of the appliance-controller service. The logs should contain an error describing the misconfiguration.
2. Fix the misconfigured appliances and *update* them.

8.2.15 Inspect SCION paths used for IP-in-SCION tunneling

While troubleshooting SCION connectivity, it is often useful to check the available paths for each domain. This section provides an overview on how to achieve this.

1. SSH to the given machine.
2. Show the currently available paths for all domains and traffic matchers by running the following command. This also shows whether the path is alive, dead (no probes are passing through), expired or similar.

```
appliance-cli inspect scion-tunneling summary --all-paths
```

3. Show the currently used paths for a specific domain.

```
appliance-cli inspect scion-tunneling summary --all-paths \
  --domain <domain>
```

For the used paths for a specific traffic matcher within the given domain, run:

```
appliance-cli inspect scion-tunneling summary --all-paths \
  --domain <domain> --traffic-matcher <traffic matcher>
```

8.2.16 Ping the underlay network

When investigating an issue, it is often helpful to determine whether the underlying IP connectivity is the problem.

For further information, please refer to the official ping documentation.

Tip: The ping command runs indefinitely, unless specified otherwise:

```
ping -c <number> <destination>
```

Changing the source address is possible either directly via the address or the interface name:

```
ping <destination> -I <interface/address>
```

The default time interval between successive packet transmissions is one second. You can specify a custom interval in seconds:

```
ping -i <interval> <destination>
```

CHAPTER

NINE

APPLIANCE-CLI

Manage your Anapaya appliance

9.1 Synopsis

This tool helps you interact with the Anapaya appliance.

9.1.1 Capabilities

The `appliance-cli` acts as a frontend interface for the appliance management API. It can connect to appliances on the same host, or to appliances on remote hosts over the network.

The `appliance-cli` has a group of commands which are generated from the endpoints that are available in the appliance management API. To help you discover the available endpoints, we recommend that you install auto completion. See *appliance-cli completion* for further information. This enables tab completion for the `appliance-cli`, and will complete all the matching endpoints for the generated commands.

The generated commands to interact with the appliance management API are:

- *appliance-cli get*
- *appliance-cli edit*
- *appliance-cli put*
- *appliance-cli post*
- *appliance-cli delete*

Additionally, the tool comes with debugging capabilities that, depending on the command, are only available if the `appliance-cli` is run on the appliance host itself:

- *appliance-cli info*
- *appliance-cli inspect*
- *appliance-cli debug*

9.1.2 Basic Usage

If the `appliance-cli` is run on an appliance host directly and no authentication for the appliance management API is enabled, no additional setup is required.

Tip: Check out *appliance context* if you want to use the `appliance-cli` remotely, or if you want to configure authentication.

For example, you can request the appliance configuration with the following command:

```
appliance-cli get config
```

You will see a response like this:

```
{
  config: {...}
}
```

Tip: By default, the output is the JSON encoding of the response body. You can get access to other response fields or formats. See *output* for more information.

Input

Various inputs can be passed as needed:

```
# Pass a query parameter (flag)
appliance-cli get cppki/trcs --query all=true

# Pass a query parameter (environment)
APPLIANCE_CLI_QUERY="all=true" appliance-cli get cppki/trcs

# Pass a header (flag)
appliance-cli get config --header ETag:W/d393abc

# Pass a header (environment)
APPLIANCE_CLI_HEADER="ETag:W/d393abc" appliance-cli get config

# Pass a file as the body
appliance-cli put config < appliance.json

# Pass body as CLI Shorthand
appliance-cli post cppki/certificates/renew 'subject.isd_as: "1-ff00:0:111"'
```

See the official documentation for more information on how to compose complex payloads with the CLI shorthand.

Tip: If you want to persistently add headers or query parameters for an appliance, consider configuring them in the *appliance context*.

Editing Resources

For endpoints that support both a GET and PUT request, a convenience `edit` command is created which allows you to modify the resource. There are two ways of editing the resource.

You can edit the resource with the CLI shorthand:

```
appliance-cli edit config 'config.management.hostname: host1.site1.org'
```

You can edit the configuration interactively with your editor of choice:

```
appliance-cli edit config -i
```

To use the interactive mode, you need to have the `VISUAL` or `EDITOR` environment variable set. Here, you can pass your editor of choice. E.g., to configure VSCode, you can set the `VISUAL` environment variable:

```
export VISUAL="code --wait"
```

Output

The `appliance-cli` output defaults to the response body in JSON format. Filters are used to tailor the output that you want to see. Via filters, you have access to the response body, the response headers, and the status codes. Furthermore, you can narrow down the information that is relevant to you.

Tip: You can change the body format to YAML with the `--format` flag. E.g., to return the appliance configuration as YAML, run the following command:

```
appliance-cli get config --format yaml
```

A filter must begin with one of the following (or be empty):

- `body`: The response body (default)
- `headers`: The response headers
- `status`: The reponse status code

For example, to only show the response headers, and not the response body pass the following filter:

```
appliance-cli get config -f headers
```

To get a human-readable output that lists both the response body and response headers, pass the empty filter:

```
appliance-cli get config -f ""
```

The filter supports the CLI shorthand querying syntax and allows you to narrow down the output even more. Here are some examples to illustrate this:

Return the hostname of an appliance:

```
$ appliance-cli get config -f body.config.management.hostname

host1.site1.org
```

Show some basic AS properties:

```
$ appliance-cli get config -f 'body.config.scion.ases.{isd_as, core, scion_mtu}'

[
  {
    core: false
    isd_as: "1-ff00:0:1"
    scion_mtu: 1500
  },
  {
    core: true
    isd_as: "2-ff00:0:1"
    scion_mtu: 1472
  },
]
```

List the ISD-AS identifier of all ASes with more than one neighboring AS:

```
$ appliance-cli get config -f 'body.config.scion.ases[neighbors.length > 0].isd_as'

["1-ff00:0:1", "2-ff00:0:1"]
```

List all certificates that are valid on May 4th:

```
$ appliance-cli get cppki/certificates -f 'body.certificate_chains[
  validity.not_before before "2023-05-04" and
  validity.not_after after "2023-05-04"
].{id, validity}'

[
  {
    id: "1b267d9a208603618db8f9dcc02d58906af18e2992e5412e5fa42a715d350841"
    validity: {
      not_after: "2023-05-06T00:50:03Z"
      not_before: "2023-05-03T00:49:33Z"
    }
  }
]
```

9.1.3 Environment Variables

- `APPLIANCE_CLI_PATH`: The directory where the `appliance-cli` stores the configurations and appliance contexts. Defaults to `$HOME/.appliance-cli`.
- `APPLIANCE_CLI_NOCOLOR`: Disable colored output on the tty.
- `HTTP_PROXY` and `NO_PROXY`: Configure proxies for output HTTPS traffic. See http.ProxyFromEnvironment for more information.

You can also pass all flags to the `appliance-cli` via environment variables. Add `APPLIANCE_CLI_<flag>` to your flag of choice. Note that the flag name must be all upper case. E.g., `APPLIANCE_CLI_FORMAT=yaml appliance-cli get config` is equivalent to `appliance-cli get config --format yaml`. The precedence order is: flags > environment variables > configuration file.

9.1.4 Appliance Context

By default, the `appliance-cli` connects to the appliance on the localhost with the default address `127.0.0.1:443`. If you enable basic auth, OAuth2, or want to manage appliances remotely, you will need to set up one (or multiple) appliance contexts explicitly.

When configuring an appliance context, you can define the appliance address and a default profile with the necessary credentials. You can then quickly switch between different appliance contexts, such that you can easily work with various appliances.

To configure a new or edit an existing appliance, run the following command:

```
appliance-cli context configure <name>
```

Note: If you are configuring an appliance with a self-signed certificate, you will need to pass the `--insecure` flag to disable TLS verification.

This interactive command guides you through the configuration, and sets up the the appliance context with a default profile in the `APPLIANCE_CLI_PATH`.

A sample execution to configure basic authorization for an appliance `host1.site1.org` that is running on the local host is shown here:

```
$ appliance-cli context configure host1.site1.org --insecure

? Base URI https://127.0.0.1:443
Setting up a `default` profile
? Select option for profile `default` Setup auth
? API auth type http-basic
? Auth parameter username admin
? Auth parameter password super-secret
? Add additional auth param? No
? Select option for profile `default` Finished with profile
? Select option Save and exit
```

To switch between different appliances, use the following command:

```
$ appliance-cli context select

? Select appliance context  [Use arrows to move, type to filter]
> host1.site1.org (https://127.0.0.1:443)
  host2.site1.org (https://10.00.0.1:443)
```

You can also override this setting for single commands by specifying the `--context` flag or `APPLIANCE_CLI_CONTEXT` environment variable explicitly.

For more information on appliance contexts, consult the help of `appliance-cli context`:

```
appliance-cli context --help
```

An appliance context can have multiple profiles. However, in most cases you are served best by sticking with one `default` profile per appliance context.

9.2 Options

```
-h, --help   help for appliance-cli
```

9.3 SEE ALSO

- *appliance-cli completion* - Generate the autocompletion script for the specified shell
- *appliance-cli context* - Manage the configured appliance contexts
- *appliance-cli crypto* - Useful cryptographic commands
- *appliance-cli debug* - Use the debugging features of the appliance
- *appliance-cli delete* - Delete an API resource
- *appliance-cli edit* - Edit an API resource
- *appliance-cli get* - Request an API resource
- *appliance-cli info* - Show basic appliance information
- *appliance-cli inspect* - Show human-readable debugging reports
- *appliance-cli post* - Create an API resource
- *appliance-cli put* - Create or update an API resource
- *appliance-cli version* - Show the SCION version information

9.3.1 appliance-cli completion

Generate the autocompletion script for the specified shell

Synopsis

Generate the autocompletion script for appliance-cli for the specified shell. See each sub-command's help for details on how to use the generated script.

Options

```
-h, --help   help for completion
```

SEE ALSO

- *appliance-cli* - Manage your Anapaya appliance
- *appliance-cli completion bash* - Generate the autocompletion script for bash
- *appliance-cli completion fish* - Generate the autocompletion script for fish
- *appliance-cli completion powershell* - Generate the autocompletion script for powershell
- *appliance-cli completion zsh* - Generate the autocompletion script for zsh

appliance-cli completion bash

Generate the autocompletion script for bash

Synopsis

Generate the autocompletion script for the bash shell.

This script depends on the 'bash-completion' package. If it is not installed already, you can install it via your OS's package manager.

To load completions in your current shell session:

```
source <(appliance-cli completion bash)
```

To load completions for every new session, execute once:

Linux:

```
appliance-cli completion bash > /etc/bash_completion.d/appliance-cli
```

macOS:

```
appliance-cli completion bash > $(brew --prefix)/etc/bash_completion.d/appliance-cli
```

You will need to start a new shell for this setup to take effect.

```
appliance-cli completion bash
```

Options

```
  -h, --help              help for bash
      --no-descriptions   disable completion descriptions
```

SEE ALSO

- *appliance-cli completion* - Generate the autocompletion script for the specified shell

appliance-cli completion fish

Generate the autocompletion script for fish

Synopsis

Generate the autocompletion script for the fish shell.

To load completions in your current shell session:

```
appliance-cli completion fish | source
```

To load completions for every new session, execute once:

```
appliance-cli completion fish > ~/.config/fish/completions/appliance-cli.fish
```

You will need to start a new shell for this setup to take effect.

```
appliance-cli completion fish [flags]
```

Options

```
  -h, --help              help for fish
      --no-descriptions   disable completion descriptions
```

SEE ALSO

- *appliance-cli completion* - Generate the autocompletion script for the specified shell

appliance-cli completion powershell

Generate the autocompletion script for powershell

Synopsis

Generate the autocompletion script for powershell.

To load completions in your current shell session:

```
appliance-cli completion powershell | Out-String | Invoke-Expression
```

To load completions for every new session, add the output of the above command to your powershell profile.

```
appliance-cli completion powershell [flags]
```

Options

```
  -h, --help              help for powershell
      --no-descriptions   disable completion descriptions
```

SEE ALSO

- *appliance-cli completion* - Generate the autocompletion script for the specified shell

appliance-cli completion zsh

Generate the autocompletion script for zsh

Synopsis

Generate the autocompletion script for the zsh shell.

If shell completion is not already enabled in your environment you will need to enable it. You can execute the following once:

```
echo "autoload -U compinit; compinit" >> ~/.zshrc
```

To load completions in your current shell session:

```
source <(appliance-cli completion zsh)
```

To load completions for every new session, execute once:

Linux:

```
appliance-cli completion zsh > "${fpath[1]}/_appliance-cli"
```

macOS:

```
appliance-cli completion zsh > $(brew --prefix)/share/zsh/site-functions/_appliance-cli
```

You will need to start a new shell for this setup to take effect.

```
appliance-cli completion zsh [flags]
```

Options

```
  -h, --help              help for zsh
      --no-descriptions   disable completion descriptions
```

SEE ALSO

- *appliance-cli completion* - Generate the autocompletion script for the specified shell

9.3.2 appliance-cli context

Manage the configured appliance contexts

Synopsis

Manage the appliance contexts that are configured in the `appliance-cli`.

An appliance context allows the `appliance-cli` to configure authentication schemes and connect to remote appliances. If no appliance context is configured, the `appliance-cli` connects by default to the appliance on the local host.

If multiple appliance contexts are configured in the `appliance-cli`, you will need to select a current context. This determines what appliance is currently targeted by all the commands that are executed by the `appliance-cli`.

The `appliance-cli` supports targeting multiple appliances with different versions. The generated commands that are available depend on the version of the appliance. The `appliance-cli` fetches the API specification directly from the appliance. This API specification is cached locally. After a version upgrade of the appliance, the `appliance-cli` might still have the old API specification cached. You can use the *appliance-cli context sync* to force synchronization of the API specification.

Options

```
  -h, --help   help for context
```

SEE ALSO

- *appliance-cli* - Manage your Anapaya appliance
- *appliance-cli context configure* - Interactively configure an appliance context
- *appliance-cli context current* - Show the currently selected appliance context
- *appliance-cli context delete* - Delete a configured appliance context
- *appliance-cli context list* - List all the configured appliance contexts
- *appliance-cli context select* - Set the current appliance context
- *appliance-cli context show* - Show the configured appliance context
- *appliance-cli context sync* - Synchronize an appliance context

appliance-cli context configure

Interactively configure an appliance context

Synopsis

Create or update an appliance context.

Your are guided through setting up or editing an appliance context. To configure an appliance context, you are required to provide a `base URI`. This URI defines the address where the appliance management API is reachable.

For example, if your appliance is reachable at `172.16.32.64`, you can provide the following base URI:

```
172.16.32.64
```

If the address of your appliance can be looked up via DNS, you can also configure the DNS name in the URI. For example, to configure an appliance with a DNS entry `appliance.example.com`, you can provide the following base URI:

```
appliance.example.com
```

Note: If you do not set the protocol in the URI, https is assumed.

After you have set the base URI, you are asked to configure the default profile. Here, you can set up the authentication credentials (basic auth and Oauth2). If you are done configuring the default profile, you can save and exit.

If you want to start using this appliance context, select it with the *appliance-cli context select* command.

Note: Set the `--insecure` flag during appliance context configuration, if you will be connecting to an appliance with a self-signed certificate. This will disable certificate verification during the TLS handshake for all subsequent calls to the API when using that appliance context.

```
appliance-cli context configure <name> [flags]
```

Examples

```
  context configure host1.site1.org
  context configure host1.site1.org --insecure
```

Options

```
  -h, --help       help for configure
      --insecure   Disable SSL verification
```

SEE ALSO

- *appliance-cli context* - Manage the configured appliance contexts

appliance-cli context current

Show the currently selected appliance context

```
appliance-cli context current [flags]
```

Options

```
  -h, --help   help for current
```

SEE ALSO

- *appliance-cli context* - Manage the configured appliance contexts

appliance-cli context delete

Delete a configured appliance context

```
appliance-cli context delete <name> [flags]
```

Options

```
  -h, --help   help for delete
```

SEE ALSO

- *appliance-cli context* - Manage the configured appliance contexts

appliance-cli context list

List all the configured appliance contexts

```
appliance-cli context list [flags]
```

Options

```
  -h, --help   help for list
```

SEE ALSO

- *appliance-cli context* - Manage the configured appliance contexts

appliance-cli context select

Set the current appliance context

Synopsis

Select the current appliance context from the configured appliance contexts.

If you do not set the appliance context in the arguments, you can chose the appliance context interactively.

By setting the current appliance context, you can decide which appliance will be targeted by all the `appliance-cli` commands that you execute. You can override this choice on every command with the `--context` flag.

```
appliance-cli context select [name] [flags]
```

Examples

```
  context select
  context select host1.site1.org
```

Options

```
  -h, --help   help for select
```

SEE ALSO

- *appliance-cli context* - Manage the configured appliance contexts

appliance-cli context show

Show the configured appliance context

```
appliance-cli context show <name> [flags]
```

Options

```
  -h, --help   help for show
```

SEE ALSO

- *appliance-cli context* - Manage the configured appliance contexts

appliance-cli context sync

Synchronize an appliance context

Synopsis

Force-fetch the latest appliance management API specification.

The API specification is cached for each appliance context individually. This API specification is used to generate the appliance version dependent commands. After a version upgrade of the appliance, you can use this command to force fetch the appropriate API specification.

```
appliance-cli context sync <name> [flags]
```

Options

```
  -h, --help   help for sync
```

SEE ALSO

- *appliance-cli context* - Manage the configured appliance contexts

9.3.3 appliance-cli crypto

Useful cryptographic commands

Options

```
  -h, --help   help for crypto
```

SEE ALSO

- *appliance-cli* - Manage your Anapaya appliance
- *appliance-cli crypto kdf* - Key derivation utility functions for password hashing and verification

appliance-cli crypto kdf

Key derivation utility functions for password hashing and verification

Options

```
  -h, --help   help for kdf
```

SEE ALSO

- *appliance-cli crypto* - Useful cryptographic commands
- *appliance-cli crypto kdf hash* - Create a password hash
- *appliance-cli crypto kdf validate* - Validate the password hash

appliance-cli crypto kdf hash

Create a password hash

Synopsis

Create a hash for the provided password.

Currently, the only supported algorithm is bcrypt.

```
appliance-cli crypto kdf hash [flags]
```

Options

```
  -h, --help   help for hash
```

SEE ALSO

- *appliance-cli crypto kdf* - Key derivation utility functions for password hashing and verification

appliance-cli crypto kdf validate

Validate the password hash

```
appliance-cli crypto kdf validate [flags]
```

Options

```
  -h, --help   help for validate
```

SEE ALSO

- *appliance-cli crypto kdf* - Key derivation utility functions for password hashing and verification

9.3.4 appliance-cli debug

Use the debugging features of the appliance

Synopsis

This command group helps you interact with the Anapaya appliance debugging features.

Because it exposes appliance internals, it is only supposed to be used during debugging and is not meant for regular use. Be cautious with the actions that you take.

Options

```
  -h, --help   help for debug
```

SEE ALSO

- *appliance-cli* - Manage your Anapaya appliance
- *appliance-cli debug notifications* - Show the periodic notification status in the appliance controller

appliance-cli debug notifications

Show the periodic notification status in the appliance controller

Synopsis

Show whether the periodic notifications are enabled or disabled.

The appliance controller sends periodically notifications to the service managers, which ensures the service configurations are up-to-date. This ensures eventual consistency for the service configurations. However, it can be useful during debugging to disable the notifications, such that you can do manual changes to the service configuration. If the notifications are enabled, manual changes will be overwritten as soon as the period has passed.

```
appliance-cli debug notifications [sub-command] [flags]
```

Examples

```
  appliance-cli debug notifications --context host1.site1.org
```

Options

```
      --context string   Select the appliance context
  -h, --help             help for notifications
      --insecure         Disable SSL verification
```

SEE ALSO

- *appliance-cli debug* - Use the debugging features of the appliance
- *appliance-cli debug notifications disable* - Disables the periodic notification updates
- *appliance-cli debug notifications enable* - Enables the periodic notification updates

appliance-cli debug notifications disable

Disables the periodic notification updates

Synopsis

Disable the periodic notification updates.

Disabling the notifications causes the service managers in the appliance controller to not synchronize the configuration. For the time frame that the notifications are disabled, you can add manual changes without them being overwritten automatically.

The notifications are only disabled for a certain amount of time, to prevent you from forgetting to re-enable them. You can pass `indefinite` to pause notifications indefinitely.

> **Warning:** Disabling notifications indefinitely should be used with caution. The appliance will not apply any configuration changes or reconciliate state.

```
appliance-cli debug notifications disable [flags]
```

Examples

```
  appliance-cli debug notifications disable
  appliance-cli debug notifications disable --timeout 3h30m
  appliance-cli debug notifications disable --timeout indefinite
```

Options

```
      --context string   Select the appliance context
  -h, --help             help for disable
      --insecure         Disable SSL verification
      --timeout string   Configure the disable duration (duration string or 'indefinite
↪') (default "1h")
```

SEE ALSO

- *appliance-cli debug notifications* - Show the periodic notification status in the appliance controller

appliance-cli debug notifications enable

Enables the periodic notification updates

```
appliance-cli debug notifications enable [flags]
```

Examples

```
  appliance-cli debug notifications enable
```

Options

```
      --context string   Select the appliance context
  -h, --help             help for enable
      --insecure         Disable SSL verification
```

SEE ALSO

- *appliance-cli debug notifications* - Show the periodic notification status in the appliance controller

9.3.5 appliance-cli delete

Delete an API resource

Synopsis

Delete a specific API resource

The available API resources are generated based on the release version of the appliance. To discover the available endpoints, have a look at the `--help` out. You can also use tab completion, or refer to the appliance management API specification to discover them.

Tip: To enable tab completion, refer to *appliance-cli completion*

You can pass parameters with help of the CLI shorthand. Have a look at the *basic usage section* of the `appliance-cli --help`, or refer to the official documentation.

You can filter the output to narrow down the information that is relevant to you. Also have a look at the *basic usage section* of the `appliance-cli --help`, or refer to the CLI shorthand querying syntax

Authentication

Your deployment setup may require authentication. If you receive an Unauthorized response code (401/403), it indicates that the authentication configuration is missing or faulty. Please follow these steps to set it up correctly:

1. Run `appliance-cli context configure <name>` to add a new configuration, or amend an existing configuration.
2. Ensure the correct base address is set.
3. Setup authentication credentials, if necessary (`http-basic`, `auth2`)

If you have multiple profiles in an appliance context, make sure you have selected the correct one. The `appliance-cli` uses `default` unless specified otherwise. You can override this by passing the `--profile` flag.

```
appliance-cli delete <resource> [flags]
```

Examples

```
  appliance-cli delete software/scion/packages/local/v1.0.1
```

Options

```
      --ca-cert string   Set trusted CA certificate
      --context string   Select the appliance context
  -f, --filter string    Filter response using CLI shorthand (default "body")
      --format string    Output format [auto, json, yaml, table] (default "auto")
      --header strings   Add custom header(s)
  -h, --help             help for delete
      --insecure         Disable SSL verification
      --no-cache         Disable HTTP cache (default true)
      --no-paginate      Disable auto-pagination
      --profile string   Select authentication profile (default "default")
      --query strings    Add custom query parameter(s)
      --raw              Output response in raw format
  -v, --verbose          Enable verbose log output
```

SEE ALSO

- *appliance-cli* - Manage your Anapaya appliance

9.3.6 appliance-cli edit

Edit an API resource

Synopsis

Conveniently edit a specific API resource. This is a combination of an invocation to `appliance-cli get` and `appliance-cli put` on the endpoint.

You can either use interactive mode or pass modifications with CLI shorthand syntax as arguments.

To launch the interactive mode, you will need to pass the `-i` flag. You will need to have either the `VISUAL` or `EDITOR` environment variables set, which configures the editor of your choice. E.g., to configure VSCode, you can set the `VISUAL` environment variable:

```
export VISUAL="code --wait"
```

To configure vim, you can set the `EDITOR` environment variable:

```
export EDITOR="vim"
```

To configure nano, you can set the `EDITOR` environment variable:

```
export EDITOR="nano"
```

The available API resources are generated based on the release version of the appliance. To discover the available endpoints, have a look at the `--help` out. You can also use tab completion, or refer to the appliance management API specification to discover them.

Tip: To enable tab completion, refer to *appliance-cli completion*

You can pass parameters with help of the CLI shorthand. Have a look at the *basic usage section* of the `appliance-cli --help`, or refer to the official documentation.

You can filter the output to narrow down the information that is relevant to you. Also have a look at the *basic usage section* of the `appliance-cli --help`, or refer to the CLI shorthand querying syntax

Authentication

Your deployment setup may require authentication. If you receive an Unauthorized response code (401/403), it indicates that the authentication configuration is missing or faulty. Please follow these steps to set it up correctly:

1. Run `appliance-cli context configure <name>` to add a new configuration, or amend an existing configuration.
2. Ensure the correct base address is set.
3. Setup authentication credentials, if necessary (`http-basic`, `auth2`)

If you have multiple profiles in an appliance context, make sure you have selected the correct one. The `appliance-cli` uses `default` unless specified otherwise. You can override this by passing the `--profile` flag. `

```
appliance-cli edit <resource> [arguments] [flags]
```

Examples

```
appliance-cli edit -i config
appliance-cli edit config 'config.management.hostname: host1.site1.org'
```

Options

```
      --ca-cert string       Set trusted CA certificate
      --context string       Select the appliance context
      --edit-format string   Format to edit [json, yaml] (default "json")
  -f, --filter string        Filter response using CLI shorthand (default "body")
      --format string        Output format [auto, json, yaml, table] (default "auto")
      --header strings       Add custom header(s)
  -h, --help                 help for edit
      --insecure             Disable SSL verification
  -i, --interactive          Open an interactive editor
      --no-cache             Disable HTTP cache (default true)
      --no-paginate          Disable auto-pagination
      --profile string       Select authentication profile (default "default")
      --query strings        Add custom query parameter(s)
      --raw                  Output response in raw format
  -v, --verbose              Enable verbose log output
  -y, --yes                  Disable prompt (answer yes automatically)
```

SEE ALSO

- *appliance-cli* - Manage your Anapaya appliance

9.3.7 appliance-cli get

Request an API resource

Synopsis

Request a specific API resource.

The available API resources are generated based on the release version of the appliance. To discover the available endpoints, have a look at the `--help` out. You can also use tab completion, or refer to the appliance management API specification to discover them.

Tip: To enable tab completion, refer to *appliance-cli completion*

You can pass parameters with help of the CLI shorthand. Have a look at the *basic usage section* of the `appliance-cli --help`, or refer to the official documentation.

You can filter the output to narrow down the information that is relevant to you. Also have a look at the *basic usage section* of the `appliance-cli --help`, or refer to the CLI shorthand querying syntax

Authentication

Your deployment setup may require authentication. If you receive an Unauthorized response code (401/403), it indicates that the authentication configuration is missing or faulty. Please follow these steps to set it up correctly:

1. Run `appliance-cli context configure <name>` to add a new configuration, or amend an existing configuration.
2. Ensure the correct base address is set.
3. Setup authentication credentials, if necessary (`http-basic`, `auth2`)

If you have multiple profiles in an appliance context, make sure you have selected the correct one. The `appliance-cli` uses `default` unless specified otherwise. You can override this by passing the `--profile` flag.

```
appliance-cli get <resource> [flags]
```

Examples

```
  appliance-cli get config --profile with-auth
  appliance-cli get config -f body.config.scion_tunneling
  appliance-cli get cppki/trcs
```

Options

```
      --ca-cert string   Set trusted CA certificate
      --context string   Select the appliance context
  -f, --filter string    Filter response using CLI shorthand (default "body")
      --format string    Output format [auto, json, yaml, table] (default "auto")
      --header strings   Add custom header(s)
  -h, --help             help for get
      --insecure         Disable SSL verification
      --no-cache         Disable HTTP cache (default true)
      --no-paginate      Disable auto-pagination
      --profile string   Select authentication profile (default "default")
      --query strings    Add custom query parameter(s)
      --raw              Output response in raw format
  -v, --verbose          Enable verbose log output
```

SEE ALSO

- *appliance-cli* - Manage your Anapaya appliance

9.3.8 appliance-cli info

Show basic appliance information

```
appliance-cli info [sub-command] [flags]
```

Examples

```
  appliance-cli info --context host1.site1.org
```

Options

```
      --context string   Select the appliance context
      --format string    The Output format (human|json|yaml) (default "human")
  -h, --help             help for info
      --insecure         Disable SSL verification
```

SEE ALSO

- *appliance-cli* - Manage your Anapaya appliance
- *appliance-cli info scion* - Show SCION configuration information
- *appliance-cli info tunneling* - Show IP-in-SCION tunneling configuration information

appliance-cli info scion

Show SCION configuration information

```
appliance-cli info scion [flags]
```

Examples

```
  appliance-cli info scion --context host1.site1.org
  appliance-cli info scion --format yaml
```

Options

```
      --context string   Select the appliance context
      --format string    The Output format (yaml|json) (default "yaml")
  -h, --help             help for scion
      --insecure         Disable SSL verification
```

SEE ALSO

- *appliance-cli info* - Show basic appliance information

appliance-cli info tunneling

Show IP-in-SCION tunneling configuration information

```
appliance-cli info tunneling [flags]
```

Examples

```
  appliance-cli info tunneling --context host1.site1.org
  appliance-cli info tunneling --format yaml
```

Options

```
      --context string   Select the appliance context
      --format string    The Output format (yaml|json) (default "yaml")
  -h, --help             help for tunneling
      --insecure         Disable SSL verification
```

SEE ALSO

- *appliance-cli info* - Show basic appliance information

9.3.9 appliance-cli inspect

Show human-readable debugging reports

Options

```
  -h, --help   help for inspect
```

SEE ALSO

- *appliance-cli* - Manage your Anapaya appliance
- *appliance-cli inspect tunneling* - Show human-readable IP-in-SCION tunneling-related reports

appliance-cli inspect tunneling

Show human-readable IP-in-SCION tunneling-related reports

Options

```
  -h, --help   help for tunneling
```

SEE ALSO

- *appliance-cli inspect* - Show human-readable debugging reports
- *appliance-cli inspect tunneling paths* - Show monitored SCION paths
- *appliance-cli inspect tunneling summary* - Show IP-over-SCION tunneling summary

appliance-cli inspect tunneling paths

Show monitored SCION paths

Synopsis

Show monitored SCION paths. If fingerprint is specified only the paths with the specified fingerprint prefix are shown.

```
appliance-cli inspect tunneling paths [fingerprint...] [flags]
```

Options

```
      --context string   Select the appliance context
  -h, --help             help for paths
      --insecure         Disable SSL verification
```

SEE ALSO

- *appliance-cli inspect tunneling* - Show human-readable IP-in-SCION tunneling-related reports

appliance-cli inspect tunneling summary

Show IP-over-SCION tunneling summary

Synopsis

Show IP-over-SCION tunneling summary. The summary is a multi-level hierarchy consisting of: domains. sets of prefixes, traffic matchers, path filters, remotes and SCION paths. Various command line options can be used to customize the output.

```
appliance-cli inspect tunneling summary [flags]
```

Options

```
      --all-paths                If set, all eligible paths, i.e. all known paths not␣
↪rejected by the path policy
                                 are displayed. (The path selected for the given remote␣
↪will be marked by > sign.)
                                 Otherwise only the currently selected path for each␣
↪remote is shown.
      --context string           Select the appliance context
      --domain strings           Domains to display. If none, show all domains.
  -h, --help                     help for summary
      --insecure                 Disable SSL verification
      --no-hops                  If set, paths are displayed as fingerprints rather␣
↪than a list of hops.
      --prefixes int             Maximal number of prefixes to display. If negative,␣
↪all prefixes are displayed. (default 5)
      --traffic-matcher strings  Traffic matchers to display. If none, show all traffic␣
↪matchers.
```

SEE ALSO

- *appliance-cli inspect tunneling* - Show human-readable IP-in-SCION tunneling-related reports

9.3.10 appliance-cli post

Create an API resource

Synopsis

Create a new API resource.

The available API resources are generated based on the release version of the appliance. To discover the available endpoints, have a look at the `--help` out. You can also use tab completion, or refer to the appliance management API specification to discover them.

Tip: To enable tab completion, refer to *appliance-cli completion*

You can pass parameters with help of the CLI shorthand. Have a look at the *basic usage section* of the `appliance-cli --help`, or refer to the official documentation.

You can filter the output to narrow down the information that is relevant to you. Also have a look at the *basic usage section* of the `appliance-cli --help`, or refer to the CLI shorthand querying syntax

Authentication

Your deployment setup may require authentication. If you receive an Unauthorized response code (401/403), it indicates that the authentication configuration is missing or faulty. Please follow these steps to set it up correctly:

1. Run `appliance-cli context configure <name>` to add a new configuration, or amend an existing configuration.
2. Ensure the correct base address is set.
3. Setup authentication credentials, if necessary (`http-basic`, `auth2`)

If you have multiple profiles in an appliance context, make sure you have selected the correct one. The `appliance-cli` uses `default` unless specified otherwise. You can override this by passing the `--profile` flag.

```
appliance-cli post <resource> [arguments] [flags]
```

Examples

```
appliance-cli post cppki/trcs <trc.pem
appliance-cli post cppki/certificates/renew 'subject.isd_as: "1-ff00:0:111"'
```

Options

```
      --ca-cert string   Set trusted CA certificate
      --context string   Select the appliance context
  -f, --filter string    Filter response using CLI shorthand (default "body")
      --format string    Output format [auto, json, yaml, table] (default "auto")
      --header strings   Add custom header(s)
  -h, --help             help for post
      --insecure         Disable SSL verification
      --no-cache         Disable HTTP cache (default true)
      --no-paginate      Disable auto-pagination
      --profile string   Select authentication profile (default "default")
      --query strings    Add custom query parameter(s)
      --raw              Output response in raw format
  -v, --verbose          Enable verbose log output
```

SEE ALSO

- *appliance-cli* - Manage your Anapaya appliance

9.3.11 appliance-cli put

Create or update an API resource

Synopsis

Create or update a specific API resource.

The available API resources are generated based on the release version of the appliance. To discover the available endpoints, have a look at the `--help` out. You can also use tab completion, or refer to the appliance management API specification to discover them.

Tip: To enable tab completion, refer to *appliance-cli completion*

You can pass parameters with help of the CLI shorthand. Have a look at the *basic usage section* of the `appliance-cli --help`, or refer to the official documentation.

You can filter the output to narrow down the information that is relevant to you. Also have a look at the *basic usage section* of the `appliance-cli --help`, or refer to the CLI shorthand querying syntax

Authentication

Your deployment setup may require authentication. If you receive an Unauthorized response code (401/403), it indicates that the authentication configuration is missing or faulty. Please follow these steps to set it up correctly:

1. Run `appliance-cli context configure <name>` to add a new configuration, or amend an existing configuration.
2. Ensure the correct base address is set.
3. Setup authentication credentials, if necessary (`http-basic`, `auth2`)

If you have multiple profiles in an appliance context, make sure you have selected the correct one. The `appliance-cli` uses `default` unless specified otherwise. You can override this by passing the `--profile` flag.

```
appliance-cli put <resource> [arguments] [flags]
```

Examples

```
  appliance-cli put config <appliance.json
```

Options

```
      --ca-cert string   Set trusted CA certificate
      --context string   Select the appliance context
  -f, --filter string    Filter response using CLI shorthand (default "body")
      --format string    Output format [auto, json, yaml, table] (default "auto")
      --header strings   Add custom header(s)
  -h, --help             help for put
      --insecure         Disable SSL verification
      --no-cache         Disable HTTP cache (default true)
```

(continues on next page)

(continued from previous page)

```
      --no-paginate      Disable auto-pagination
      --profile string   Select authentication profile (default "default")
      --query strings    Add custom query parameter(s)
      --raw              Output response in raw format
  -v, --verbose          Enable verbose log output
```

SEE ALSO

- *appliance-cli* - Manage your Anapaya appliance

9.3.12 appliance-cli version

Show the SCION version information

```
appliance-cli version [flags]
```

Examples

```
  appliance-cli version
```

Options

```
  -h, --help   help for version
```

SEE ALSO

- *appliance-cli* - Manage your Anapaya appliance

CHAPTER

TEN

SCION

SCION networking utilities.

10.1 Synopsis

scion is a collection of command line utilities for hosts in the SCION Internet.

10.2 Options

```
  -h, --help   help for scion
```

10.3 SEE ALSO

- *scion address* - Show (one of) this host's SCION address(es)
- *scion completion* - Generate the autocompletion script for the specified shell
- *scion ping* - Test connectivity to a remote SCION host using SCMP echo packets
- *scion showpaths* - Display paths to a SCION AS
- *scion traceroute* - Trace the SCION route to a remote SCION AS using SCMP traceroute packets
- *scion version* - Show the SCION version information

10.3.1 scion address

Show (one of) this host's SCION address(es)

Synopsis

'address' show address information about this SCION host.

This command returns the relevant SCION address information for this host.

Currently, this returns a sensible but arbitrary local address. In the general case, the host could have multiple SCION addresses.

```
scion address [flags]
```

Examples

```
  scion address
```

Options

```
      --dispatcher string   Path to the dispatcher socket (default "/run/shm/dispatcher/
↪default.sock")
  -h, --help                help for address
      --isd-as isd-as       The local ISD-AS to use. (default 0-0)
      --json                Write the output as machine readable json
  -l, --local ip            Local IP address to listen on. (default invalid IP)
      --sciond string       SCION Daemon address. (default "127.0.0.1:30255")
```

SEE ALSO

- *scion* - SCION networking utilities.

10.3.2 scion completion

Generate the autocompletion script for the specified shell

Synopsis

Generate the autocompletion script for scion for the specified shell. See each sub-command's help for details on how to use the generated script.

Options

```
  -h, --help   help for completion
```

SEE ALSO

- *scion* - SCION networking utilities.
- *scion completion bash* - Generate the autocompletion script for bash
- *scion completion fish* - Generate the autocompletion script for fish
- *scion completion powershell* - Generate the autocompletion script for powershell
- *scion completion zsh* - Generate the autocompletion script for zsh

scion completion bash

Generate the autocompletion script for bash

Synopsis

Generate the autocompletion script for the bash shell.

This script depends on the 'bash-completion' package. If it is not installed already, you can install it via your OS's package manager.

To load completions in your current shell session:

```
source <(scion completion bash)
```

To load completions for every new session, execute once:

Linux:

```
scion completion bash > /etc/bash_completion.d/scion
```

macOS:

```
scion completion bash > $(brew --prefix)/etc/bash_completion.d/scion
```

You will need to start a new shell for this setup to take effect.

```
scion completion bash
```

Options

```
  -h, --help              help for bash
      --no-descriptions   disable completion descriptions
```

SEE ALSO

- *scion completion* - Generate the autocompletion script for the specified shell

scion completion fish

Generate the autocompletion script for fish

Synopsis

Generate the autocompletion script for the fish shell.

To load completions in your current shell session:

```
scion completion fish | source
```

To load completions for every new session, execute once:

```
scion completion fish > ~/.config/fish/completions/scion.fish
```

You will need to start a new shell for this setup to take effect.

```
scion completion fish [flags]
```

Options

```
  -h, --help              help for fish
      --no-descriptions   disable completion descriptions
```

SEE ALSO

- *scion completion* - Generate the autocompletion script for the specified shell

scion completion powershell

Generate the autocompletion script for powershell

Synopsis

Generate the autocompletion script for powershell.

To load completions in your current shell session:

```
scion completion powershell | Out-String | Invoke-Expression
```

To load completions for every new session, add the output of the above command to your powershell profile.

```
scion completion powershell [flags]
```

Options

```
  -h, --help              help for powershell
      --no-descriptions   disable completion descriptions
```

SEE ALSO

- *scion completion* - Generate the autocompletion script for the specified shell

scion completion zsh

Generate the autocompletion script for zsh

Synopsis

Generate the autocompletion script for the zsh shell.

If shell completion is not already enabled in your environment you will need to enable it. You can execute the following once:

```
echo "autoload -U compinit; compinit" >> ~/.zshrc
```

To load completions in your current shell session:

```
source <(scion completion zsh)
```

To load completions for every new session, execute once:

Linux:

```
scion completion zsh > "${fpath[1]}/_scion"
```

macOS:

```
scion completion zsh > $(brew --prefix)/share/zsh/site-functions/_scion
```

You will need to start a new shell for this setup to take effect.

```
scion completion zsh [flags]
```

Options

```
  -h, --help              help for zsh
      --no-descriptions   disable completion descriptions
```

SEE ALSO

- *scion completion* - Generate the autocompletion script for the specified shell

10.3.3 scion ping

Test connectivity to a remote SCION host using SCMP echo packets

Synopsis

'ping' test connectivity to a remote SCION host using SCMP echo packets.

When the –count option is set, ping sends the specified number of SCMP echo packets and reports back the statistics.

When the –healthy-only option is set, ping first determines healthy paths through probing and chooses amongst them.

If no reply packet is received at all, ping will exit with code 1. On other errors, ping will exit with code 2.

The paths can be filtered according to a sequence. A sequence is a string of space separated HopPredicates. A Hop Predicate (HP) is of the form 'ISD-AS#IF,IF'. The first IF means the inbound interface (the interface where packet enters the AS) and the second IF means the outbound interface (the interface where packet leaves the AS). 0 can be used as a wildcard for ISD, AS and both IF elements independently.

HopPredicate Examples:

== ================== Match any: 0 Match ISD 1: 1 Match AS 1-ff00:0:133: 1-ff00:0:133 Match IF 2 of AS 1-ff00:0:133: 1-ff00:0:133#2 Match inbound IF 2 of AS 1-ff00:0:133: 1-ff00:0:133#2,0 Match outbound IF 2 of AS 1-ff00:0:133: 1-ff00:0:133#0,2 == ==================

Sequence Examples:

========== === sequence: "1-ff00:0:133#0 1-ff00:0:120#2,1 0 0 1-ff00:0:110#0" ========== ==========

The above example specifies a path from any interface in AS 1-ff00:0:133 to two subsequent interfaces in AS 1-ff00:0:120 (entering on interface 2 and exiting on interface 1), then there are two wildcards that each match any AS. The path must end with any interface in AS 1-ff00:0:110.

========== === sequence: "1-ff00:0:133#1 1+ 2-ff00:0:1? 2-ff00:0:233#1" ========== ==========

The above example includes operators and specifies a path from interface 1-ff00:0:133#1 through multiple ASes in ISD 1, that may (but does not need to) traverse AS 2-ff00:0:1 and then reaches its destination on 2-ff00:0:233#1.

Available operators:

====== == ? (the preceding HopPredicate may appear at most once) + (the preceding ISD-level HopPredicate must appear at least once) * (the preceding ISD-level HopPredicate may appear zero or more times) | (logical OR) ====== ==========

```
scion ping [flags] <remote>
```

Examples

```
  scion ping 1-ff00:0:110,10.0.0.1
  scion ping 1-ff00:0:110,10.0.0.1 -c 5
```

Options

```
  -c, --count uint16           total number of packets to send
      --dispatcher string      Path to the dispatcher socket (default "/run/shm/
↪dispatcher/default.sock")
      --epic                   Enable EPIC for path probing.
      --format string          Specify the output format (human|json|yaml) (default
↪"human")
      --healthy-only           only use healthy paths
  -h, --help                   help for ping
  -i, --interactive            interactive mode
      --interval duration      time between packets (default 1s)
      --isd-as isd-as          The local ISD-AS to use. (default 0-0)
  -l, --local ip               Local IP address to listen on. (default invalid IP)
      --log.level string       Console logging level verbosity (debug|info|error)
      --max-mtu                choose the payload size such that the sent SCION packet␣
↪including the SCION Header,
                               SCMP echo header and payload are equal to the MTU of the␣
↪path. This flag overrides the
                               'payload_size' and 'packet_size' flags.
      --no-color               disable colored output
      --packet-size uint       number of bytes to be sent including the SCION Header and␣
↪SCMP echo header,
                               the desired size must provide enough space for the␣
↪required headers. This flag
                               overrides the 'payload_size' flag.
  -s, --payload-size uint      number of bytes to be sent in addition to the SCION␣
↪Header and SCMP echo header;
                               the total size of the packet is still variable size due␣
↪to the variable size of
                               the SCION path.
      --refresh                set refresh flag for path request
      --sciond string          SCION Daemon address. (default "127.0.0.1:30255")
      --sequence string        Space separated list of hop predicates
      --timeout duration       timeout per packet (default 1s)
      --tracing.agent string   Tracing agent address
```

SEE ALSO

- *scion* - SCION networking utilities.

10.3.4 scion showpaths

Display paths to a SCION AS

Synopsis

'showpaths' lists available paths between the local and the specified SCION ASe a.

By default, the paths are probed. Paths served from the SCION Daemon's might not forward traffic successfully (e.g. if a network link went down, or there is a black hole on the path). To disable path probing, set the appropriate flag.

If no alive path is discovered, json output is not enabled, and probing is not disabled, showpaths will exit with the code 1. On other errors, showpaths will exit with code 2.

The paths can be filtered according to a sequence. A sequence is a string of space separated HopPredicates. A Hop Predicate (HP) is of the form 'ISD-AS#IF,IF'. The first IF means the inbound interface (the interface where packet enters the AS) and the second IF means the outbound interface (the interface where packet leaves the AS). 0 can be used as a wildcard for ISD, AS and both IF elements independently.

HopPredicate Examples:

========== ========== Match any: 0 Match ISD 1: 1 Match AS 1-ff00:0:133: 1-ff00:0:133 Match IF 2 of AS 1-ff00:0:133: 1-ff00:0:133#2 Match inbound IF 2 of AS 1-ff00:0:133: 1-ff00:0:133#2,0 Match outbound IF 2 of AS 1-ff00:0:133: 1-ff00:0:133#0,2 ========== ==========

Sequence Examples:

========== ========== sequence: "1-ff00:0:133#0 1-ff00:0:120#2,1 0 0 1-ff00:0:110#0" ========== ==========

The above example specifies a path from any interface in AS 1-ff00:0:133 to two subsequent interfaces in AS 1-ff00:0:120 (entering on interface 2 and exiting on interface 1), then there are two wildcards that each match any AS. The path must end with any interface in AS 1-ff00:0:110.

========== ========== sequence: "1-ff00:0:133#1 1+ 2-ff00:0:1? 2-ff00:0:233#1" ========== ==========

The above example includes operators and specifies a path from interface 1-ff00:0:133#1 through multiple ASes in ISD 1, that may (but does not need to) traverse AS 2-ff00:0:1 and then reaches its destination on 2-ff00:0:233#1.

Available operators:

====== ========== ? (the preceding HopPredicate may appear at most once) + (the preceding ISD-level HopPredicate must appear at least once) * (the preceding ISD-level HopPredicate may appear zero or more times) | (logical OR) ====== ==========

```
scion showpaths [flags]
```

Examples

```
scion showpaths 1-ff00:0:110 --extended
scion showpaths 1-ff00:0:110 --local 127.0.0.55 --json
scion showpaths 1-ff00:0:111 --sequence="0-0#2 0*" # outgoing IfID=2
scion showpaths 1-ff00:0:111 --sequence="0* 0-0#41" # incoming IfID=41 at dstIA
scion showpaths 1-ff00:0:111 --sequence="0* 1-ff00:0:112 0*" # 1-ff00:0:112 on the path
scion showpaths 1-ff00:0:110 --no-probe
```

Options

```
      --dispatcher string      Path to the dispatcher socket (default "/run/shm/
↪dispatcher/default.sock")
      --epic                   Enable EPIC.
  -e, --extended               Show extended path meta data information
      --format string          Specify the output format (human|json|yaml) (default
↪"human")
  -h, --help                   help for showpaths
      --isd-as isd-as          The local ISD-AS to use. (default 0-0)
  -l, --local ip               Local IP address to listen on. (default invalid IP)
      --log.level string       Console logging level verbosity (debug|info|error)
  -m, --maxpaths int           Maximum number of paths that are displayed (default 10)
      --no-color               disable colored output
      --no-probe               Do not probe the paths and print the health status
  -r, --refresh                Set refresh flag for SCION Daemon path request
      --sciond string          SCION Daemon address. (default "127.0.0.1:30255")
      --sequence string        Space separated list of hop predicates
      --timeout duration       Timeout (default 5s)
      --tracing.agent string   Tracing agent address
```

SEE ALSO

- *scion* - SCION networking utilities.

10.3.5 scion traceroute

Trace the SCION route to a remote SCION AS using SCMP traceroute packets

Synopsis

'traceroute' traces the SCION path to a remote AS using SCMP traceroute packets.

If any packet is dropped, traceroute will exit with code 1. On other errors, traceroute will exit with code 2. The paths can be filtered according to a sequence. A sequence is a string of space separated HopPredicates. A Hop Predicate (HP) is of the form 'ISD-AS#IF,IF'. The first IF means the inbound interface (the interface where packet enters the AS) and the second IF means the outbound interface (the interface where packet leaves the AS). 0 can be used as a wildcard for ISD, AS and both IF elements independently.

HopPredicate Examples:

========== ========== Match any: 0 Match ISD 1: 1 Match AS 1-ff00:0:133: 1-ff00:0:133 Match IF 2 of AS 1-ff00:0:133: 1-ff00:0:133#2 Match inbound IF 2 of AS 1-ff00:0:133: 1-ff00:0:133#2,0 Match outbound IF 2 of AS 1-ff00:0:133: 1-ff00:0:133#0,2 ========== ==========

Sequence Examples:

========== ========== sequence: "1-ff00:0:133#0 1-ff00:0:120#2,1 0 0 1-ff00:0:110#0" ========== ==========

The above example specifies a path from any interface in AS 1-ff00:0:133 to two subsequent interfaces in AS 1-ff00:0:120 (entering on interface 2 and exiting on interface 1), then there are two wildcards that each match any AS. The path must end with any interface in AS 1-ff00:0:110.

========== ========== sequence: "1-ff00:0:133#1 1+ 2-ff00:0:1? 2-ff00:0:233#1" ========== ==========

The above example includes operators and specifies a path from interface 1-ff00:0:133#1 through multiple ASes in ISD 1, that may (but does not need to) traverse AS 2-ff00:0:1 and then reaches its destination on 2-ff00:0:233#1.

Available operators:

====== ========== ? (the preceding HopPredicate may appear at most once) + (the preceding ISD-level HopPredicate must appear at least once) * (the preceding ISD-level HopPredicate may appear zero or more times) | (logical OR) ====== ==========

```
scion traceroute [flags] <remote>
```

Examples

```
  scion traceroute 1-ff00:0:110,10.0.0.1
```

Options

```
      --dispatcher string      Path to the dispatcher socket (default "/run/shm/
↪dispatcher/default.sock")
      --epic                   Enable EPIC.
      --format string          Specify the output format (human|json|yaml) (default
↪"human")
  -h, --help                   help for traceroute
  -i, --interactive            interactive mode
      --isd-as isd-as          The local ISD-AS to use. (default 0-0)
  -l, --local ip               Local IP address to listen on. (default invalid IP)
      --log.level string       Console logging level verbosity (debug|info|error)
      --no-color               disable colored output
      --refresh                set refresh flag for path request
      --sciond string          SCION Daemon address. (default "127.0.0.1:30255")
      --sequence string        Space separated list of hop predicates
      --timeout duration       timeout per packet (default 1s)
      --tracing.agent string   Tracing agent address
```

SEE ALSO

- *scion* - SCION networking utilities.

10.3.6 scion version

Show the SCION version information

```
scion version [flags]
```

Examples

```
  scion version
```

Options

```
  -h, --help   help for version
```

SEE ALSO

- *scion* - SCION networking utilities.

CHAPTER

ELEVEN

SCION-PKI

SCION Control Plane PKI Management Tool

11.1 Options

```
-h, --help   help for scion-pki
```

11.2 SEE ALSO

- *scion-pki certificate* - Manage certificates for the SCION control plane PKI.
- *scion-pki completion* - Generate the autocompletion script for the specified shell
- *scion-pki key* - Manage private and public keys
- *scion-pki trc* - Manage TRCs for the SCION control plane PKI
- *scion-pki version* - Show the scion-pki version information

11.2.1 scion-pki certificate

Manage certificates for the SCION control plane PKI.

Options

```
-h, --help   help for certificate
```

SEE ALSO

- *scion-pki* - SCION Control Plane PKI Management Tool
- *scion-pki certificate create* - Create a certificate or certificate signing request
- *scion-pki certificate fingerprint* - Calculate the SHA256 fingerprint of a certificate or certificate chain
- *scion-pki certificate inspect* - Inspect a certificate or a certificate signing request
- *scion-pki certificate match* - Match the certificate with other trust objects

- *scion-pki certificate renew* - Renew an AS certificate
- *scion-pki certificate request* - Request an AS certificate from a CA
- *scion-pki certificate sign* - Sign a certificate based on a certificate signing request
- *scion-pki certificate validate* - Validate a SCION cert according to its type
- *scion-pki certificate verify* - Verify a certificate chain

scion-pki certificate create

Create a certificate or certificate signing request

Synopsis

'create' generates a certificate or a certificate signing request (CSR).

The command takes the following positional arguments:

- is the template for the certificate subject distinguished name.
- is the file path where the certificate or certificate requests is written to. The parent directory must exist and must be writable.
- is the file path where the fresh private key is written to. The parent directory must exist and must be writable.

By default, the command creates a SCION control-plane PKI AS certificate. Another certificate type can be selected by providing the –profile flag. If a certificate chain is desired, specify the –bundle flag.

A fresh key is created in the provided , unless the –key flag is set. If the –key flag is set, an existing private key is used and the is ignored.

The –ca and –ca-key flags are required if a AS certificate or CA certificate is being created. Otherwise, they are not allowed.

The –not-before and –not-after flags can either be a timestamp or a relative time offset from the current time.

A timestamp can be provided in two different formats: unix timestamp and RFC 3339 timestamp. For example, 2021-06-24T12:01:02Z represents 1 minute and 2 seconds after the 12th hour of June 26th, 2021 in UTC.

The relative time offset can be formated as a time duration string with the following units: y, w, d, h, m, s. Negative offsets are also allowed. For example, -1h indicates the time of tool invocation minus one hour. Note that –not-after is relative to the current time if a relative time offset is used, and not to –not-before.

The is the template for the distinguished name of the requested certificate and must either be a x.509 certificate or a JSON file. The common name can be overridden by supplying the –common-name flag.

If it is a x.509 certificate, the subject of the template is used as the subject of the created certificate or certificate chain request.

A valid example for a JSON formatted template::

{ "common_name": "1-ff00:0:110 AS certificate", "country": "CH", "isd_as": "1-ff00:0:110" }

All configurable fields with their type are defined by the following JSON schema::

{ "type": "object", "properties": { "isd_as": { "type": "string" }, "common_name": { "type": "string" }, "country": { "type": "string" }, "locality": { "type": "string" }, "organization": { "type": "string" }, "organizational_unit": { "type": "string" }, "postal_code": { "type": "string" }, "province": { "type": "string" }, "serial_number": { "type": "string" }, "street_address": { "type": "string" }, }, "required": ["isd_as"] }

For more information on JSON schemas, see https://json-schema.org/.

```
scion-pki certificate create [flags] <subject-template> <cert-file> <key-file>
```

Examples

```
  scion-pki certificate create --profile cp-root subject.tmpl cp-root.crt cp-root.key
  scion-pki certificate create --ca cp-ca.crt --ca-key cp-ca.key subject.tmpl chain.pem␣
↪cp-as.key
  scion-pki certificate create --csr subject.tmpl chain.csr cp-as.key
```

Options

```
      --bundle                Bundle the certificate with the issuer certificate as a␣
↪certificate chain
      --ca string             The path to the issuer certificate
      --ca-key string         The path to the issuer private key used to sign the new␣
↪certificate
      --common-name string    The common name that replaces the common name in the␣
↪subject template
      --csr                   Generate a certificate signign request instead of a␣
↪certificate
      --curve string          The elliptic curve to use (P-256|P-384|P-521) (default "P-
↪256")
      --force                 Force overwritting existing files
  -h, --help                  help for create
      --key string            The path to the existing private key to use instead of␣
↪creating a new one
      --not-after time        The NotAfter time of the certificate. Can either be a␣
↪timestamp or an offset.

                              If the value is a timestamp, it is expected to either be an␣
↪RFC 3339 formatted
                              timestamp or a unix timestamp. If the value is a duration,␣
↪it is used as the
                              offset from the current time. (default depends on profile)
      --not-before time       The NotBefore time of the certificate. Can either be a␣
↪timestamp or an offset.

                              If the value is a timestamp, it is expected to either be an␣
↪RFC 3339 formatted
                              timestamp or a unix timestamp. If the value is a duration,␣
↪it is used as the
                              offset from the current time. (default 0s)
      --profile string        The type of certificate to generate (cp-as|cp-ca|cp-
↪root|sensitive-voting|regular-voting) (default "cp-as")
```

SEE ALSO

- *scion-pki certificate* - Manage certificates for the SCION control plane PKI.

scion-pki certificate fingerprint

Calculate the SHA256 fingerprint of a certificate or certificate chain

Synopsis

'fingerprint' computes the SHA256 fingerprint of the raw certificate or certificate chain.

If 'cert-file' contains a single certificate, the SHA256 is computed over the raw DER encoding. If it contains a certificate chain, the SHA256 is computed over the concatenation of the raw DER encoding of the certificates in order of appearance.

If the flag –format is set to "emoji", the format of the output is a string of emojis

```
scion-pki certificate fingerprint [flags] <cert-file>
```

Examples

```
  scion-pki certificate fingerprint ISD1-ASff00_0_110.pem
  scion-pki certificate fingerprint --format emoji ISD1-ASff00_0_110.pem
  scion-pki certificate fingerprint --format hex ISD1-ASff00_0_110.pem
```

Options

```
      --format string   The format of the fingerprint (hex|base64|base64-url|base64-
↪raw|base64-url-raw|emoji). (default "hex")
  -h, --help            help for fingerprint
```

SEE ALSO

- *scion-pki certificate* - Manage certificates for the SCION control plane PKI.

scion-pki certificate inspect

Inspect a certificate or a certificate signing request

Synopsis

outputs the certificate chain or a certificat signing request (CSR) in human readable format.

```
scion-pki certificate inspect [flags] <certificate-file|CSR-file>
```

Examples

```
  scion-pki certificate inspect ISD1-ASff00_0_110.pem
  scion-pki certificate inspect --short ISD1-ASff00_0_110.pem
```

Options

```
  -h, --help    help for inspect
      --short   Print details of certificate or CSR in short format
```

SEE ALSO

- *scion-pki certificate* - Manage certificates for the SCION control plane PKI.

scion-pki certificate match

Match the certificate with other trust objects

Options

```
  -h, --help   help for match
```

SEE ALSO

- *scion-pki certificate* - Manage certificates for the SCION control plane PKI.
- *scion-pki certificate match private* - Find the matching private keys for the certificate

scion-pki certificate match private

Find the matching private keys for the certificate

Synopsis

'private' finds all the matching private keys for the certificate. If the file contains a certificate chain, only the keys authenticated by the first certificate in the chain are considered.

The output contains all the private keys that are authenticated by the certificate.

```
scion-pki certificate match private <certificate> <private-key> [<private-key> ...]␣
↪[flags]
```

Examples

```
  scion-pki certificate match private ISD1-ASff00_0_110.pem cp-as.key
  scion-pki certificate match private ISD1-ASff00_0_110.pem *.key
```

Options

```
  -h, --help               help for private
      --separator string   The separator between file names (default "\n")
```

SEE ALSO

- *scion-pki certificate match* - Match the certificate with other trust objects

scion-pki certificate renew

Renew an AS certificate

Synopsis

'renew' requests a renewed AS certificate from a remote CA control service.

The provided and are used to sign the CSR. They must be valid and verifiable by the CA in order for the request to be served.

The renewed certificate chain is requested with a fresh private key, unless the –reuse-key flag is set.

By default, the target CA for the request is extracted from the certificate chain that is renewed. To select a different CA, you can specify the –ca flag with one or multiple target CAs. If multiple CAs are specified, they are tried in the order that they are declared until the first successful certificate chain renewal. If none of the declared CAs issued a verifiable certificate chain, the command returns a non-zero exit code.

The TRCs are used to validate and verify the renewed certificate chain. If the chain is not verifiable with any of the active TRCs, the certificate chain and, if applicable, the fresh private key are written to the provided file paths with the '.unverified' suffix, where CA is the ISD-AS number of the CA AS that issued the unverifiable certificate chain.

The resulting certificate chain is written to the file system, either to or to –out, if specified.

The fresh private key is is written to the file system, either to or to –out-key, if specified.

Files are not allowed to be overwritten, by default. Either you have to specify the –out and –out-key flags explicitly, or specify the –force or –backup flags. In case the –backup flag is set, every file that would be overwritten is renamed to contain a local execution time timestamp before the file extension. E.g., ...

This command supports the –expires-in flag in order for it to be run in a periodic task runner (e.g., cronjob). The flag indicates the acceptable remaining time before certificate expiration. If the remaining time is larger or equal to the specified value, the command immediately exits with code zero. If the remaining time is less than the specified value, a renewal run is executed. The time can either be specified as a time duration or a relative factor of the existing certificate chain. For the time duration, the following units are supported: d, h, m, s. The relative factor is supplied as a floating point number. For example, a factor of 0.75 indicates that the certificate chain should be renewed after one quarter of its lifetime has passed, and it still has three quarters of its validity period until it expires.

Unless a subject template is specified, the subject of the existing certificate chain is used as the subject for the renewal request.

The template is expressed in JSON. A valid example::

{ “common_name”: “1-ff00:0:110 AS certificate”, “country”: “CH”, “isd_as”: “1-ff00:0:110” }

All configurable fields with their type are defined by the following JSON schema::

{ “type”: “object”, “properties”: { “isd_as”: { “type”: “string” }, “common_name”: { “type”: “string” }, “country”: { “type”: “string” }, “locality”: { “type”: “string” }, “organization”: { “type”: “string” }, “organizational_unit”: { “type”: “string” }, “postal_code”: { “type”: “string” }, “province”: { “type”: “string” }, “serial_number”: { “type”: “string” }, “street_address”: { “type”: “string” }, }, “required”: [“isd_as”] }

For more information on JSON schemas, see https://json-schema.org/.

```
scion-pki certificate renew [flags] <chain-file> <key-file>
```

Examples

```
  scion-pki certificate renew --trc ISD1-B1-S1.trc --backup cp-as.pem cp-as.key
  scion-pki certificate renew --trc ISD1-B1-S1.trc,ISD1-B1-S2.trc --force cp-as.pem cp-
↪as.key
  scion-pki certificate renew --trc ISD1-B1-S1.trc --reuse-key --out cp-as.new.pem cp-as.
↪pem cp-as.key
  scion-pki certificate renew --trc ISD1-B1-S1.trc --backup --expires-in 56h cp-as.pem␣
↪cp-as.key
  scion-pki certificate renew --trc ISD1-B1-S1.trc --backup --expires-in 0.75 cp-as.pem␣
↪cp-as.key
  scion-pki certificate renew --trc ISD1-B1-S1.trc --backup --ca 1-ff00:0:110,1-
↪ff00:0:120 cp-as.pem cp-as.key
  scion-pki certificate renew --trc ISD1-B1-S1.trc --backup \
          --remote 1-ff00:0:110,10.0.0.3 --remote 1-ff00:0:120,172.30.200.2 cp-as.pem cp-
↪as.key
```

Options

```
      --backup                  Back up existing files before overwriting
      --ca strings              Comma-separated list of ISD-AS identifiers of target CAs.
                                The CAs are tried in order until success or all of them␣
↪failed.
                                --ca is mutually exclusive with --remote
      --common-name string      The common name that replaces the common name in the␣
↪subject template
      --curve string            The elliptic curve to use (P-256|P-384|P-521) (default "P-
↪256")
      --dispatcher string       Path to the dispatcher socket (default "/run/shm/
↪dispatcher/default.sock")
      --expires-in string       Remaining time threshold for renewal
      --features strings        enable development features ()
      --force                   Force overwritting existing files
  -h, --help                    help for renew
  -i, --interactive             interactive mode
      --isd-as isd-as           The local ISD-AS to use. (default 0-0)
  -l, --local ip                Local IP address to listen on. (default invalid IP)
      --log.level string        Console logging level verbosity (debug|info|error)
      --no-color                disable colored output
      --no-probe                do not probe paths for health
      --out string              The path to write the renewed certificate chain
      --out-cms string          The path to write the CMS signed CSR sent to the CA
      --out-csr string          The path to write the CSR sent to the CA
      --out-key string          The path to write the fresh private key
      --refresh                 set refresh flag for path request
      --remote stringArray      The remote CA address to use for certificate renewal.
                                The address is of the form <ISD-AS>,<IP>. --remote can be␣
↪specified multiple times
                                and all specified remotes are tried in order until␣
↪success or all of them failed.
                                --remote is mutually exclusive with --ca.
      --reuse-key               Reuse the provided private key instead of creating a␣
↪fresh private key
      --sciond string           SCION Daemon address. (default "127.0.0.1:30255")
      --sequence string         Space separated list of hop predicates
      --subject string          The path to the custom subject for the CSR
      --timeout duration        The timeout for the renewal request per CA (default 10s)
      --tracing.agent string    The tracing agent address
      --trc strings             Comma-separated list of trusted TRC files or glob␣
↪patterns. If more than two TRCs are specified,
                                 only up to two active TRCs with the highest Base version␣
↪are used (required)
```

SEE ALSO

- *scion-pki certificate* - Manage certificates for the SCION control plane PKI.

scion-pki certificate request

Request an AS certificate from a CA

Synopsis

'request' requests an AS certificate from a remote CA using the provided CSR.

The provided are used to sign the CSR provided in . They must be valid and verifiable by the remote CA in order for the request to be served.

By default, the target CA for the request is extracted from the certificate chain that is used to sign the CSR. To select a different CA, you can specify the –ca flag with one or multiple target CAs. If multiple CAs are specified, they are tried in the order that they are declared until the first successful certificate chain renewal. If none of the declared CAs issued a verifiable certificate chain, the command returns a non-zero exit code.

The TRCs are used to validate and verify the renewed certificate chain. If the chain is not verifiable with any of the active TRCs the command returns a non-zero exit code.

The resulting certificate chain is written to stdout by default. This can be changed by specifying the –out flag.

```
scion-pki certificate request [flags] <csr-file> <chain-file> <key-file>
```

Examples

```
  scion-pki certificate request --trc ISD1-B1-S1.trc csr.pem cp-as.pem cp-as.key
  scion-pki certificate request --trc ISD1-B1-S1.trc,ISD1-B1-S2.trc --out cp-as.new.pem␣
↪csr.pem cp-as.pem cp-as.key
  scion-pki certificate request --trc ISD1-B1-S1.trc --ca 1-ff00:0:110,1-ff00:0:111 csr.
↪pem cp-as.pem cp-as.key
  scion-pki certificate request --trc ISD1-B1-S1.trc --remote 1-ff00:0:110,172.30.200.2␣
↪csr.pem cp-as.pem cp-as.key
```

Options

```
      --ca strings             Comma-separated list of ISD-AS identifiers of target CAs.
                               The CAs are tried in order until success or all of them␣
↪failed.
                               --ca is mutually exclusive with --remote
      --dispatcher string      Path to the dispatcher socket (default "/run/shm/
↪dispatcher/default.sock")
  -h, --help                   help for request
  -i, --interactive            interactive mode
      --isd-as isd-as          The local ISD-AS to use. (default 0-0)
```

(continues on next page)

(continued from previous page)

```
  -l, --local ip                Local IP address to listen on. (default invalid IP)
      --log.level string        Console logging level verbosity (debug|info|error)
      --no-color                disable colored output
      --no-probe                do not probe paths for health
      --out string              The path to write the renewed certificate chain
      --refresh                 set refresh flag for path request
      --remote stringArray      The remote CA address to use for certificate renewal.
                                The address is of the form <ISD-AS>,<IP>. --remote can be␣
↪specified multiple times
                                and all specified remotes are tried in order until␣
↪success or all of them failed.
                                --remote is mutually exclusive with --ca.
      --sciond string           SCION Daemon address. (default "127.0.0.1:30255")
      --sequence string         Space separated list of hop predicates
      --timeout duration        The timeout for the renewal request per CA (default 10s)
      --tracing.agent string    The tracing agent address
      --trc strings             Comma-separated list of trusted TRC files or glob␣
↪patterns. If more than two TRCs are specified,
                                 only up to two active TRCs with the highest Base version␣
↪are used (required)
```

SEE ALSO

- *scion-pki certificate* - Manage certificates for the SCION control plane PKI.

scion-pki certificate sign

Sign a certificate based on a certificate signing request

Synopsis

'sign' creates a certificate based on a certificate signing request (CSR).

The command takes the following positional arguments:

- is the file path where the PEM-encoded certificate signing request is located.

By default, the command creates a SCION control-plane PKI AS certificate. Another certificate type can be selected by providing the –profile flag. If a certificate chain is desired, specify the –bundle flag.

The –ca and –ca-key flags are required.

The –not-before and –not-after flags can either be a timestamp or a relative time offset from the current time.

A timestamp can be provided in two different formats: unix timestamp and RFC 3339 timestamp. For example, 2021-06-24T12:01:02Z represents 1 minute and 2 seconds after the 12th hour of June 26th, 2021 in UTC.

The relative time offset can be formated as a time duration string with the following units: y, w, d, h, m, s. Negative offsets are also allowed. For example, -1h indicates the time of tool invocation minus one hour. Note that –not-after is relative to the current time if a relative time offset is used, and not to –not-before.

```
scion-pki certificate sign [flags] <csr-file>
```

Examples

```
  scion-pki certificate sign --ca cp-ca.crt --ca-key cp-ca.key cp-as.csr
  scion-pki certificate sign --profile cp-ca --ca cp-root.crt --ca-key cp-root.key cp-ca.
↪csr
```

Options

```
      --bundle            Bundle the certificate with the issuer certificate as a␣
↪certificate chain
      --ca string         The path to the issuer certificate
      --ca-key string     The path to the issuer private key used to sign the new␣
↪certificate
  -h, --help              help for sign
      --not-after time    The NotAfter time of the certificate. Can either be a␣
↪timestamp or an offset.

                          If the value is a timestamp, it is expected to either be an␣
↪RFC 3339 formatted
                          timestamp or a unix timestamp. If the value is a duration, it␣
↪is used as the
                          offset from the current time. (default depends on profile)
      --not-before time   The NotBefore time of the certificate. Can either be a␣
↪timestamp or an offset.

                          If the value is a timestamp, it is expected to either be an␣
↪RFC 3339 formatted
                          timestamp or a unix timestamp. If the value is a duration, it␣
↪is used as the
                          offset from the current time. (default 0s)
      --profile string    The type of certificate to sign (cp-as|cp-ca) (default "cp-as")
```

SEE ALSO

- *scion-pki certificate* - Manage certificates for the SCION control plane PKI.

scion-pki certificate validate

Validate a SCION cert according to its type

Synopsis

'validate' checks if the certificate is valid and of the specified type.

In case the 'any' type is specified, this command attempts to identify what type a certificate is and validates it accordingly. The identified type is stated in the output.

By default, the command does not check that the certificate is in its validity period. This can be enabled by specifying the –check-time flag.

```
scion-pki certificate validate [flags]
```

Examples

```
  scion-pki certificate validate --type cp-root /tmp/certs/cp-root.crt
  scion-pki certificate validate --type any /tmp/certs/cp-root.crt
```

Options

```
      --check-time          Check that the certificate covers the current time.
      --current-time time   The time that needs to be covered by the certificate.
                            Can either be a timestamp or an offset.

                            If the value is a timestamp, it is expected to either be an␣
↪RFC 3339 formatted
                            timestamp or a unix timestamp. If the value is a duration,␣
↪it is used as the
                            offset from the current time. (default 0s)
  -h, --help                help for validate
      --type string         type of cert (any|chain|cp-as|cp-ca|cp-root|regular-
↪voting|sensitive-voting) (required)
```

SEE ALSO

- *scion-pki certificate* - Manage certificates for the SCION control plane PKI.

scion-pki certificate verify

Verify a certificate chain

Synopsis

'verify' verifies the certificate chains based on a trusted TRC.

The chain must be a PEM bundle with the AS certificate first, and the CA certificate second.

The ISD-AS property of the subject identified by the certificate (or in the case of a certificate chain, the leaf certificate) can be validated by specifying the –subject-isd-as flag and the expected ISD-AS value.

```
scion-pki certificate verify [flags]
```

Examples

```
  scion-pki certificate verify --trc ISD1-B1-S1.trc,ISD1-B1-S2.trc ISD1-ASff00_0_110.pem
  scion-pki certificate verify --trc ISD1-*.trc ISD1-ASff00_0_110.pem
```

Options

```
      --currenttime int         Optional unix timestamp that sets the current time
  -h, --help                    help for verify
      --subject-isd-as string   ISD-AS property of the subject of the certificate
      --trc strings             Comma-separated list of trusted TRC files or glob␣
↪patterns. If more than two TRCs are specified,
                                only up to two active TRCs with the highest Base␣
↪version are used (required)
```

SEE ALSO

- *scion-pki certificate* - Manage certificates for the SCION control plane PKI.
- *scion-pki certificate verify ca* - Verify a CA certificate

scion-pki certificate verify ca

Verify a CA certificate

Synopsis

'ca' verifies the CA certificate based on a trusted TRC.

The CA certificate must be a PEM encoded.

```
scion-pki certificate verify ca [flags]
```

Examples

```
  scion-pki certificate verify --trc ISD1-B1-S1.trc ISD1-ASff00_0_110.ca.crt
```

Options

```
      --currenttime int   Optional unix timestamp that sets the current time
  -h, --help              help for ca
      --trc string        trusted TRC (required)
```

SEE ALSO

- *scion-pki certificate verify* - Verify a certificate chain

11.2.2 scion-pki completion

Generate the autocompletion script for the specified shell

Synopsis

Generate the autocompletion script for scion-pki for the specified shell. See each sub-command's help for details on how to use the generated script.

Options

```
  -h, --help   help for completion
```

SEE ALSO

- *scion-pki* - SCION Control Plane PKI Management Tool
- *scion-pki completion bash* - Generate the autocompletion script for bash
- *scion-pki completion fish* - Generate the autocompletion script for fish
- *scion-pki completion powershell* - Generate the autocompletion script for powershell
- *scion-pki completion zsh* - Generate the autocompletion script for zsh

scion-pki completion bash

Generate the autocompletion script for bash

Synopsis

Generate the autocompletion script for the bash shell.

This script depends on the 'bash-completion' package. If it is not installed already, you can install it via your OS's package manager.

To load completions in your current shell session:

```
source <(scion-pki completion bash)
```

To load completions for every new session, execute once:

Linux:

```
scion-pki completion bash > /etc/bash_completion.d/scion-pki
```

macOS:

```
scion-pki completion bash > $(brew --prefix)/etc/bash_completion.d/scion-pki
```

You will need to start a new shell for this setup to take effect.

```
scion-pki completion bash
```

Options

```
  -h, --help              help for bash
      --no-descriptions   disable completion descriptions
```

SEE ALSO

- *scion-pki completion* - Generate the autocompletion script for the specified shell

scion-pki completion fish

Generate the autocompletion script for fish

Synopsis

Generate the autocompletion script for the fish shell.

To load completions in your current shell session:

```
scion-pki completion fish | source
```

To load completions for every new session, execute once:

```
scion-pki completion fish > ~/.config/fish/completions/scion-pki.fish
```

You will need to start a new shell for this setup to take effect.

```
scion-pki completion fish [flags]
```

Options

```
  -h, --help              help for fish
      --no-descriptions   disable completion descriptions
```

SEE ALSO

- *scion-pki completion* - Generate the autocompletion script for the specified shell

scion-pki completion powershell

Generate the autocompletion script for powershell

Synopsis

Generate the autocompletion script for powershell.

To load completions in your current shell session:

```
scion-pki completion powershell | Out-String | Invoke-Expression
```

To load completions for every new session, add the output of the above command to your powershell profile.

```
scion-pki completion powershell [flags]
```

Options

```
  -h, --help              help for powershell
      --no-descriptions   disable completion descriptions
```

SEE ALSO

- *scion-pki completion* - Generate the autocompletion script for the specified shell

scion-pki completion zsh

Generate the autocompletion script for zsh

Synopsis

Generate the autocompletion script for the zsh shell.

If shell completion is not already enabled in your environment you will need to enable it. You can execute the following once:

```
echo "autoload -U compinit; compinit" >> ~/.zshrc
```

To load completions in your current shell session:

```
source <(scion-pki completion zsh)
```

To load completions for every new session, execute once:

Linux:

```
scion-pki completion zsh > "${fpath[1]}/_scion-pki"
```

macOS:

```
scion-pki completion zsh > $(brew --prefix)/share/zsh/site-functions/_scion-pki
```

You will need to start a new shell for this setup to take effect.

```
scion-pki completion zsh [flags]
```

Options

```
  -h, --help              help for zsh
      --no-descriptions   disable completion descriptions
```

SEE ALSO

- *scion-pki completion* - Generate the autocompletion script for the specified shell

11.2.3 scion-pki key

Manage private and public keys

Options

```
  -h, --help   help for key
```

SEE ALSO

- *scion-pki* - SCION Control Plane PKI Management Tool
- *scion-pki key fingerprint* - Computes the fingerprint of the provided key
- *scion-pki key match* - Match the key with other trust objects
- *scion-pki key private* - Generate private key at the specified location
- *scion-pki key public* - Generate public key for the provided private key
- *scion-pki key symmetric* - Generate symmetric key at the specified location

scion-pki key fingerprint

Computes the fingerprint of the provided key

Synopsis

'fingerprint' computes the fingerprint of the provided key.

The fingerprint of a private key will be based on the public part of the key. For certificates or certificate chains the fingerprint is computed on the public key of the first certificate in the file.

By default the fingerprint calculated is SHA-1 hash of the marshaled public key as defined in https://tools.ietf.org/html/rfc5280#section-4.2.1.2 (1). With the '–full-key-digest' flag, the computed fingerprint is the SHA-1 hash with ASN.1 DER-encoded subjectPublicKey.

The subject key ID is written to standard out.

```
scion-pki key fingerprint [flags] <key-file>
```

Examples

```
scion-pki key fingerprint cp-as.key --format base64
scion-pki key fingerprint ISD1-ASff00_-_110.pem --full-key-digest
```

Options

```
      --format string     The format of the fingerprint (hex|base64|base64-url|base64-
↪raw|base64-url-raw|emoji). (default "emoji")
      --full-key-digest   Calculate the SHA1 sum of the marshaled public key
  -h, --help              help for fingerprint
```

SEE ALSO

- *scion-pki key* - Manage private and public keys

scion-pki key match

Match the key with other trust objects

Options

```
  -h, --help   help for match
```

SEE ALSO

- *scion-pki key* - Manage private and public keys
- *scion-pki key match certificate* - Find the matching certificate for the key

scion-pki key match certificate

Find the matching certificate for the key

Synopsis

'certificate' finds all the matching certificates for the key. If a file contains a certificate chain, only the first certificate in the chain is considered.

The output contains all certificates that authenticate the key.

```
scion-pki key match certificate <private-key> <certificate> [<certificate> ...] [flags]
```

Examples

```
scion-pki key match certificate cp-as.key ISD1-ASff00_0_110.pem
scion-pki key match certificate cp-as.key *.pem
```

Options

```
-h, --help               help for certificate
    --separator string   The separator between file names (default "\n")
```

SEE ALSO

- *scion-pki key match* - Match the key with other trust objects

scion-pki key private

Generate private key at the specified location

Synopsis

'private' generates a PEM encoded private key at the specified location.

The contents are the private key in PKCS #8 ASN.1 DER format.

```
scion-pki key private [flags] <private-key-file>
```

Examples

```
scion-pki key private cp-as.key
scion-pki key private --curve P-384 cp-as.key
```

Options

```
    --curve string   The elliptic curve to use (P-256|P-384|P-521) (default "P-256")
    --force          Force overwritting existing private key
-h, --help           help for private
```

SEE ALSO

- *scion-pki key* - Manage private and public keys

scion-pki key public

Generate public key for the provided private key

Synopsis

'public' generates a PEM encoded public key.

By default, the public key is written to standard out.

```
scion-pki key public [flags] <private-key-file>
```

Examples

```
  scion-pki key public cp-as.key
  scion-pki key public cp-as.key --out cp-as.pub
```

Options

```
      --force        Force overwritting existing public key
  -h, --help         help for public
      --out string   Path to write public key
```

SEE ALSO

- *scion-pki key* - Manage private and public keys

scion-pki key symmetric

Generate symmetric key at the specified location

Synopsis

'symmetric' generates a symmetric key at the specified location.

The content is the symmetrics key in the specified format (base64 or pem with SYMMETRIC KEY block).

```
scion-pki key symmetric [flags] <symmetric-key-file>
```

Examples

```
scion-pki key symmetric master-0.key
scion-pki key symmetric --format base64 --size 512 master-0.key
```

Options

```
      --force           Force overwritting existing symmetric key
      --format string   The output format (pem|base64) (default "pem")
  -h, --help            help for symmetric
      --size int        The number of bits in the symmetric key (default 256)
```

SEE ALSO

- *scion-pki key* - Manage private and public keys

11.2.4 scion-pki trc

Manage TRCs for the SCION control plane PKI

Options

```
  -h, --help   help for trc
```

SEE ALSO

- *scion-pki* - SCION Control Plane PKI Management Tool
- *scion-pki trc combine* - Combine partially signed TRCs
- *scion-pki trc extract* - Extract parts of a signed TRC
- *scion-pki trc format* - Reformat a TRC or TRC payload
- *scion-pki trc inspect* - Represent TRC in a human readable form
- *scion-pki trc payload* - Generate new TRC payload
- *scion-pki trc sign* - Sign a TRC
- *scion-pki trc verify* - Verify a TRC chain

scion-pki trc combine

Combine partially signed TRCs

Synopsis

'combine' combines the signatures on partially signed TRCs into one single TRC. The command checks that all parts sign the same TRC payload content.

No further checks are made. Check that the TRC is valid and verifiable with the appropriate commands.

```
scion-pki trc combine [flags]
```

Examples

```
  scion-pki trc combine --payload ISD1-B1-S1.pld -o ISD1-B1-S1.trc ISD1-B1-S1.org1 ISD1-
↪B1-S1.org2
```

Options

```
      --format string    Output format (der|pem) (default "der")
  -h, --help             help for combine
  -o, --out string       Output file (required)
  -p, --payload string   The TRC payload. If provided, it will be used as a reference␣
↪payload to compare the partially signed TRC payloads against. It can be either DER or␣
↪PEM encoded.
```

SEE ALSO

- *scion-pki trc* - Manage TRCs for the SCION control plane PKI

scion-pki trc extract

Extract parts of a signed TRC

Options

```
  -h, --help   help for extract
```

SEE ALSO

- *scion-pki trc* - Manage TRCs for the SCION control plane PKI
- *scion-pki trc extract certificates* - Extract the bundled certificates
- *scion-pki trc extract payload* - Extract the TRC payload

scion-pki trc extract certificates

Extract the bundled certificates

Synopsis

'certificates' extracts the certificates into a bundeld PEM file.

```
scion-pki trc extract certificates [flags]
```

Examples

```
  scion-pki trc extract certificates -o bundle.pem input.trc
```

Options

```
  -h, --help         help for certificates
  -o, --out string   Output file (required)
```

SEE ALSO

- *scion-pki trc extract* - Extract parts of a signed TRC

scion-pki trc extract payload

Extract the TRC payload

Synopsis

'payload' extracts the asn.1 encoded DER TRC payload.

To inspect the created asn.1 file you can use the openssl tool::

openssl asn1parse -inform DER -i -in payload.der

(for more information see 'man asn1parse')

```
scion-pki trc extract payload [flags]
```

Examples

```
scion-pki trc extract payload -o payload.der input.trc
```

Options

```
      --format string   Output format (der|pem) (default "der")
  -h, --help            help for payload
  -o, --out string      Output file (required)
```

SEE ALSO

- *scion-pki trc extract* - Extract parts of a signed TRC

scion-pki trc format

Reformat a TRC or TRC payload

Synopsis

'format' prints the TRC or TRC payload in a different format.

The PEM type for a TRC is 'TRC', and for a TRC payload it is 'TRC PAYLOAD'.

By default, the output is PEM encoded. DER format can be requested by providing 'der' in the –format flag. When selecting DER output, ensure stdout is redirected to a file because the raw characters might mess up the terminal.

```
scion-pki trc format [flags] <trc-file>
```

Examples

```
scion-pki trc format ISD1-B1-S1.trc.der
scion-pki trc --format der ISD1-B1-S2.pld --out ISD1-B1-S2.pld.der
```

Options

```
      --force           Force overwriting existing output file
      --format string   The Output format (der|pem) (default "pem")
  -h, --help            help for format
      --out string      The path to write the transformation TRC or TRC payload
```

SEE ALSO

- *scion-pki trc* - Manage TRCs for the SCION control plane PKI

scion-pki trc inspect

Represent TRC in a human readable form

Synopsis

'human' outputs the TRC contents in a human readable form.

The input file can either be a TRC payload, or a signed TRC. The output can either be in yaml, or json.

By default, this command attempts to handle decoding errors gracefully. To return an error if parts of a TRC fail to decode, enable the strict mode.

```
scion-pki trc inspect [flags]
```

Examples

```
  scion-pki trc human ISD1-B1-S1.pld.der
  scion-pki trc human ISD1-B1-S1.trc
```

Options

```
      --format string        Output format (yaml|json) (default "yaml")
  -h, --help                 help for inspect
      --predecessor string   Predecessor TRC (needed to display signature purpose)
      --strict               Enable strict decoding mode
```

SEE ALSO

- *scion-pki trc* - Manage TRCs for the SCION control plane PKI

scion-pki trc payload

Generate new TRC payload

Synopsis

'payload' creates the asn.1 encoded der file.

To update an existing TRC the predecessor TRC needs to be specified.

To inspect the created asn.1 file you can use the openssl tool::

openssl asn1parse -inform DER -i -in payload.der

(for more information see 'man asn1parse')

```
scion-pki trc payload [flags]
```

Examples

```
  scion-pki trc payload -t template.toml -o payload.der
  scion-pki trc payload -t template.toml -o payload.der -p predecessor.trc
```

Options

```
      --format string        Output format (der|pem) (default "der")
  -h, --help                 help for payload
  -o, --out string           Output file (required)
  -p, --predecessor string   Predecessor TRC
  -t, --template string      Template file (required)
```

SEE ALSO

- *scion-pki trc* - Manage TRCs for the SCION control plane PKI

scion-pki trc sign

Sign a TRC

Synopsis

'sign' signs a TRC payload with the signing key and signing certificate.

Voting, proof-of-possession, and root acknowledgement signatures can be added by using the corresponding signing keys and certificates.

By default, the resulting signed object is written to a file with the following naming pattern::

```
ISD<isd>-B<base_version>-S<serial_number>.<signing-isd_as>-<signature-type>.trc
```

An alternative name can be specified with the –out flag.

```
scion-pki trc sign <payload_file> <crt_file> <key_file> [flags]
```

Examples

```
  scion-pki trc sign ISD1-B1-S1.pld.der sensitive-voting.crt sensitive-voting.key
  scion-pki trc sign ISD1-B1-S1.pld.der regular-voting.crt regular-voting.key --out ISD1-
↪B1-S1.regular.trc
```

Options

```
  -h, --help             help for sign
  -o, --out string       Output file path. If --out is set, --out-dir is ignored.
      --out-dir string   Output directory. If --out is set, --out-dir is ignored.␣
↪(default ".")
```

SEE ALSO

- *scion-pki trc* - Manage TRCs for the SCION control plane PKI

scion-pki trc verify

Verify a TRC chain

Synopsis

'verify' verifies a TRC chain based on a trusted anchor point.

The anchor can either be a collection of trusted certificates bundled in a PEM file, or a trusted TRC. TRC update chains that start with a base TRC can be verified with either type of anchor. TRC update chains that start with a non-base TRC must have a TRC as anchor. With the optional flag –isd, the ID of the ISD for which the TRC claims to be the root of trust can be matched against an expected value.

```
scion-pki trc verify [flags]
```

Examples

```
  scion-pki trc verify --anchor bundle.pem ISD1-B1-S1.trc
  scion-pki trc verify --anchor ISD1-B1-S1.trc ISD1-B1-S2.trc ISD1-B1-S3.trc
```

Options

```
  -a, --anchor string   trust anchor (required)
  -h, --help            help for verify
      --isd uint16      ISD identifier
```

SEE ALSO

- *scion-pki trc* - Manage TRCs for the SCION control plane PKI

11.2.5 scion-pki version

Show the scion-pki version information

```
scion-pki version [flags]
```

Options

```
  -h, --help   help for version
```

SEE ALSO

- *scion-pki* - SCION Control Plane PKI Management Tool

CHAPTER

TWELVE

TELEMETRY

In setups where a SCION CA needs to be configured, HashiCorp Vault is a common tool that provides a secure implementation of a certificate authority and secret storage. To ensure that the Vault instances are operating correctly, there is a variety of Prometheus metrics exposed.

For further information on Prometheus metrics, please refer to the official Prometheus documentation and the explanation offered in *Overview*.

12.1 Metrics

Vault instances configured by Anapaya expose two categories of Prometheus metrics:

- **Built-in metrics as defined by HashiCorp Vault**

  For a list of all the metrics exposed by HashiCorp Vault, please refer to the official documentation. Metrics of high importance are the seal status of the Vault instance and the number of existing mount points. At the time of writing, the metric names for these features are `vault.core.unsealed` and `vault.core.mount_table.num_entries`.

- **Anapaya-defined metrics that relate to the SCION CA operations of the host**

  To ensure that the Vault instances are operating correctly as SCION CAs, Anapaya has defined some additional metrics that are exposed:

  vaultca_build_info

Description	vaultca build information
Type	gauge
Labels	`version`

  vaultca_certificate_matching_distinguished_name

Description	Indicate that the distinguished name in the certificate contains the expectedvalue for this engine.
Type	gauge
Labels	`isd_as` `type` `dn`

  vaultca_certificate_not_after_time

Description	The NotAfter time of the last certificate that was successfully updated in vault
Type	gauge
Labels	`isd_as` `type`

vaultca_certificate_not_before_time

Description	The NotBefore time of the last certificate that was successfully updated in vault
Type	gauge
Labels	`isd_as` `type`

vaultca_instance_id

Description	The instance identifier of the last CA certificate that was successfully updated in vault. This is the number of the instance in the CommonName of the CA certificate. E.g., for CA certificate with CommonName 'Anapaya CA - GEN I 2021.42', the instance identifier is 42. If the CA certificate does not have a instance identifier in the CommonName, the value is set to -1.
Type	gauge
Labels	`isd_as` `type`

vaultca_latest_trc_base_version

Description	The base version of the latest TRC stored in the secret engine for the given ISD.
Type	gauge
Labels	`isd_as`

vaultca_latest_trc_contains_root_certificate

Description	Indicates if the current root certificate is included in the latest TRC.
Type	gauge
Labels	`isd_as`

vaultca_latest_trc_not_after_time

Description	The NotAfter time of the latest TRC stored in the secret engine for the given ISD.
Type	gauge
Labels	`isd_as`

vaultca_latest_trc_not_before_time

Description	The NotBefore time of the latest TRC stored in the secret engine for the given ISD.
Type	gauge
Labels	`isd_as`

vaultca_latest_trc_serial_version

Description	The serial version of the latest TRC stored in the secret engine for the given ISD.
Type	gauge
Labels	`isd_as`

vaultca_update_task_enabled

Description	Indicates if the periodic vaultca update task is enabled.
Type	gauge
Labels	None

vaultca_update_task_errors_total

Description	The amount of errors in runs of the vaultca update periodic task.
Type	counter
Labels	None

vaultca_update_task_successes_total

Description	The amount of successful runs of the vaultca update periodic task.
Type	counter
Labels	None

12.2 Setting up Monitoring for Vault instances

Our monitoring stack is based on Prometheus, Grafana, Loki and AlertManager. They are all open-source tools with plenty of documentation and support online.

In order to set up monitoring of Anapaya appliances or Vault instances, there are a few technical requirements.

1. The monitoring host must be able to reach the management interface of the target appliances.
2. Firewall rules must allow opening an HTTP(S) connection on the monitoring port of the appliance.

12.2.1 Enabling Telemetry on the Host

If you are enabling telemetry in an instance of Vault, ensure that the Vault configuration (usually stored in `/etc/vault.d/vault.hcl`) contains the section:

```
telemetry {
  # retention time of metrics, only new metrics (updated < 1m ago) would be exposed
  prometheus_retention_time = "1m"
  disable_hostname = true
}
```

12.3 Setting up Prometheus

In order to set up Prometheus, follow the official Prometheus instructions. Specifically,

1. Ensure you have the latest version of Prometheus installed. Consult the installation guide for reference.
2. Follow the instructions in Section Configure Prometheus to monitor the sample targets.
   - To monitor each Vault instance, add the following target configuration in the `prometheus.yml` file. For each Vault instance that will be monitored, add an entry in the targets section. The vault listening address is the `api_addr` as set in the Vault configuration file (usually stored at `/etc/vault.d/vault.hcl`).

     ```
     - job_name: 'vault'
       honor_labels: true
       static_configs:
         - targets:
             - <vault listening address>
     ```

3. Start Prometheus. The exact command depends on the method of installation.

12.3.1 Recording and Alerting rules

Prometheus allows the configuration of rules for recording data or creating alerts when an event happens. These alerts can later be picked up by AlertManager and be integrated with your alerting system. You can specify the events that trigger an alert, the scope and severity of the alert, and also provide a description and summary of the firing alert. Below, we provide an example of how to monitor the seal state of the Vault instance.

```
- alert: InfraVaultSealed
  expr: vault_core_unsealed == 0
  for: 1m
```

12.4 Monitoring Stack

We suggest you set up Grafana, and Alertmanager to visualize metrics and keep track of alerts. For instructions on how to achieve this, please refer to *Setting up Grafana* and *Setting up AlertManager*.

CHAPTER

THIRTEEN

TROUBLESHOOTING & RUNBOOKS

This documentation page contains runbooks for the alerts sent out by the Anapaya CA product as well as common operations that are helpful when troubleshooting.

13.1 Alerts

13.1.1 InfraVaultMountPointsLimit

This alert fires if Vault is reaching the limit of mount points it can handle.

According to the official documentation, Vault can handle up to ~14000 mount points. Under normal conditions, the amount of mount points should not be remotely close to the limit.

Actions

1. Check the current amount of mounts via the /sys/mounts endpoint.
2. Disable the old PKI engines that are no longer needed for auditing purposes following the instructions in the documentation.

13.1.2 InfraVaultSealed

This alert fires if the Vault instance is sealed. This can happen for example after a reboot of the machine running Vault.

Vault stores all secrets encrypted on disk. When the Vault instance is sealed, it does not have access to the decryption key and therefore cannot access the stored secrets. For a Vault instance to operate as expected, it needs to be unsealed.

Actions

1. Follow the official documentation on how to unseal Vault via the CLI tool or the HTTP API.

13.1.3 SCIONVaultCACertificateExpiring

This alert fires if the Vault CA Backend for the ISD-AS in question has a CA certificate provisioned that will expire in less than 4 days. If the remaining CA certificate validity falls below 3 days, AS certificate chain renewal will not be possible anymore.

Actions

Temporary fix

- You can temporarily lower the validity period of the issued AS certificates. This permits issuance to continue, even if the remaining CA certificate validity falls below the minimum of 3 days. To do so, reduce the default TTL in the `ca/{{ isd_as }}/pki` engine to a value smaller than 3 days, where `isd_as` is the value of the ISD-AS number for which the certificate did not renew. This can be achieved through the CLI tool or the HTTP API.

Warning: This action does not solve the source of the problem issue. It only enables AS certificates to be renewed and have a validity time that is smaller than the default value of 3 days. Once the CA certificate expires, AS certificate renewal will also stop.

Warning: This action must be manually reverted as soon as there is a new CA certificate, otherwise AS certificates will be renewed with validity periods below the recommended 3 days.

Troubleshooting

1. Check if the *vaultca service is running*.
2. If the `vaultca service` is running, *check if the logs* have any errors for the CA certificate renewal.
3. If the vaultca service is not running, *start the vaultca service* and *check the logs* for the details of the renewed CA certificate or errors.
4. *Manually trigger the CA certificate update* for the relevant ISD-AS(es).

13.1.4 SCIONVaultCACertificateInFuture

This alert fires if the Vault CA Backend for the ISD-AS in question has a CA certificate provisioned with the `NotBefore` time in the future. This means that no AS certificate chains can be issued for the current moment.

This issue usually occurs when clock synchronization is off.

Actions

1. Check if the time synchronization of the clocks is working as expected. The specific commands to use differ depending on the operating system and the setup.
2. If time synchronization is off, synchronize the clock on the device.

13.1.5 SCIONVaultCertificateDestinguishedNameMismatch

This alert fires if the Vault CA Backend for the ISD-AS in question has a certificate provisioned where the distinguished name of the issuer or the subject belongs to a different ISD-AS. This should never happen in practice. If the alert fires, something has gone really wrong in the Vault setup and renewal process.

Actions

1. This alert should never be triggered. Contact Anapaya Support.

13.1.6 SCIONVaultCANotRolledOver

This alert fires if the Vault CA Backend for the Vault instance has not been rolled over within the last 7 days.

In general, Vault instances have a periodic task that renews the CA certificate and rolls over the CA PKI engine every week. This alert shows that there was a problem either with renewing the CA certificate or with rolling over the PKI engine to the newest one.

Note: As soon as the CA certificate expiration date is less than 3 days in the future, the automatic AS certificate renewal will stop working.

Actions

1. Check if the *vaultca service is running*.
2. If the `vaultca service` is running, *check if the logs* have any errors for the CA rollover process.
3. If the `vaultca service` is not running, *start the vaultca service* and *check the logs* for the details of the renewed CA certificate, the CA rollover or errors.
4. *Manually trigger the CA certificate update* for the relevant ISD-AS(es).

13.1.7 SCIONVaultCARenewalNotOperating

This alert fires if the automatic CA certificate renewal, run by the `vaultca service`, is not operating.

Actions

1. Check if the *vaultca service is running*.
2. If the `vaultca service` is running, *check if the logs* have any errors for the CA rollover process.
3. If the `vaultca service` is not running, *start the vaultca service* and *check the logs* for the details of the renewed CA certificate, the CA rollover or errors.

13.1.8 SCIONVaultRootCertificateExpiring

This alert fires if the Vault CA Backend for the ISD-AS in question has a root certificate provisioned that expires in the next 45 days. If the root certificate expires, then no CA certificate chains and eventually no AS certificate chains will be able to be issued.

There are two cases where this alert can be triggered:

1. The TRC for the ISD was updated, but the root engine for the ISD-AS in question was not re-provisioned.
2. The TRC is also expiring, along with the root certificate for the ISD-AS in question.

Actions

1. If the TRC was recently updated but the root engine was not updated, then *re-provision the root engine*.
2. If the TRC is also expiring, start the provisioning process for a new TRC. Initiate the TRC update process with a quorum of voting members. The details of this process depend on the governance rules of your respective ISD.

13.1.9 SCIONVaultRootCertificateInFuture

This alert fires if the Vault CA Backend for the ISD-AS in question has a root certificate provisioned with the `NotBefore` time in the future. This means that no CA certificate chains can be issued at the current moment.

This issue usually occurs when clock synchronization is off.

Actions

1. Check if the time synchronization of the clocks is working as expected. The specific commands to use differ depending on the operating system and the setup.
2. If time synchronization is off, synchronize the clock on the device.

13.1.10 SCIONVaultTRCExpiring

This alert fires if the TRC stored in the Vault CA Backend for the ISD-AS in question is expiring within 30 days. If the TRC expires, then the ISD-AS will not be able to validate information related to the path discovery process.

There are two cases in which this alert can be triggered:

1. The TRC was recently updated, but it was not pushed to the secret engine storage for the ISD-AS in question.
2. The TRC has not been recently updated and will expire soon.

Actions

1. If the TRC was recently updated but not pushed to the secret engine, *push the new TRC*.
2. If the TRC is expiring and there is no new TRC available, follow this process:
    1. *Inspect the TRC* to verify that it expires soon.
    2. Initiate the TRC update process with a quorum of voting members. The details of this process depend on the governance rules of your respective ISD.

13.1.11 SCIONVaultTRCInFuture

This alert fires if the TRC stored in the Vault CA Backend for the ISD-AS in question is provisioned with a `Not_Before` value in the future.

This can occur in two cases:

1. The latest TRC is pushed to the secrets engine before the agreed time with the other voting parties for the TRC.
2. Clock synchronization is off.

Actions

1. If you pushed the TRC before the agreed time, you need to *push the previous TRC* to engine storage for the ISD-AS in question.
2. If time synchronization is off, synchronize the clock on the device. The specific commands to use differ depending on the operating system and the setup.

13.2 Common Operations

13.2.1 Start the vaultca service

1. SSH to the given machine.
2. Run the following command:

```
systemctl start vaultca.service
```

3. Ensure that there are no errors in the output and that the status of the service is `active (running)`.

13.2.2 Check whether vaultca service is running

1. SSH to the given machine.
2. Run the following command:

```
systemctl status vaultca.service
```

3. The output will include the status of the service, which should be `active (running)`.

13.2.3 Inspect vaultca service logs

1. SSH to the given machine.
2. Run the following command:

```
journalctl -u vaultca.service
```

Tip: To see only the recent logs use:

```
journalctl -u vaultca.service --since=<time-duration>
```

For example, to check the logs of the last minute, run

```
journalctl -u vaultca.service --since=1m
```

Tip: The logs are also exposed to Loki and can be viewed with Grafana.

13.2.4 Manually renew CA Certificates

1. SSH to the given machine.
2. Run the following command. By default, the configuration file of the `vaultca` service is located at `/etc/vaultca/config.toml`. If the configuration file is located somewhere else, make sure to pass the correct path as input to the `--config` flag.

```
vaultca renew --config=/etc/vaultca/config.toml
```

Note: This will update the CA Certificates for all the ISD-ASes configured on the host. To update the CA Certificate for a specific ISD-AS, you can pass the `--isd-as` flag as an input.

For example, to renew the CA Certificate for ISD-AS `1-ff00:0:1`, you can run:

```
vaultca renew --config=/etc/vaultca/config.toml --isd-as=1-ff00:0:1
```

13.2.5 Inspect TRC

1. SSH to the machine that runs the Vault instance.
2. Get the latest version of the TRC using the `vault kv get` command and following the official Vault documentation. The latest TRC is stored under the path `ca/<isd-as>/kv/trcs/latest`.
3. Use the *scion-pki tool* to inspect the TRC details.

13.2.6 Push new TRC

1. SSH to the machine that runs the Vault instance.
2. Push the new version of the TRC by following the official Vault documentation. The latest TRC is stored under the path `ca/<isd-as>/kv/trcs/latest`.

13.2.7 Provision new Root CA engine

1. SSH to the machine that runs the Vault instance.
2. Ensure that the Root PKI engine is enabled by following the official Vault documentation. Add the following fields in the JSON-formatted body of the request:
   - The path for the PKI engine is `root/<ISD-AS>/pki`.
   - The type should be marked as `pki`.
   - You can add an optional description such as `Root PKI for <ISD-AS>`.
3. Tune the Root PKI engine with default and max lease TTL. Follow the official Vault documentation and configure the `default_lease_ttl` to be `264h` and the `max_lease_ttl` to be `720h`.
4. Generate a new root key and certificate bundle.
5. Configure Root PKI with the root key and root certificate bundle. Follow the official Vault documentation and use the `root/<ISD-AS>/pki/config/ca` path. Include as `pem_bundle` the root key and certificate generated in the previous step.

CHAPTER

FOURTEEN

ANAPAYA EDGE

The Anapaya EDGE can be deployed in various deployment setups. Depending on the requirements different deployment architectures should be used. In general, more instances increase redundancy and thus provide more reliability. In the following, we present the most common deployment architectures.

14.1 Single EDGE setup

The single EDGE setup consists of a single Appliance configured as an EDGE. Note that with this setup, there will be downtime when the Appliance is being upgraded or fails.

The diagram shows an EDGE appliance with three network interfaces. The WAN interface has the IP address `169.254.10.2/30` and directly connects to the ISP which provides it with SCION connectivity. The SCION link to the ISP has the interface identifier `1`. On the LAN side the EDGE has an interface with the IP address `10.10.0.2/24`, additionally it has a default gateway on the LAN interface towards `10.10.0.1`. The LAN interface should connect to the internal network of the organization which wants to use a SCION connection. The services which should be reachable by other participants of the SCION Internet can be placed anywhere inside the LAN, but must be reachable from the EDGE appliance. Lastly, the EDGE appliance is configured with a management interface with the IP address `192.168.1.2/24`.

In order to send traffic from the LAN into the SCION network, the LAN needs to be configured with the LAN interface IP (`10.10.0.2`) of the EDGE appliance as a next hop.

Full Configuration

```
{
    "config": {
        "interfaces": {
            "ethernets": [
                {
                    "addresses": [
                        "10.10.0.2/24"
                    ],
                    "name": "lan",
                    "gateway": {
                        "ipv4_gateway": "10.10.0.1"
                    }
                },
                {
                    "addresses": [
```

(continues on next page)

(continued from previous page)

```
                    "169.254.10.2/30"
                ],
                "name": "wan"
            },
            {
                "addresses": [
                    "192.168.1.2/24"
                ],
                "name": "mgmt",
                "driver": "LINUX"
            }
        ]
    },
    "management": {
        "api": {
            "address": "192.168.1.2:42000"
        },
        "hostname": "host.site.org",
        "telemetry": {
            "address": "192.168.1.2:42001"
        }
    },
    "scion": {
        "ases": [
            {
                "control": {
                    "address": "10.10.0.2:40001",
                    "enabled": true
                },
                "cppki": {
                    "issuers": [
                        {
                            "isd_as": "1-ff00:0:24",
                            "priority": 0
                        }
                    ]
                },
                "details": {
                    "name": "Example AS Name"
                },
                "forwarding_key": "PLACEHOLDER_FW_KEY",
                "isd_as": "1-ff00:0:14",
                "router": {
                    "enabled": true,
                    "internal_interface": "10.10.0.2:30100"
                },
                "scion_mtu": 1472,
                "neighbors": [
                    {
                        "neighbor_isd_as": "1-ff00:0:24",
                        "relationship": "PARENT",
                        "interfaces": [
```

(continues on next page)

(continued from previous page)

```
                            {
                                "address": "169.254.10.2:30100",
                                "administrative_state": "UP",
                                "interface_id": 1,
                                "remote": {
                                    "address": "169.254.10.1:30100",
                                    "interface_id": 10
                                },
                                "scion_mtu": 1472
                            }
                        ]
                    }
                ]
            }
        ]
    },
    "scion_tunneling": {
        "endpoint": {
            "control_port": 40201,
            "data_port": 40200,
            "enabled": true,
            "ip": "10.10.0.2",
            "probe_port": 40202
        },
        "domains": [
            {
                "local_isd_ases": [
                    "1-ff00:0:14"
                ],
                "name": "default-domain",
                "prefixes": {
                    "accept_filter": [
                        {
                            "action": "ACCEPT",
                            "prefixes": [
                                "10.20.0.0/24"
                            ],
                            "sequence_id": 0
                        }
                    ],
                    "announce_filter": [
                        {
                            "action": "ACCEPT",
                            "prefixes": [
                                "10.10.0.15/32"
                            ],
                            "sequence_id": 0
                        }
                    ]
                },
                "remote_isd_ases": [
                    {
```

(continues on next page)

(continued from previous page)

```
                        "action": "ACCEPT",
                        "isd_as": "1-ff00:0:25",
                        "sequence_id": 0
                    }
                ],
                "traffic_policies": [
                    {
                        "path_policy": "allow-all",
                        "sequence_id": 0,
                        "traffic_matcher": "default"
                    }
                ]
            }
        ],
        "path_policies": [
            {
                "name": "allow-all",
                "path_filters": [
                    {
                        "acl": [
                            "+"
                        ],
                        "sequence_id": 0
                    }
                ]
            }
        ],
        "remotes": [
            {
                "isd_as": "1-ff00:0:25"
            }
        ],
        "static_announcements": [
            {
                "prefixes": [
                    "10.10.0.15/24"
                ],
                "sequence_id": 0
            }
        ],
        "traffic_matchers": [
            {
                "condition": "BOOL=true",
                "name": "default"
            }
        ]
    },
    "system": {
        "ntp": {
            "servers": [
                {
                    "address": "time.example.org"
```

(continues on next page)

(continued from previous page)

```
                    }
                ]
            }
        }
    }
}
```

14.2 Redundant EDGE Appliances

Two or more appliances provide more resilience against failures and allow for downtime-free upgrades of the appliances. This is the recommended architecture for all EDGE deployments. Within the redundant appliance setups, there are multiple viable setups depending on the organization requirements.

We recommend using at least two connected appliances with either static or dynamic redundancy. If a connected setup is not possible, i.e. it is not possible or desired for the EDGE appliances to communicate, it is recommended to use multiple *Single EDGE setup* setups, each being its own ISD-AS. This applies for example if the two EDGE appliances are placed in different physical sites.

The advantage of connected appliances is that it allows both appliances to take advantage of the provider uplinks of both appliances. In this setup, the appliances automatically share path information between each other and thus enable provider uplink redundancy.

14.2.1 Static redundancy (VRRP)

Static redundancy is the easiest way to achieve appliance and provider uplink redundancy. In this setup the appliances form a cluster using the Virtual Router Redundancy Protocol (VRRP) and are configured with a virtual IP (VIP) which is shared between the appliances. The VIP is configured as the next hop on the LAN side for all traffic that should be sent via the SCION network.

Static redundancy is recommended when no dynamic routing protocols are available in the part of the organization's network where the EDGE appliances are placed. It is required that the EDGE appliances are placed within a layer 2 network on the LAN side, such that they can form a VRRP cluster.

Note: In the static redundancy setup, only one EDGE appliance will be used for outgoing traffic, the second EDGE appliance acts as a backup. Incoming traffic from the SCION network may arrive on both EDGE appliances and is not tied to the state of the VIP.

The diagram shows two EDGE appliances that are configured very similarly to the *Single EDGE setup* setup. On the LAN side, the two EDGE appliances are connected using a Layer 2 network, such that they can form a VRRP cluster.

In order to send traffic from the LAN into the SCION network, the LAN needs to be configured with the VIP (`10.10.0.5`) as a next hop.

LAN Ethernet Interface

```
"ethernets": [
    {
```

(continues on next page)

(continued from previous page)

```
        "addresses": [
            "10.10.0.2/24"
        ],
        "name": "lan",
        "gateway": {
            "ipv4_gateway": "10.10.0.1"
        },
        "vrrp": [
            {
                "addresses": [
                    "10.10.0.5"
                ],
                "vrid": 1
            }
        ]
    }
]
```

Full Configuration

```
{
    "config": {
        "cluster": {
            "peers": [
                {
                    "name": "host2.site.org",
                    "scion": {
                        "ases": [
                            {
                                "control": {
                                    "address": "10.10.1.1:40001"
                                },
                                "isd_as": "1-ff00:0:14",
                                "shard_id": 2,
                                "neighbors": [
                                    {
                                        "neighbor_isd_as": "1-ff00:0:33",
                                        "interfaces": [
                                            {
                                                "interface_id": 2,
                                                "next_hop": "10.10.1.1:31000"
                                            }
                                        ]
                                    }
                                ]
                            }
                        ]
                    },
                    "scion_tunneling": {
                        "endpoint": {
```

(continues on next page)

(continued from previous page)

```
                        "control_port": 40201,
                        "data_port": 40200,
                        "ip": "10.10.1.1",
                        "probe_port": 40202
                    }
                }
            }
        ]
    },
    "interfaces": {
        "ethernets": [
            {
                "addresses": [
                    "10.10.0.2/24"
                ],
                "name": "lan",
                "gateway": {
                    "ipv4_gateway": "10.10.0.1"
                },
                "vrrp": [
                    {
                        "addresses": [
                            "10.10.0.5"
                        ],
                        "vrid": 1
                    }
                ]
            },
            {
                "addresses": [
                    "169.254.10.2/30"
                ],
                "name": "wan"
            },
            {
                "addresses": [
                    "192.168.1.2/24"
                ],
                "name": "mgmt",
                "driver": "LINUX"
            }
        ]
    },
    "management": {
        "api": {
            "address": "192.168.1.2:42000"
        },
        "hostname": "host1.site.org",
        "telemetry": {
            "address": "192.168.1.2:42001"
        }
    },
```

(continues on next page)

(continued from previous page)

```
        "scion": {
            "ases": [
                {
                    "control": {
                        "address": "10.10.0.2:40001",
                        "enabled": true
                    },
                    "cppki": {
                        "issuers": [
                            {
                                "isd_as": "1-ff00:0:24",
                                "priority": 0
                            }
                        ]
                    },
                    "details": {
                        "name": "Example AS Name"
                    },
                    "forwarding_key": "PLACEHOLDER_FW_KEY",
                    "isd_as": "1-ff00:0:14",
                    "router": {
                        "enabled": true,
                        "internal_interface": "10.10.0.2:30100"
                    },
                    "scion_mtu": 1472,
                    "shard_id": 1,
                    "neighbors": [
                        {
                            "neighbor_isd_as": "1-ff00:0:24",
                            "relationship": "PARENT",
                            "interfaces": [
                                {
                                    "address": "169.254.10.2:30100",
                                    "administrative_state": "UP",
                                    "interface_id": 1,
                                    "remote": {
                                        "address": "169.254.10.1:30100",
                                        "interface_id": 10
                                    },
                                    "scion_mtu": 1472
                                }
                            ]
                        }
                    ]
                }
            ]
        },
        "scion_tunneling": {
            "endpoint": {
                "control_port": 40201,
                "data_port": 40200,
                "enabled": true,
```

(continues on next page)

(continued from previous page)

```
            "ip": "10.10.0.2",
            "probe_port": 40202
        },
        "domains": [
            {
                "local_isd_ases": [
                    "1-ff00:0:14"
                ],
                "name": "default-domain",
                "prefixes": {
                    "accept_filter": [
                        {
                            "action": "ACCEPT",
                            "prefixes": [
                                "10.20.0.0/24"
                            ],
                            "sequence_id": 0
                        }
                    ],
                    "announce_filter": [
                        {
                            "action": "ACCEPT",
                            "prefixes": [
                                "10.10.0.15/32"
                            ],
                            "sequence_id": 0
                        }
                    ]
                },
                "remote_isd_ases": [
                    {
                        "action": "ACCEPT",
                        "isd_as": "1-ff00:0:25",
                        "sequence_id": 0
                    }
                ],
                "traffic_policies": [
                    {
                        "path_policy": "allow-all",
                        "sequence_id": 0,
                        "traffic_matcher": "default"
                    }
                ]
            }
        ],
        "path_policies": [
            {
                "name": "allow-all",
                "path_filters": [
                    {
                        "acl": [
                            "+"
```

(continues on next page)

(continued from previous page)

```
                        ],
                        "sequence_id": 0
                    }
                ]
            }
        ],
        "remotes": [
            {
                "isd_as": "1-ff00:0:25"
            }
        ],
        "static_announcements": [
            {
                "prefixes": [
                    "10.10.0.15/24"
                ],
                "sequence_id": 0
            }
        ],
        "traffic_matchers": [
            {
                "condition": "BOOL=true",
                "name": "default"
            }
        ]
    },
    "system": {
        "ntp": {
            "servers": [
                {
                    "address": "time.example.org"
                }
            ]
        }
    }
}
```

14.2.2 Dynamic Redundancy (BGP)

Dynamic redundancy is slightly more complicated to set up than static redundancy but also offers more flexibility for the user. In this setup, each EDGE appliance is configured with a BGP session towards a router on the LAN side. IP prefixes that are learned from the SCION network are advertised to the LAN via BGP, conversely, IP prefixes that are reachable in the LAN need to be advertised to the EDGE appliances via BGP as well. When an IP prefix is no longer reachable via the SCION network, it is retracted from BGP and the LAN will no longer use that EDGE appliance to reach the SCION network.

Note: In the dynamic redundancy setup both EDGE appliances can be used for outgoing traffic simultaneously, this can be influenced by the operator of the LAN side BGP router.

The diagram shows two EDGE appliances that are configured very similarly to the *Single EDGE setup* setup. In this setup, instead of directly connecting to the LAN, the EDGE appliances are configured to establish a BGP session with their counterparts on the LAN side (`10.10.0.2` and `10.10.1.2`). The BGP sessions are configured using eBGP and can use private ASNs.

The EDGE appliances advertise the IP prefixes which they learn via the SCION network to their BGP peer towards the LAN. Like this, it's the duty of the BGP peers to route traffic from the LAN towards the EDGE appliances for sending it via the SCION network.

BGP

```
"bgp": {
    "global": {
        "as": 64545,
        "router_id": "10.10.0.1"
    },
    "neighbors": [
        {
            "auth_password": "PLACEHOLDER_AUTH_PASS",
            "enabled": true,
            "local_as": 64545,
            "neighbor_address": "10.10.0.2",
            "peer_as": 64546
        }
    ]
}
```

Full Configuration

```
{
    "config": {
        "bgp": {
            "global": {
                "as": 64545,
                "router_id": "10.10.0.1"
            },
            "neighbors": [
                {
                    "auth_password": "PLACEHOLDER_AUTH_PASS",
                    "enabled": true,
                    "local_as": 64545,
                    "neighbor_address": "10.10.0.2",
                    "peer_as": 64546
                }
            ]
        },
        "cluster": {
            "peers": [
                {
                    "name": "host2.site.org",
```

(continues on next page)

(continued from previous page)

```
                "scion": {
                    "ases": [
                        {
                            "control": {
                                "address": "10.10.1.1:40001"
                            },
                            "isd_as": "1-ff00:0:14",
                            "shard_id": 2,
                            "neighbors": [
                                {
                                    "neighbor_isd_as": "1-ff00:0:33",
                                    "interfaces": [
                                        {
                                            "interface_id": 2,
                                            "next_hop": "10.10.1.1:31000"
                                        }
                                    ]
                                }
                            ]
                        }
                    ]
                },
                "scion_tunneling": {
                    "endpoint": {
                        "control_port": 40201,
                        "data_port": 40200,
                        "ip": "10.10.1.1",
                        "probe_port": 40202
                    }
                }
            }
        ]
    },
    "interfaces": {
        "ethernets": [
            {
                "addresses": [
                    "10.10.0.2/24"
                ],
                "name": "lan",
                "gateway": {
                    "ipv4_gateway": "10.10.0.1"
                }
            },
            {
                "addresses": [
                    "169.254.10.2/30"
                ],
                "name": "wan"
            },
            {
                "addresses": [
```

(continues on next page)

(continued from previous page)

```
                    "192.168.1.2/24"
                ],
                "name": "mgmt",
                "driver": "LINUX"
            }
        ]
    },
    "management": {
        "api": {
            "address": "192.168.1.2:42000"
        },
        "hostname": "host1.site.org",
        "telemetry": {
            "address": "192.168.1.2:42001"
        }
    },
    "scion": {
        "ases": [
            {
                "control": {
                    "address": "10.10.0.2:40001",
                    "enabled": true
                },
                "cppki": {
                    "issuers": [
                        {
                            "isd_as": "1-ff00:0:24",
                            "priority": 0
                        }
                    ]
                },
                "details": {
                    "name": "Example AS Name"
                },
                "forwarding_key": "PLACEHOLDER_FW_KEY",
                "isd_as": "1-ff00:0:14",
                "router": {
                    "enabled": true,
                    "internal_interface": "10.10.0.2:30100"
                },
                "scion_mtu": 1472,
                "shard_id": 1,
                "neighbors": [
                    {
                        "neighbor_isd_as": "1-ff00:0:24",
                        "relationship": "PARENT",
                        "interfaces": [
                            {
                                "address": "169.254.10.2:30100",
                                "administrative_state": "UP",
                                "interface_id": 1,
                                "remote": {
```

(continues on next page)

(continued from previous page)

```
                                    "address": "169.254.10.1:30100",
                                    "interface_id": 10
                                },
                                "scion_mtu": 1472
                            }
                        ]
                    }
                ]
            }
        ]
    },
    "scion_tunneling": {
        "endpoint": {
            "control_port": 40201,
            "data_port": 40200,
            "enabled": true,
            "ip": "10.10.0.2",
            "probe_port": 40202
        },
        "domains": [
            {
                "local_isd_ases": [
                    "1-ff00:0:14"
                ],
                "name": "default-domain",
                "prefixes": {
                    "accept_filter": [
                        {
                            "action": "ACCEPT",
                            "prefixes": [
                                "10.20.0.0/24"
                            ],
                            "sequence_id": 0
                        }
                    ],
                    "announce_filter": [
                        {
                            "action": "ACCEPT",
                            "prefixes": [
                                "10.10.0.15/32"
                            ],
                            "sequence_id": 0
                        }
                    ]
                },
                "remote_isd_ases": [
                    {
                        "action": "ACCEPT",
                        "isd_as": "1-ff00:0:25",
                        "sequence_id": 0
                    }
                ],
```

(continues on next page)

(continued from previous page)

```
                "traffic_policies": [
                    {
                        "path_policy": "allow-all",
                        "sequence_id": 0,
                        "traffic_matcher": "default"
                    }
                ]
            }
        ],
        "path_policies": [
            {
                "name": "allow-all",
                "path_filters": [
                    {
                        "acl": [
                            "+"
                        ],
                        "sequence_id": 0
                    }
                ]
            }
        ],
        "remotes": [
            {
                "isd_as": "1-ff00:0:25"
            }
        ],
        "static_announcements": [
            {
                "prefixes": [
                    "10.10.0.15/24"
                ],
                "sequence_id": 0
            }
        ],
        "traffic_matchers": [
            {
                "condition": "BOOL=true",
                "name": "default"
            }
        ]
    },
    "system": {
        "ntp": {
            "servers": [
                {
                    "address": "time.example.org"
                }
            ]
        }
    }
}
```

(continues on next page)

(continued from previous page)

```
}
```

CHAPTER

FIFTEEN

ANAPAYA CORE

The Anapaya CORE implements a SCION core router and is typically deployed by ISPs to provide connectivity to other SCION ISPs and downstream customers.

In the following, we describe a typical small deployment scenario, where an ISP (ISP 1) operates two CORE appliances, both connected to a peering SCION ISP. Furthermore, the ISP has two downstream customers that are connected to each of the two CORE appliances.

This redundant access design is a best practice because it increases the redundancy of the customer access and allows the ISP to perform maintenance on one of the CORE appliances without causing a service interruption.

15.1 Topology

Our target topology contains the following elements:

- ISP 1 deploys two CORE appliances, `s01.chzrh1.isp1` and `s01.chbrn1.isp1`, that are connected internally via the internal network `10.0.0.0/24`.
- ISP 1 is peering with ISP 2 (ISD-AS 1-ff00:2:1) at `s01.chzrh1.isp1`. The local SCION interface ID is `10` and the remote SCION interface ID is `20`. The underlay IPv6 network used to establish the link is `fe80:1::/64`, where `fe80:1::1/64` is the local IPv6 address and `fe80:1::2/64` is the remote IPv6 address.
- ISP 1 is peering with ISP 3 (ISD-AS 3-ff00:3:1) at `s01.chbrn1.isp1`. The local SCION interface ID is `20` and the remote SCION interface ID is `5`. The underlay IPv6 network used to establish the link is `fe80:2::/64`, where `fe80:2::1/64` is the local IPv6 address and `fe80:2::/64` is the remote IPv6 address.
- **ISP 1 is connecting Customer 1 (ISD-AS 1-ff00:1:10) redundantly:**
  - `s01.chzrh1.isp1`: The local SCION interface ID is `100` and the remote SCION interface ID is `1`. The underlay IPv4 network used to establish the link is `169.254.0.0/30`, where `.1` is the local IPv4 address and `.2` is the remote IPv4 address.
  - **`s01.chbrn1.isp1`: The local SCION interface ID is `201` and the** remote SCION interface ID is `2`. The underlay IPv4 network used to establish the link is `169.254.2.0/30`, where `.1` is the local IPv4 address and `.2` is the remote IPv4 address.
- **ISP 1 is connecting Customer 2 (ISD-AS 1-ff00:1:20) redundantly:**
  - `s01.chzrh1.isp1`: The local SCION interface ID is `101` and the remote SCION interface ID is `2`. The underlay IPv4 network used to establish the link is `169.254.3.0/30`, where `.1` is the local IPv4 address and `.2` is the remote IPv4 address.

- **`s01.chbrn1.isp1`: The local SCION interface ID is `200` and the** remote SCION interface ID is `1`. The underlay IPv4 network used to establish the link is `169.254.1.0/30`, where `.1` is the local IPv4 address and `.2` is the remote IPv4 address.

We will now configure the CORE appliances step-by-step to implement the above scenario.

15.2 Network Interface Configuration

First, we will look at the configuration for the network interfaces. In the setup, there are three physical network interfaces - one for the internal network `lan`, one for the peering link to the neighboring SCION ISP `ens1`, and one for the links to the downstream customer `ens2`. The links to the downstream customers are configured as VLANs on the `ens2` interface.

Note: It is not required that each link is on a separate physical interface. It is also possible to have multiple links on the same physical interface in the same or different VLANs, or multiple virtual functions on top of a single physical interface if the underlying hardware supports it.

In this example, we also configure the `lan` interface to be the default route.

Network Interface Configuration

s01.chzrh1.isp1

```
"interfaces": {
  "ethernets": [
    {
      "addresses": ["10.0.0.100/24"],
      "name": "lan",
      "gateway": {
        "ipv4_gateway": "10.0.0.1"
      }
    },
    {
      "addresses": ["fe80:1::1/64"],
      "name": "ens1"
    },
    {
      "name": "ens2"
    }
  ],
  "vlans": [
    {
      "name": "ens2.100",
      "addresses": ["169.254.0.1/30"],
      "link": "ens2",
      "id": 100
    },
    {
      "name": "ens2.200",
      "addresses": ["169.254.3.1/30"],
      "link": "ens2",
```

(continues on next page)

(continued from previous page)

```
      "id": 200
    }
  ]
},
```

s01.chbrn1.isp1

```
"interfaces": {
  "ethernets": [
    {
      "addresses": ["10.0.0.200/24"],
      "name": "lan",
      "gateway": {
        "ipv4_gateway": "10.0.0.1"
      }
    },
    {
      "addresses": ["fe80:2::1/64"],
      "name": "ens1"
    },
    {
      "name": "ens2"
    }
  ],
  "vlans": [
    {
      "name": "ens2.100",
      "addresses": ["169.254.1.1/30"],
      "link": "ens2",
      "id": 100
    },
    {
      "name": "ens2.200",
      "addresses": ["169.254.2.1/30"],
      "link": "ens2",
      "id": 200
    }
  ]
},
```

Please refer to *Network Interfaces* for the full documentation on network interface configuration.

15.3 SCION Configuration

Next, we will look at the SCION configuration. The SCION section contains the configuration of the SCION protocol and AS and can be grouped into the *general AS*, the *control plane*, and the *data plane* configuration.

15.3.1 General AS Configuration

Each SCION AS has several general AS configuration options such as the ISD-AS identifier, the AS forwarding secret key, or a human-readable description of the AS. For the full list of the general AS configuration options, please refer to *General AS Configuration*.

In this example, we configure the following fields:

isd_as
: The ISD-AS identifier of the AS. In this example, the ISD-AS identifier is `1-ff00:0:1`. This must be the same for all CORE appliances.

forwarding_key
: The base64 encoded secret key used to verify the integrity and authenticity of the SCION packet header during forwarding on the data plane. This must be the same for all CORE appliances.

> **Note:** In the example, we use a dummy key. In production, the secret key should be a strong password that is at least 12 characters long and contain uppercase letters, lowercase letters, numbers, and special characters.

core
: The `core` flag indicates whether the AS is a core AS. In this example, ISP 1 is a core AS of ISD 1 and therefore sets this flag to `true`.

scion_mtu
: The SCION MTU is the maximum size of a SCION packet supported in the internal network. Since we are using an IPv4 underlay network, we set this to 1472 bytes. This assumes an Ethernet frame size of 1500 bytes, minus 20 bytes for the IPv4 header and 8 bytes for the UDP header that is used to transport SCION packets. This must be the same for all CORE appliances.

shard_id
: CORE appliances are typically deployed in a sharded manner, where each shard can function independently. The `shard_id` is used to identify the shard. The `shard_id` must be unique for each CORE appliance in a given AS, as each constitutes its own shard.

General AS Configuration

s01.chzrh1.isp1

```
"scion": {
  "ases": [
    {
      "isd_as": "1-ff00:0:1",
      "forwarding_key": "Zm9yd2FyZGluZ19rZXk=",
      "core": true,
      "scion_mtu": 1472,
      "shard_id": 1,
```

s01.chbrn1.isp1

```
"scion": {
  "ases": [
    {
      "isd_as": "1-ff00:0:1",
      "forwarding_key": "Zm9yd2FyZGluZ19rZXk=",
      "core": true,
      "scion_mtu": 1472,
      "shard_id": 2,
```

15.3.2 Control Plane Configuration

The SCION control plane configuration consists of the `control` and `cppki` sections that configure the network endpoint of the control service and the AS certificate renewal. The full documentation on the SCION control plane configuration can be found in *Control Plane Configuration*.

The network endpoint of the control service is typically configured to listen on the internal interface. The port can be set to `0` to let the appliance automatically allocate a free port.

The `cppki` section configures the SCION AS certificate issuers. These are required to periodically renew the AS certificate on the CORE appliances. In this example, we assume that there are two certificate issuers for ISD 1 at `1-ff00:0:100` and `1-ff00:0:200`. We want to configure one as the primary issuer and the other as the fallback in case the primary issuer is not reachable.

SCION Control Plane Configuration

s01.chzrh1.isp1

```
"control": {
  "address": "10.0.0.100:0",
  "enabled": true
},
"cppki": {
  "issuers": [
    {
      "isd_as": "1-ff00:0:100",
      "priority": 0
    },
    {
      "isd_as": "1-ff00:0:200",
      "priority": 1
    }
  ]
},
```

s01.chbrn1.isp1

```
"control": {
  "address": "10.0.0.200:0",
  "enabled": true
},
"cppki": {
  "issuers": [
```

(continues on next page)

(continued from previous page)

```
    {
      "isd_as": "1-ff00:0:100",
      "priority": 0
    },
    {
      "isd_as": "1-ff00:0:200",
      "priority": 1
    }
  ]
},
```

15.3.3 SCION Data Plane Configuration

The SCION data plane configuration consists of the `router` and the `neighbors` sections. The `router` section contains the configuration of the *internal* SCION interface `internal_interface`, while the `neighbors` section defines all the SCION links to other SCION ASes, i.e., the *external* SCION interfaces. The full documentation on the SCION data plane configuration can be found in *Data Plane Configuration*.

The internal SCION interface is where the appliance receives SCION packets from AS internal hosts for forwarding. It is defined as a UDP/IP network endpoint on which the SCION packets are received from the internal network. The endpoint needs to be specified as a string in the format `<ip>:<port>`. If the port is set to 0 it will be automatically assigned.

In our case, we configure the internal SCION interface to listen on the `lan` interface.

Each entry in the `neighbors` section contains a remote SCION AS to which one or multiple links are established. Each of the CORE appliances has two neighbors - `CORE` link to ISP 2/3 and a `CHILD` link to Customer 1/2. For each neighbor, we need to specify the local and remote interface network address endpoints, the link type (`CORE` or `CHILD`), the SCION interface ID, and the SCION MTU supported on this link. The SCION interface ID must be unique for each SCION interface in an AS. We set the SCION MTU to 1472 bytes for the links to the customers that use an IPv4 underlay network and 1452 bytes for the links to the other ISPs that use an IPv6 underlay network.

SCION Data Plane Configuration

s01.chzrh1.isp1

```
"router": {
  "enabled": true,
  "internal_interface": "10.0.0.100:30100"
},
"neighbors": [
  {
    "neighbor_isd_as": "1-ff00:2:1",
    "relationship": "CORE",
    "interfaces": [
      {
        "address": "[fe80:1::1]:31000",
        "administrative_state": "UP",
        "interface_id": 10,
        "remote": {
          "address": "[fe80:1::2]:31000",
```

(continues on next page)

(continued from previous page)

```
          "interface_id": 20
        },
        "scion_mtu": 1452
      }
    ]
  },
  {
    "neighbor_isd_as": "1-ff00:1:10",
    "relationship": "CHILD",
    "interfaces": [
      {
        "address": "169.254.0.1:31000",
        "administrative_state": "UP",
        "interface_id": 100,
        "remote": {
          "address": "169.254.0.2:31000",
          "interface_id": 1
        },
        "scion_mtu": 1472
      },
      {
        "address": "169.254.3.1:31000",
        "administrative_state": "UP",
        "interface_id": 101,
        "remote": {
          "address": "169.254.2.2:31000",
          "interface_id": 2
        },
        "scion_mtu": 1472
      }
    ]
  }
]
```

s01.chbrn1.isp1

```
"router": {
  "enabled": true,
  "internal_interface": "10.0.0.200:30100"
},
"neighbors": [
  {
    "neighbor_isd_as": "3-ff00:3:1",
    "relationship": "CORE",
    "interfaces": [
      {
        "address": "[fe80:2::1]:31000",
        "administrative_state": "UP",
        "interface_id": 20,
        "remote": {
          "address": "[fe80:2::2]:31000",
          "interface_id": 5
```

(continues on next page)

(continued from previous page)

```
        },
        "scion_mtu": 1452
      }
    ]
  },
  {
    "neighbor_isd_as": "1-ff00:1:20",
    "relationship": "CHILD",
    "interfaces": [
      {
        "address": "169.254.1.1:31000",
        "administrative_state": "UP",
        "interface_id": 200,
        "remote": {
          "address": "169.254.1.2:31000",
          "interface_id": 1
        },
        "scion_mtu": 1472
      },
      {
        "address": "169.254.2.1:31000",
        "administrative_state": "UP",
        "interface_id": 201,
        "remote": {
          "address": "169.254.3.2:31000",
          "interface_id": 2
        },
        "scion_mtu": 1472
      }
    ]
  }
]
```

15.4 Cluster Configuration

The CORE appliances are deployed in a sharded manner as part of a cluster. To still work as a single unit, the appliances exchange topology and SCION beacon and path information amongst each other.

The cluster configuration includes the local cluster endpoint and the list of peers that are part of the cluster. For CORE deployments we recommend to use automatic topology synchronization (see *Cluster* for more details).

Cluster Configuration

s01.chzrh1.isp1

```
"cluster": {
  "synchronization": {
    "address": "10.0.0.100:40000"
  },
```

(continues on next page)

(continued from previous page)

```
  "peers": [
    {
      "name": "s01.chbrn1.isp1",
      "synchronization": {
        "address": "10.0.0.200:40000"
      }
    }
  ]
},
```

s01.chbrn1.isp1

```
"cluster": {
  "synchronization": {
    "address": "10.0.0.200:40000"
  },
  "peers": [
    {
      "name": "s01.chzrh1.isp1",
      "synchronization": {
        "address": "10.0.0.100:40000"
      }
    }
  ]
},
```

15.5 Complete Configuration

For completeness, we provide the full configuration of each CORE appliance:

Complete Configuration s01.chzrh1.isp1

```
{
  "config": {
    "interfaces": {
      "ethernets": [
        {
          "addresses": ["10.0.0.100/24"],
          "name": "lan",
          "gateway": {
            "ipv4_gateway": "10.0.0.1"
          }
        },
        {
          "addresses": ["fe80:1::1/64"],
          "name": "ens1"
        },
        {
          "name": "ens2"
```

(continues on next page)

(continued from previous page)

```
        }
      ],
      "vlans": [
        {
          "name": "ens2.100",
          "addresses": ["169.254.0.1/30"],
          "link": "ens2",
          "id": 100
        },
        {
          "name": "ens2.200",
          "addresses": ["169.254.3.1/30"],
          "link": "ens2",
          "id": 200
        }
      ]
    },
    "scion": {
      "ases": [
        {
          "isd_as": "1-ff00:0:1",
          "forwarding_key": "Zm9yd2FyZGluZ19rZXk=",
          "core": true,
          "scion_mtu": 1472,
          "shard_id": 1,
          "control": {
            "address": "10.0.0.100:0",
            "enabled": true
          },
          "cppki": {
            "issuers": [
              {
                "isd_as": "1-ff00:0:100",
                "priority": 0
              },
              {
                "isd_as": "1-ff00:0:200",
                "priority": 1
              }
            ]
          },
          "router": {
            "enabled": true,
            "internal_interface": "10.0.0.100:30100"
          },
          "neighbors": [
            {
              "neighbor_isd_as": "1-ff00:2:1",
              "relationship": "CORE",
              "interfaces": [
                {
                  "address": "[fe80:1::1]:31000",
```

(continues on next page)

(continued from previous page)


```
                "administrative_state": "UP",
                "interface_id": 10,
                "remote": {
                  "address": "[fe80:1::2]:31000",
                  "interface_id": 20
                },
                "scion_mtu": 1452
              }
            ]
          },
          {
            "neighbor_isd_as": "1-ff00:1:10",
            "relationship": "CHILD",
            "interfaces": [
              {
                "address": "169.254.0.1:31000",
                "administrative_state": "UP",
                "interface_id": 100,
                "remote": {
                  "address": "169.254.0.2:31000",
                  "interface_id": 1
                },
                "scion_mtu": 1472
              },
              {
                "address": "169.254.3.1:31000",
                "administrative_state": "UP",
                "interface_id": 101,
                "remote": {
                  "address": "169.254.2.2:31000",
                  "interface_id": 2
                },
                "scion_mtu": 1472
              }
            ]
          }
        ]
      }
    ]
  },
  "cluster": {
    "synchronization": {
      "address": "10.0.0.100:40000"
    },
    "peers": [
      {
        "name": "s01.chbrn1.isp1",
        "synchronization": {
          "address": "10.0.0.200:40000"
        }
      }
    ]
```


(continues on next page)

(continued from previous page)

```
    },
    "management": {
      "api": {
        "address": "0.0.0.0:443"
      },
      "hostname": "s01.chzrh1.isp1"
    }
  }
}
```

Complete Configuration s01.chbrn1.isp1

```
{
  "config": {
    "interfaces": {
      "ethernets": [
        {
          "addresses": ["10.0.0.200/24"],
          "name": "lan",
          "gateway": {
            "ipv4_gateway": "10.0.0.1"
          }
        },
        {
          "addresses": ["fe80:2::1/64"],
          "name": "ens1"
        },
        {
          "name": "ens2"
        }
      ],
      "vlans": [
        {
          "name": "ens2.100",
          "addresses": ["169.254.1.1/30"],
          "link": "ens2",
          "id": 100
        },
        {
          "name": "ens2.200",
          "addresses": ["169.254.2.1/30"],
          "link": "ens2",
          "id": 200
        }
      ]
    },
    "scion": {
      "ases": [
        {
          "isd_as": "1-ff00:0:1",
```

(continues on next page)

(continued from previous page)

```
        "forwarding_key": "Zm9yd2FyZGluZ19rZXk=",
        "core": true,
        "scion_mtu": 1472,
        "shard_id": 2,
        "control": {
          "address": "10.0.0.200:0",
          "enabled": true
        },
        "cppki": {
          "issuers": [
            {
              "isd_as": "1-ff00:0:100",
              "priority": 0
            },
            {
              "isd_as": "1-ff00:0:200",
              "priority": 1
            }
          ]
        },
        "router": {
          "enabled": true,
          "internal_interface": "10.0.0.200:30100"
        },
        "neighbors": [
          {
            "neighbor_isd_as": "3-ff00:3:1",
            "relationship": "CORE",
            "interfaces": [
              {
                "address": "[fe80:2::1]:31000",
                "administrative_state": "UP",
                "interface_id": 20,
                "remote": {
                  "address": "[fe80:2::2]:31000",
                  "interface_id": 5
                },
                "scion_mtu": 1452
              }
            ]
          },
          {
            "neighbor_isd_as": "1-ff00:1:20",
            "relationship": "CHILD",
            "interfaces": [
              {
                "address": "169.254.1.1:31000",
                "administrative_state": "UP",
                "interface_id": 200,
                "remote": {
                  "address": "169.254.1.2:31000",
                  "interface_id": 1
```

(continues on next page)

(continued from previous page)

```
                  },
                  "scion_mtu": 1472
                },
                {
                  "address": "169.254.2.1:31000",
                  "administrative_state": "UP",
                  "interface_id": 201,
                  "remote": {
                    "address": "169.254.3.2:31000",
                    "interface_id": 2
                  },
                  "scion_mtu": 1472
                }
              ]
            }
          ]
        }
      ]
    },
    "cluster": {
      "synchronization": {
        "address": "10.0.0.200:40000"
      },
      "peers": [
        {
          "name": "s01.chzrh1.isp1",
          "synchronization": {
            "address": "10.0.0.100:40000"
          }
        }
      ]
    },
    "management": {
      "api": {
        "address": "0.0.0.0:443"
      },
      "hostname": "s01.chbrn1.isp1"
    }
  }
}
```

CHAPTER

SIXTEEN

ANAPAYA GATE

The Anapaya GATE implements the IP-in-SCION tunneling module and is typically deployed by ISPs to collect the IP traffic of their residential customers which should be routed into the SCION network.

In the following, we describe a typical small deployment scenario, where an ISP operates two CORE appliances (as described in *Anapaya CORE*) and additionally two GATE instances.

This GATE setup is a best practice because it increases the redundancy for residential customers. Further, it allows the ISP to perform maintenance on one of the GATE appliances without causing a service interruption.

16.1 Topology

Our target topology contains the following elements:

- ISP 1 deploys two CORE appliances, `s01.chzrh1.isp1` and `s01.chbrn1.isp1`, that are connected via the internal network `10.0.0.0/24`.
- ISP 1 deploys two GATE appliances, `s02.chzrh1.isp1` and `s02.chbrn1.isp1`, that are connected via the internal network `10.0.0.0/24` to the CORE appliances. The two GATE appliances are both connected to the BGP network of the ISP via a BGP peering (using the `10.10.0.0/24` and `10.20.0.0/24` networks).
- Residential customers, which belong to the BGP network of ISP 1.
- `gate-customer-1` and `gate-customer-2` which are two organizations that are connected to the SCION Internet using an Anapaya EDGE appliance. They are customers of the Anapaya GATE solution of ISP 1, through which they allow their employees to access critical services.
  - `gate-customer-1` exposes services, namely a VPN server, in the `192.0.2.0/28` range.
  - `gate-customer-2` exposes services in the `203.0.113.0/24` range.

We will now configure the GATE appliances step-by-step to implement the above scenario.

16.2 Network Interface Configuration

First, we will look at the configuration for the network interfaces. In the setup, there are two physical network interfaces - one for the internal network `lan` and one for the BGP peering to the ISP's BGP network `bgp`.

Please refer to the *Network Interface Configuration* for guidance on how to configure network interfaces and to *Network Interfaces* for the full documentation on network interface configuration.

Network Interface Configuration

s02.chzrh1.isp1

```
"interfaces": {
  "ethernets": [
    {
      "addresses": ["10.0.0.101/24"],
      "name": "lan",
      "gateway": {
        "ipv4_gateway": "10.0.0.1"
      }
    },
    {
      "addresses": ["10.10.0.2/24"],
      "name": "bgp"
    }
  ]
},
```

s02.chbrn1.isp1

```
"interfaces": {
  "ethernets": [
    {
      "addresses": ["10.0.0.201/24"],
      "name": "lan",
      "gateway": {
        "ipv4_gateway": "10.0.0.1"
      }
    },
    {
      "addresses": ["10.20.0.2/24"],
      "name": "bgp"
    }
  ]
},
```

16.3 BGP Configuration

GATE appliances are connected to the BGP network of the ISP. Over these BGP peerings, the GATE announces reachable remote prefixes into the ISP's internal BGP network and reannounces BGP announcements received from the ISP's BGP routers to remote SCION ASes. Therefore, each GATE appliance needs to set up at least one BGP session to a BGP router of the ISP.

In the example the following sessions are configured:

- `s02.chzrh1.isp1` with local IP `10.10.0.2` has a BGP session with the BGP router of the ISP with peer IP `10.10.0.1`.
- `s02.chbrn1.isp1` with local IP `10.20.0.2` has a BGP session with the BGP router of the ISP with peer IP `10.20.0.1`.

Note: For the peering a private BGP AS number is used on the GATE appliance. The BGP router of the ISP can use its usual public BGP AS number or use a private AS number as well.

BGP Configuration

s02.chzrh1.isp1

```
"bgp": {
  "global": {
    "as": 65302,
    "router_id": "10.10.0.2"
  },
  "neighbors": [
    {
      "enabled": true,
      "local_as": 65302,
      "neighbor_address": "10.10.0.1",
      "peer_as": 64496
    }
  ]
},
```

s02.chbrn1.isp1

```
"bgp": {
  "global": {
    "as": 65302,
    "router_id": "10.20.0.2"
  },
  "neighbors": [
    {
      "enabled": true,
      "local_as": 65302,
      "neighbor_address": "10.20.0.1",
      "peer_as": 64496
    }
  ]
},
```

The full documentation on the BGP configuration can be found in *Border Gateway Protocol (BGP)*.

16.4 SCION Configuration

Next, we will take a look at the SCION configuration. The SCION section contains the configuration of the SCION protocol and AS. For GATE appliances, we only need the *general AS* configuration section.

16.4.1 General AS Configuration

Each SCION AS has several general AS configuration options such as the ISD-AS identifier, the AS forwarding secret key, and a human-readable description of the AS. For the full list of the general AS configuration options, please refer to *General AS Configuration*.

For the configuration of a GATE appliances, we need the following fields:

- `isd_as`
- `scion_mtu`

Please refer to *General AS Configuration* for details, since the values for the CORE appliances are equal to the values of the GATE appliances.

General AS Configuration

s02.chzrh1.isp1

```
"scion": {
  "ases": [
    {
      "isd_as": "1-ff00:0:1",
      "scion_mtu": 1472
    }
  ]
},
```

s02.chbrn1.isp1

```
"scion": {
  "ases": [
    {
      "isd_as": "1-ff00:0:1",
      "scion_mtu": 1472
    }
  ]
},
```

16.5 Cluster Configuration

The GATE appliances are deployed in a sharded manner as part of a cluster together with the CORE appliances. The GATE appliances exchange topology information with the CORE appliances.

The cluster configuration includes the local cluster endpoint and the list of peers that are part of the cluster. For CORE and GATE deployments we recommend to use automatic topology synchronization (see *Cluster* for more details).

Cluster Configuration

s02.chzrh1.isp1

```
"cluster": {
  "synchronization": {
    "address": "10.0.0.101:40000"
  },
  "peers": [
    {
      "name": "s01.chbrn1.isp1",
      "synchronization": {
        "address": "10.0.0.200:40000"
      }
    },
    {
      "name": "s01.chzrh1.isp1",
      "synchronization": {
        "address": "10.0.0.100:40000"
      }
    },
    {
      "name": "s02.chbrn1.isp1",
      "synchronization": {
        "address": "10.0.0.201:40000"
      }
    }
  ]
},
```

s02.chbrn1.isp1

```
"cluster": {
  "synchronization": {
    "address": "10.0.0.201:40000"
  },
  "peers": [
    {
      "name": "s01.chbrn1.isp1",
      "synchronization": {
        "address": "10.0.0.200:40000"
      }
    },
    {
      "name": "s01.chzrh1.isp1",
```

(continues on next page)

(continued from previous page)

```
      "synchronization": {
        "address": "10.0.0.100:40000"
      }
    },
    {
      "name": "s02.chzrh1.isp1",
      "synchronization": {
        "address": "10.0.0.101:40000"
      }
    }
  ]
},
```

For GATE appliances to be integrated into the existing cluster of CORE appliances, they need to be added to the `cluster/peers` section of the CORE appliances.

Additional CORE Cluster Configuration

```
{
  "name": "s02.chbrn1.isp1",
  "synchronization": {
    "address": "10.0.0.201:40000"
  }
},
{
  "name": "s02.chzrh1.isp1",
  "synchronization": {
    "address": "10.0.0.101:40000"
  }
}
```

16.6 SCION Tunneling Configuration

The SCION tunneling configuration enables the IP-in-SCION tunneling module of the appliance and can be used to configure IP tunnels towards customers of the GATE.

The full documentation on the IP-in-SCION tunneling configuration can be found in *IP-in-SCION Tunneling*.

In this example, we configure both GATE appliances for two customers `gate-customer-1` and `gate-customer-2`.

- `gate-customer-1` owns the SCION AS with ISD-AS number `1-ff00:1:234`. This SCION AS announces the IP prefix `192.0.2.0/28` to the GATE instances via the *SCION Gateway Routing Protocol (SGRP)*.
- `gate-customer-2` owns the SCION AS with ISD-AS number `1-ff00:2:5b`. This SCION AS announces the IP prefix `203.0.113.0/24` to the GATE instances.

For both customers we create a domain configuration which contains:

- `prefixes.accept_filter` to filter the prefixes which the GATE receives from the customer,
- `prefixes.announce_filter` to filter the prefixes which the GATE announces to the customer,

- `remote_isd_ases` to list the SCION ISD-AS numbers of the customer,
- `traffic_policies` to influence what paths are chosen towards the customer. For simplicity, we configure the a default traffic policy that allows the GATE to choose any SCION path to the remote destination for any kind of traffic. Refer to *IP-in-SCION Tunneling* for much more details on how to configure traffic policies.

SCION Tunneling Configuration

s02.chzrh1.isp1


```
"scion_tunneling": {
  "endpoint": {
    "control_port": 40201,
    "data_port": 40200,
    "enabled": true,
    "ip": "10.0.0.101",
    "probe_port": 40202
  },
  "domains": [
    {
      "default": false,
      "local_isd_ases": [
        "1-ff00:0:1"
      ],
      "name": "gate-customer-1",
      "prefixes": {
        "accept_filter": [
          {
            "action": "ACCEPT",
            "prefixes": [
              "192.0.2.0/28"
            ],
            "sequence_id": 0
          }
        ],
        "announce_filter": [
          {
            "action": "ACCEPT",
            "prefixes": [
              "0.0.0.0/0"
            ],
            "sequence_id": 0
          }
        ]
      },
      "remote_isd_ases": [
        {
          "action": "ACCEPT",
          "isd_as": "1-ff00:1:234",
          "sequence_id": 0
        }
      ],
      "traffic_policies": [
        {
```


(continues on next page)

(continued from previous page)

```
        "sequence_id": 0,
        "traffic_matcher": "default",
        "failover_sequence": [
          {
            "path_filter": "default",
            "sequence_id": 0
          }
        ]
      }
    ]
  },
  {
    "default": false,
    "local_isd_ases": [
      "1-ff00:0:1"
    ],
    "name": "gate-customer-2",
    "prefixes": {
      "accept_filter": [
        {
          "action": "ACCEPT",
          "prefixes": [
            "203.0.113.0/24"
          ],
          "sequence_id": 0
        }
      ],
      "announce_filter": [
        {
          "action": "ACCEPT",
          "prefixes": [
            "0.0.0.0/0"
          ],
          "sequence_id": 0
        }
      ]
    },
    "remote_isd_ases": [
      {
        "action": "ACCEPT",
        "isd_as": "1-ff00:2:5b",
        "sequence_id": 0
      }
    ],
    "traffic_policies": [
      {
        "sequence_id": 0,
        "traffic_matcher": "default",
        "failover_sequence": [
          {
            "path_filter": "default",
            "sequence_id": 0
```

(continues on next page)

(continued from previous page)

```
            }
          ]
        }
      ]
    }
  ],
  "path_filters": [
    {
      "acl": [
        "+"
      ],
      "name": "default"
    }
  ],
  "remotes": [
    {
      "isd_as": "1-ff00:1:234"
    },
    {
      "isd_as": "1-ff00:2:5b"
    }
  ],
  "traffic_matchers": [
    {
      "condition": "BOOL=true",
      "name": "default"
    }
  ]
},
```

s02.chbrn1.isp1

```
"scion_tunneling": {
  "endpoint": {
    "control_port": 40201,
    "data_port": 40200,
    "enabled": true,
    "ip": "10.0.0.101",
    "probe_port": 40202
  },
  "domains": [
    {
      "default": false,
      "local_isd_ases": [
        "1-ff00:0:1"
      ],
      "name": "gate-customer-1",
      "prefixes": {
        "accept_filter": [
          {
            "action": "ACCEPT",
            "prefixes": [
```

(continues on next page)

(continued from previous page)

```
            "192.0.2.0/28"
          ],
          "sequence_id": 0
        }
      ],
      "announce_filter": [
        {
          "action": "ACCEPT",
          "prefixes": [
            "0.0.0.0/0"
          ],
          "sequence_id": 0
        }
      ]
    },
    "remote_isd_ases": [
      {
        "action": "ACCEPT",
        "isd_as": "1-ff00:1:234",
        "sequence_id": 0
      }
    ],
    "traffic_policies": [
      {
        "sequence_id": 0,
        "traffic_matcher": "default",
        "failover_sequence": [
          {
            "path_filter": "default",
            "sequence_id": 0
          }
        ]
      }
    ]
  },
  {
    "default": false,
    "local_isd_ases": [
      "1-ff00:0:1"
    ],
    "name": "gate-customer-2",
    "prefixes": {
      "accept_filter": [
        {
          "action": "ACCEPT",
          "prefixes": [
            "203.0.113.0/24"
          ],
          "sequence_id": 0
        }
      ],
      "announce_filter": [
```

(continues on next page)

(continued from previous page)

```
          {
            "action": "ACCEPT",
            "prefixes": [
              "0.0.0.0/0"
            ],
            "sequence_id": 0
          }
        ]
      },
      "remote_isd_ases": [
        {
          "action": "ACCEPT",
          "isd_as": "1-ff00:2:5b",
          "sequence_id": 0
        }
      ],
      "traffic_policies": [
        {
          "sequence_id": 0,
          "traffic_matcher": "default",
          "failover_sequence": [
            {
              "path_filter": "default",
              "sequence_id": 0
            }
          ]
        }
      ]
    }
  ],
  "path_filters": [
    {
      "acl": [
        "+"
      ],
      "name": "default"
    }
  ],
  "remotes": [
    {
      "isd_as": "1-ff00:1:234"
    },
    {
      "isd_as": "1-ff00:2:5b"
    }
  ],
  "traffic_matchers": [
    {
      "condition": "BOOL=true",
      "name": "default"
    }
  ]
```

(continues on next page)

(continued from previous page)

```
},
```

CHAPTER

SEVENTEEN

APPLIANCE HARDWARE QUICK START GUIDES

These Quick Start Guides help you to get started with you new physical Anapaya appliance. Please choose the corresponding guide for your appliance.

17.1 Appliance S-Type Lanner

Welcome to the SCION world! Here you can find initial connection instructions for your Anapaya S-type device. Once the device is unpacked and installed in your technical environment, proceed with the following steps on the Rear Panel:

1. Connect the RJ45 cable to the termination point of your WAN access on port `3`.
2. Connect the RJ45 cable to your LAN on port `5`.
3. Plug the power supply `DC`.
4. Turn the device on by pressing the power button `P`.

Note: Your support may indicate other ports to connect the RJ45 cables to.

After a few minutes, the LEDs on the front panel should have the following statuses:

1. System should be solid green.
2. Power should be solid green.
3. The LED 3/4 represents the WAN port(s). The link status of the port `3` should be blinking amber.
4. The LED 5/6/7/8 represents the LAN port(s). The link status of the port `5` should be blinking amber.

Note: If the IP addresses has not been assigned automatically via DHCP:

1. Connect a serial cable to the console port (`5` on the first picture).
2. Then connect to the serial console using a baud rate of `115200`. Authenticate to the appliance using the default credentials, login: `anapaya` and password: `anapaya`.
3. Once logged in, you can setup the WAN IP parameters as indicated by your support. For more information consult docs.anapaya.net.

17.1.1 The possible statuses of the RJ45 network LEDs (3 to 8) are

Upper LED (speed)	Solid green	Operating as a FastEthernet connection (100 Mbps)
	Solid amber	Operating as a GigaEthernet connection (1'000 Mbps)
	Off	No link has been established
Lower LED (link status)	Solid amber	Link has been established and there is no activity on this port
	Blinking amber	Link has been established and there is activity on this port
	Off	No link has been established

17.1.2 Interface mapping

Interface Name	Port Nr. on Device	Port Usage	Port Type
enp2s0f0	1	Unassigned	Fiber – SFP 1G
enp2s0f1	2	Unassigned	Fiber – SFP 1G
enp2s0f2	3	Default WAN interface	RJ45 – 1000BaseT
enp2s0f3	4	Unassigned	RJ45 – 1000BaseT
enp10s0f0	7	Unassigned	RJ45 – 1000BaseT
enp10s0f1	8	Unassigned	RJ45 – 1000BaseT
enp8s0f0	5	Default LAN interface	RJ45 – 1000BaseT
enp8s0f1	6	Unassigned	RJ45 – 1000BaseT

The official manual of the S-type hardware (Lanner L-1515 4C) can be found at:

https://www.whiteboxsolution.com/product/l-1515/

17.2 Appliance L-Type Supermicro

Welcome to the SCION world! Here you can find initial connection instructions for your Anapaya L-type device to set it up as an EDGE appliance. When installing the appliance as a CORE appliance you may need to change the port assignment. Once the device is unpacked and installed in your rack environment, on the rear panel (depicted below) connect power both on port `A` and port `B`.

On the front panel of the device:

1. Plug the SFP28 transceiver into port `7` and connect the LAN access fiber optics to it.
2. Plug another SFP28 transceiver into port `8` and connect the WAN access fiber optics to it.
3. Optionally, connect the BMC port to the management network.
4. Turn the device on by pressing the power button `P`.

Note: Your support may indicate other ports to connect the cables to. If, for example, RJ45 patch cables are used, port 5 will be the LAN port an port 6 the WAN port. Instead of transceivers and fibers, also DACs can be used (Direct Attached Cables).

After a few minutes, the system should have booted, and you should see a login prompt.

Note: If the IP addresses has not been assigned automatically via DHCP:

1. Connect a keyboard and a monitor to the device.
2. Authenticate to the appliance using the default credentials (login: `anapaya` and password: `anapaya`).
3. Once logged in, you can setup the WAN IP parameters as indicated by your support. For more information consult docs.anapaya.net.

17.2.1 Interface mapping

Interface Name	Port Nr. on Device	Port Usage	Port Type
eno1	1	BMC & OS Management	RJ45 1G
eno2	2	Unassigned	RJ45 1G
eno3	3	Unassigned	RJ45 1G
eno4	4	Unassigned	RJ45 1G
eno5	5	Default RJ45 LAN interface	RJ45 10G
eno6	6	Default RJ45 WAN interface	RJ45 10G
eno7	7	Default SFP28 LAN interface	SFP28 25G
eno8	8	Default SFP28 WAN interface	SFP28 25G
enp22s0f0	e0	Unassigned	SFP28 25G
enp22s0f1	e1	Unassigned	SFP28 25G
enp22s0f3	e2	Unassigned	SFP28 25G
enp22s0f4	e3	Unassigned	SFP28 25G

The official manual of the L-type hardware (SuperServer SYS-110D-8C-FRDN8TP) can be found at:

https://www.supermicro.com/en/products/system/iot/1u/sys-110d-8c-frdn8tp

17.3 Appliance XL-Type Supermicro

Welcome to the SCION world! Here you can find initial connection instructions for your Anapaya XL-type device to set it up as an EDGE appliance. When installing the appliance as a CORE appliance you may need to change the port assignment. Once the device is unpacked and installed in your rack environment, on the rear panel (depicted below) connect power both on port `A` and port `B`.

On the front panel of the device:

1. Plug the QSFP28 transceiver into port `e0` and connect the LAN access fiber optics to it.
2. Plug another QSFP28 transceiver into port `e1` and connect the WAN access fiber optics to it.
3. Optionally, connect the BMC port to the management network.
4. Turn the device on by pressing the power button `P`.

Note: Your support may indicate other ports to connect the cables to. If, for example, RJ45 patch cables are used, port 5 will be the LAN port an port 6 the WAN port. Instead of transceivers and fibers, also DACs can be used (Direct Attached Cables).

After a few minutes, the system should have booted, and you should see a login prompt.

Note: If the IP addresses has not been assigned automatically via DHCP:

1. Connect a keyboard and a monitor to the device.
2. Authenticate to the appliance using the default credentials (login: `anapaya` and password: `anapaya`).
3. Once logged in, you can setup the WAN IP parameters as indicated by your support. For more information consult docs.anapaya.net.

17.3.1 Interface mapping

Interface Name	Port Nr. on Device	Port Usage	Port Type
eno1	1	BMC & OS Management	RJ45 1G
eno2	2	Unassigned	RJ45 1G
eno3	3	Unassigned	RJ45 1G
eno4	4	Unassigned	RJ45 1G
eno5	5	Default RJ45 LAN interface	RJ45 10G
eno6	6	Default RJ45 WAN interface	RJ45 10G
eno7	7	Default SFP28 LAN interface	SFP28 25G
eno8	8	Default SFP28 WAN interface	SFP28 25G
enp22s0f0	e0	Unassigned	QSFP28 100G
enp22s0f1	e1	Unassigned	QSFP28 100G

The official manual of the XL-type hardware (SuperServer SYS-110D-16C-FRAN8TP) can be found at:

https://www.supermicro.com/en/products/system/iot/1u/sys-110d-16c-fran8tp

The guides can also be downloaded in PDF format:

- `Appliance S-Type Lanner - Quick Start Guide (One Pager)`
- `Appliance L-Type Supermicro - Quick Start Guide (One Pager)`
- `Appliance XL-Type Supermicro - Quick Start Guide (One Pager)`

CHAPTER

EIGHTEEN

INSTALLING THE APPLIANCE BASE IMAGE

The Appliance Base Image can install the Anapaya Appliance on bare metal or as a VM running on a hypervisor. The minimal system requirements are described in the Computing resources guide. The base image is provided in two different formats: (1) a `qcow2` image purposed for a fast setup of VM's and (2) an `ISO` image that can be used to provision bare metal machines.

Note: You may find the latest base image releases on the *release notes* page.

The images are hosted on cloudsmith.io. To download them, two pieces of information are required:

- Access token

  The access token is provided to you by Anapaya as part of your software license.

- Base image version

  The version of the base image needs to be specified. It consists of three parts: the anapaya system version, the anapaya scion version and a counter for the base image version. For example, `sys_v2.1.0-scion_v0.32.3-1` is an unique version identifying an appliance base image. It specifies the `anapaya-system` package version 2.1.0 and the `anapaya-scion` package version 0.32.1. The last number is a build counter which is increased if there are multiple builds per version combination.

To download the `ISO` or `qcow2` images, use the following commands:

```
# ISO Image
wget https://dl.cloudsmith.io/$ACCESS_TOKEN/anapaya/stable/raw/names/anapaya-
↪appliance-base/versions/$VERSION/anapaya-appliance-base-$VERSION.iso
# QCOW2 BIOS Version
wget https://dl.cloudsmith.io/$ACCESS_TOKEN/anapaya/stable/raw/names/anapaya-
↪appliance-base-bios-qcow2/versions/$VERSION/anapaya-appliance-base-bios-
↪$VERSION.qcow2
# QCOW2 UEFI Version
wget https://dl.cloudsmith.io/$ACCESS_TOKEN/anapaya/stable/raw/names/anapaya-
↪appliance-base-uefi-qcow2/versions/$VERSION/anapaya-appliance-base-uefi-
↪$VERSION.qcow2
```

Verify that the checksum of the downloaded image matches the one documented on the release notes page:

```
sha256sum <IMAGE>
```

Note: The variables (`$VERSION`, `$ACCESS_TOKEN`) in the commands can either be replaced directly or set as environment variables.

Note: The Anapaya Appliance Base Image is a full system image that includes all the necessary modules to bootstrap an Anapaya Appliance instance. The system image is hardened by Anapaya. However, the configuration and hardening of the host or hypervisor that hosts the Anapaya Appliance is the responsibility of the user. Please refer to the corresponding vendor guidelines and best practices.

18.1 Install the base image on a hypervisor

1. Copy the `qcow2` image to the hypervisor and install it according to the platform documentation.
2. Allocate resources to the Anapaya Appliance (at least **4G of RAM** and **2-4 vCPUs**) and then boot the VM.

18.2 Install the base image on bare metal

18.2.1 Create bootable USB installer on Ubuntu

1. Insert a USB drive and find its device name path (e.g. `/dev/sdb`) by running the following command (look for type disk):

```
lsblk
```

2. Unmount the USB device in case it is mounted, by running:

```
sudo umount /dev/<usb-dev>
```

3. Burn the ISO to the USB drive

```
sudo dd bs=4M if=<path-to-iso> of=/dev/<usb-dev> status=progress oflag=sync␣
↪conv=fsync
```

Note: If your machine is running Windows or macOS or you prefer a GUI, you can download a tool such as Etcher, to burn the `ISO` image to a USB drive to complete this step.

18.2.2 Configure the installer

In this optional step, you can configure the installer such that no interaction is required after initiating the boot.

1. To store the base installer user configuration on the USB drive, mount the third partition of the USB device, by running:

```
sudo mount /dev/<usb-dev>3 /mnt
```

2. Create an installer configuration file at `/mnt/anapaya-user-config.yaml` containing the boot disk name, and a parameter `poweroff_final` to make the machine power off, once the installation has been completed.

```
disk: sda
poweroff_final: true
```

There are a few more parameters that can be configured depending on your setup. Only set the listed parameters if you are sure you need to change the default value.

installer parameters:

```
# Hostname: the target system's hostname.
hostname: anapaya-appliance
# User: the default user name
user: anapaya
# Password: the default user password.
password: anapaya
# InteractiveNetworkSetup: If non-interactive network setup, the installer␣
↪configures IPv4 DHCP on all interfaces.
interactive_network_setup: false
# Disk: the system's disk name to install the software (default not set).
# When no disk is configured, it implies interactive storage setup which is the
# default behavior.
# Based on the disk size, the layout would be as follow:
# - < 12GB: not supported
# - < 20GB: 8GB root lvm partition, leftover space for the var lvm partition
# - < 40GB: 16GB root lvm partition, leftover space for the var lvm partition
# - >= 40GB: 24GB root lvm partition, leftover space for the var lvm partition
disk: sde
# SecondaryDisk: If a secondary disk is specified it will be used for the systems /
↪var directory.
# The root lvm partition will use the entirety of the primary disk in this case.
secondary_disk: sdb
# PowerOffPreBoot: when `true`, it will shutdown the system before the first boot.
poweroff_preboot: false
# PowerOffFinal: when `true`, it will shutdown the system after the installation has␣
↪completed.
poweroff_final: false
```

3. Unmount the third partition, by running:

```
sudo umount /dev/<usb-dev>3 /mnt
```

18.2.3 Boot from the installer

Insert the USB media into the target system and initiate the boot sequence by powering on the machine.

Note: It might be necessary to change the boot order preference settings in the BIOS of the machine to favour USB devices.

If you added a configuration to the installer, the installation will start automatically, otherwise you need to select the installation target disk manually. During the installation process, the machine will restart a few times.

Note: During the final step of the installation, the OS is booted and a login prompt appears. To check the installation progress you can authenticate with the default credentials, or the credentials that were configured and run:

```
watch cloud-init status
```

After the installation has completed, the machine shuts itself down.

18.3 Next steps

You are now ready to provision the Anapaya Appliance. You can login via SSH or use the management API that is exposed on all interfaces. The default credentials are `anapaya:anapaya`.

> **Warning:** Make sure to immediately change the SSH and API credentials. Otherwise the security of your device might be compromised.

A good place to start is the *Connecting to the Appliance* section. If you are already familiar with the appliance configuration, you can also use the *Appliance Configuration Reference* to set up and configure the appliance.

CHAPTER

NINETEEN

UPDATING AN APPLIANCE

We recommend that you update your Anapaya appliance in a timely manner to stay current with the latest enhancements and fixes. This guide provides a general step-by-step procedure to update your Anapaya appliance with the Anapaya appliance installer. For a more extensive explanation of the update procedure, refer to the *Software Updates* documentation. Upgrading between minor versions is generally safe, and automatic rollback in case of failure is supported.

Note: Make sure to consult the version specific release and upgrade notes referenced on the *Release Notes* page. They will mention known issues that potentially might affect your upgrade, or additional steps that need to be taken during an upgrade.

19.1 Prerequisites

Access to software packages
: To download the software packages, you require an access token that is provided as part of your software license. If you do not have access to that token, consult with your Anapaya Customer Sucess Engineer.

Access to appliance API
: To access the appliance API you need to know the configured API address and a set of credentials to authenticate to the appliance. Further you need to have network connectivity to the appliance such that you can reach it at the configured address. If you perform the update form a shell on the appliance itself, it is the localhost address `127.0.0.1`.

  Note: In the case where you perform the upgrade form a shell on the appliance itself where the host has no internet access, the keys, the signatures and the packages need to be downloaded on a host with internet access and copied to the appliance.

Credentials
: In case authentication was configured on the appliance, credentials need to be specified when interacting with the management API by extending the curl command with the `-u <username>:<password>` argument.

```
curl -u <username>:<password> <url>
```

Up-to-date public keys
: Ensure that the set of public key used for code verification is up-to-date with the currently published set on https://releases.anapaya.net/keys.json

```
curl https://releases.anapaya.net/keys.json > reference_keys.json
curl -k https://$APPLIANCE_API/api/v1/software/keys > keys.json
diff <(jq --sort-keys . reference_keys.json) <(jq --sort-keys . keys.json)
```

If the keys differ please follow instructions in the *Updating installed public keys* section to update them.

19.2 Update the anapaya-scion package

To update the anapaya-scion package, first download the package its signature then upload both files to the appliance and finally trigger the installation.

Note: The variables (`$VERSION`, `$APPLIANCE_API`, `$ACCESS_TOKEN`) in the commands can either be replaced directly or set as environment variables

```
export APPLIANCE_API=127.0.0.1
export VERSION=v0.33.5
```

---

1. Download the `anapaya-scion` package of the desired version from cloudsmith.io

```
wget https://dl.cloudsmith.io/$ACCESS_TOKEN/anapaya/stable/raw/names/anapaya-scion/
↪versions/$VERSION/anapaya-scion-$VERSION.tar.gz
```

2. Download the `anapaya-scion` package signature from the Anapaya website

```
curl https://releases.anapaya.net/anapaya-scion-$VERSION.tar.gz.signatures.json >␣
↪signatures.json
```

3. Upload the package signatures to the appliance installer

```
curl -k https://$APPLIANCE_API/api/v1/software/signatures/scion/$VERSION -X POST -d␣
↪@signatures.json
```

4. Upload the package to the appliance installer

```
curl -k https://$APPLIANCE_API/api/v1/software/scion/packages/local -X POST --data-
↪binary "@anapaya-scion-$VERSION.tar.gz"
```

5. Trigger the installation

```
curl -k https://$APPLIANCE_API/api/v1/software/scion/install -X POST -d '{"version
↪": "'"$VERSION"'"}'
```

Note: In case of a failed installation of an `anapaya-scion` package, a rollback to the original version is triggered automatically and no interaction is required.

Tip: You can check the installation progress by running the following command:

```
watch curl -k https://$APPLIANCE_API/api/v1/software/scion/install/$ID
```

The `$ID` corresponds to the installation identifier of the installation that you have triggered. You will find it in the output of the triggered installation.

6. Finally, check if the installation has completed and the new version is installed.

```
curl -k https://$APPLIANCE_API/api/v1/software/scion/installed
```

19.2.1 Rollback was triggered

When a rollback is triggered due to a failed installation of the new version, investigate the reason by looking at the logs of the installer:

```
journalctl -u appliance-installer.service
```

Note: Sometimes, the installation of a package fails due to a lock that cannot be acquired by the installer. In that case, a rollback is triggered. To verify that the rollback was triggered because of this known issue, you can check the installer logs for the following error:

```
Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend)
```

If that is the case, an installation can just be retried.

19.3 Update the anapaya-system package

Updating an anapaya-system package is analogous to the process for the anapaya-scion package, with the difference that no rollbacks are supported.

To update the anapaya-systems package, first download the package its signature then upload both files to the appliance and finally trigger the installation.

Note: The `$VARIABLES` (`$VERSION`, `$APPLIANCE_API`, `$ACCESS_TOKEN`) in the commands can either be replaced directly or set as environment variables

```
export APPLIANCE_API=127.0.0.1
export VERSION=v2.5.0
```

1. Download the `anapaya-system` package of the desired version from cloudsmith.io

```
wget https://dl.cloudsmith.io/$ACCESS_TOKEN/anapaya/stable/raw/names/anapaya-system/
↪versions/$VERSION/anapaya-system-$VERSION.tar.gz
```

2. Download the `anapaya-system` package signature from the Anapaya website

```
curl https://releases.anapaya.net/anapaya-system-$VERSION.tar.gz.signatures.json >␣
↪signatures.json
```

3. Upload the package signatures to the appliance installer

```
curl -k https://$APPLIANCE_API/api/v1/software/signatures/system/$VERSION -X POST -
↪d @signatures.json
```

4. Upload the package

```
curl -k https://$APPLIANCE_API/api/v1/software/system/packages/local -X POST --data-
↪binary "@anapaya-system-$VERSION.tar.gz"
```

5. Trigger the installation

```
curl -k https://$APPLIANCE_API/api/v1/software/system/install -X POST -d '{"version
↪": "'"$VERSION"'"}'
```

6. Finally, check if the installation has completed and the new version is installed.

```
curl -k https://$APPLIANCE_API/api/v1/software/system/installed
```

CHAPTER

TWENTY

CONFIGURING A SCION LINK

In this user guide, we will configure a SCION link between two Anapaya appliances. Note that it does not matter whether the two appliances are EDGE, CORE, or a combination of both. The configuration is largely the same and we will point out the differences.

20.1 Determining SCION Link Parameters

A SCION link is a connection between two SCION ASes and as such constitutes a contract between the two ASes. As part of this contract, the two ASes agree on the following:

1. The relationship between the two ASes.
2. The SCION interface identifiers used by each AS.
3. The SCION Maximum Transition Unit (MTU) of the link.
4. The IP underlay network of the link.
5. The network endpoints of the link, i.e., the IP address and UDP port that the SCION routers on each AS use to communicate with each other.
6. (Optional) Custom Bidirectional Forwarding Detection (BFD) configuration.

Note: A SCION link connects two SCION ASes identified by their ISD-AS number. In a multi-ISD configuration, i.e., when multiple ISDs are configured on the same CORE or EDGE appliance, there are multiple ISD-AS identities (one for each ISD while the AS number is the same for all ISDs). Consequently, there are multiple SCION links CORE and EDGE appliances in a multi-ISD configuration, namely one SCION link for each ISD-AS identity.

20.1.1 Relationship

The relationship between the two ASes indicates the hierarchy on the network level. The relationship can be either parent-child, peer, or core. A parent-child relationship indicates that the parent AS is upstream of the child (or downstream AS). This is usually the case when the parent AS sells a service to the child AS. A peer relationship indicates that the two ASes are on the same level of the hierarchy, i.e., neither is a customer of the other. A core relationship is a special case of a peer relationship where the two ASes involved are core ASes.

Note: The current implementation of Anapaya EDGE and CORE appliances do not support the `peer` relationship.

20.1.2 SCION Interface Identifiers

The SCION interface identifier uniquely identifies a SCION interface of an AS. They are used in the SCION packet header to define the network path of a SCION packet.

The SCION interface identifier is a 16-bit number (1-65535) and must be unique within the AS. Interface ID 0 is reserved and cannot be used. How a network operator assigns interface IDs is up to them as long as each interface ID is only used once within the AS.

Note: The SCION interface ID is part of the "public contract" of a SCION AS. Other SCION ASes might use the interface IDs in their path policies, e.g., to perform traffic engineering. Thus, it is highly discouraged to change SCION interface IDs once they have been assigned.

20.1.3 SCION MTU

The SCION MTU is the maximum size of a SCION packet including the SCION header. It usually depends on the underlying IP network. Assuming a common IP MTU of 1500 bytes, the SCION MTU is either set to 1472 bytes (IPv4) or 1452 bytes (IPv6).

20.1.4 IP Underlay Network

SCION uses an IP underlay network to transport SCION packets. SCION packets are transported as UDP datagrams between two SCION routers. This has the benefit of reusing as much as possible of the existing IP infrastructure.

Anapaya EDGE and CORE appliances support IPv4 and IPv6 underlay networks. There are no restrictions on the IP underlay network, however, it is recommended to use a range from the private IP address space. Furthermore, it is common that the two endpoints of the SCION link are in the same IP subnet, e.g., a point-to-point link:

- IPv4: `169.254.0.0/16`, e.g., `169.254.0.2/30`
- IPv6: `fe80::/10`, e.g., `fe80::2/64`

However, it is also possible for the two endpoints to be on different IP networks, i.e., there is a routed network between the two endpoints. This is commonly used by SCION service providers to connect their customer's EDGE appliances to their CORE appliances on their routed access network.

20.1.5 SCION Interface Endpoints

The SCION interface endpoints are the IP address and UDP port on which the CORE or EDGE appliance sends and receives SCION packets for the SCION link. The IP address must be chosen from the IP underlay network of the SCION link. The UDP port can be chosen freely as long as the combination `<ip>:<port>` is unique on the appliance.

Tip: The default port range we use for SCION interfaces is 30100 - 39999. Documentation of our default port ranges can be found in *Default Port Allocations*.

20.1.6 Bidirectional Forwarding Detection

If a SCION link becomes unhealthy, the information is signaled to users of the SCION link via the SCION Control Message Protocol (SCMP). Anapaya CORE and EDGE appliances use Bidirectional Forwarding Detection (BFD) to determine the health of a SCION link. In most cases, the default values work well and the BFD configuration does not need to be explicitly set. In some cases, e.g., if the underlying network of the SCION link is known to be lossy, or if BFD should be disabled, the BFD configuration can be set explicitly. Please refer to the *Bidirectional Forwarding Detection* section of the configuration manual for more details and examples.

20.2 Network Interface Configuration

A SCION interface is a virtual construct and an operator is free to map them to physical or virtual network interfaces as they see fit. In the following, we present some of the common configurations.

20.2.1 One SCION Interface per Physical Network Interface

This is the simplest configuration. Each SCION interface is mapped to a single physical network interface. It has the advantage that each SCION interface is completely separated from the others and even has a dedicated physical link. The disadvantage is that many physical network interfaces are needed if many SCION interfaces need to be configured.

This option is recommended for EDGE appliances with a single SCION link to an upstream CORE appliance. It can also be used for important core links between CORE appliances if it is crucial that the core link can use the full bandwidth of the underlying physical link.

20.2.2 One SCION Interface per Virtual Network Interface

This is a similar configuration to the previous one but overcomes the main disadvantage of requiring many physical network interfaces by creating a virtual network interface for each SCION interface. This is usually done by configuring a VLAN on a physical network interface. This configuration still achieves complete separation of the SCION interfaces, but only requires a single physical interface. The disadvantage is that all the SCION interfaces share the same physical link and thus the same bandwidth.

This option is recommended for CORE appliances with multiple SCION links to downstream EDGE appliances.

20.2.3 Multiple SCION Interfaces per Network Interface

Given that a SCION interface only needs a unique `<ip>:<port>` combination, it is straightforward to map multiple SCION interfaces to a single network interface (virtual or physical). In this configuration, all SCION interfaces have the same IP address and different UDP ports. The advantage of this configuration is that it is easy to configure and requires only a single network interface. The disadvantage is that all SCION interfaces share the same IP underlay network and thus also the same physical link.

This option is recommended (or even required) for CORE appliances that establish multiple SCION links at an Internet Exchange Point (IXP), where a single IP underlay network exists for all participants of the IXP. Furthermore, in multi-ISD configurations it is also common to have multiple SCION links over the same underlay network.

20.3 Examples

20.3.1 EDGE Connecting to a CORE

In this example, we configure the EDGE appliance of `Customer 1` to connect to the CORE of `ISP 1` via a SCION link on a dedicated physical interface as shown in the figure below.

We need to configure the following:

1. The physical network interface that is used for the SCION link in the `interfaces` section of the appliance configuration.
2. The `PARENT` SCION link to `ISP 1` in the `scion/ases/neighbors` section of the appliance configuration.

The relevant parts of the configuration for the EDGE appliance of `Customer 1` are shown below.

Network Interface

```
{
  "interfaces": {
    "ethernets": [
      {
        "name": "eth0",
        "addresses": ["169.254.0.2/30"],
      }
    ]
  }
}
```

SCION Link

```
{
  "scion": {
    "ases": [
      {
        "isd_as": "1-ff00:1:10",
        "neighbors": [
          {
            "neighbor_isd_as": "1-ff00:0:1",
            "relationship": "PARENT",
            "interfaces": [
              {
                "address": "169.254.0.2"
                "administrative_state": "UP",
                "interface_id": 1,
                "scion_mtu": 1472,
                "remote": {
                  "address": "169.254.0.1",
                  "interface_id": 100
                }
              }
```

(continues on next page)

(continued from previous page)

```
            ]
          }
        ]
      }
    ]
  }
}
```

20.3.2 CORE Connecting Multiple EDGEs

In this example, we configure the CORE appliance of `ISP 1` to connect to two EDGE appliances of `Customer 1` and `Customer 2` using two SCION links. Each SCION link is configured on a virtual network interface implemented as separate VLANs on a single physical network interface as shown in the figure below.

We need to configure the following:

1. The physical network interface that is used as the link for the two VLAN interfaces in the `interfaces` section of the appliance configuration.
2. The VLAN interfaces for the two SCION links in the `interfaces` section of the appliance configuration.
3. The `CHILD` SCION links to `Customer 1` and `Customer 2` in the `scion/ases/neighbors` section of the appliance configuration.

The relevant parts of the configuration for the CORE appliance of `ISP 1` are shown below.

Network Interfaces

```
{
  "interfaces": {
    "ethernets": [
      {
        "name": "eth0"
      }
    ],
    "vlans": [
      {
        "name": "eth0.100",
        "id": 100,
        "link": "eth0",
        "addresses": ["169.254.0.1/30"]
      },
      {
        "name": "eth0.200",
        "id": 200,
        "link": "eth0",
        "addresses": ["fe80:1::1/64"]
      }
    ]
  }
}
```

SCION Links


```
{
  "scion": {
    "ases": [
      {
        "isd_as": "1-ff00:0:11",
        "neighbors": [
          {
            "neighbor_isd_as": "1-ff00:1:10",
            "relationship": "CHILD",
            "interfaces": [
              {
                "address": "169.254.0.1:31000",
                "administrative_state": "UP",
                "interface_id": 100,
                "scion_mtu": 1472,
                "remote": {
                  "address": "169.254.0.2:31000",
                  "interface_id": 1
                }
              }
            ]
          },
          {
            "neighbor_isd_as": "1-ff00:1:11",
            "relationship": "CHILD",
            "interfaces": [
              {
                "address": "[fe80:1::1]:31000",
                "administrative_state": "UP",
                "interface_id": 200,
                "scion_mtu": 1452,
                "remote": {
                  "address":"[fe80:1::2]:31000",
                  "interface_id": 1
                }
              }
            ]
          }
        ]
      }
    ]
  }
}
```

20.3.3 CORE Connecting to Multiple COREs at an IXP

In this example, configure the CORE appliance of `ISP 1` to connect to two other CORE appliances of `ISP 2` and `ISP 3` using two SCION links. The SCION links use the physical network of the IXP as the underlay network and thus have the same IP address, but different UDP ports as shown in the figure below.

We need to configure the following:

1. The physical network interface that is used to connect to the IXP's network in the `interfaces` section of the appliance configuration.
2. The `CORE` SCION links to `ISP 2` and `ISP 3` in the `scion/ases/neighbors` section of the appliance configuration.

The relevant parts of the configuration for the CORE appliance of `ISP 1` are shown below.

Network Interface

```
{
  "interfaces": {
    "ethernets": [
      {
        "name": "eth0",
        "addresses": ["192.168.1.10/24"],
      },
    ]
  }
}
```

SCION Links

```
{
  "scion": {
    "ases": [
      {
        "isd_as": "1-ff00:0:1",
        "neighbors": [
          {
            "neighbor_isd_as": "1-ff00:1:1",
            "relationship": "CORE",
            "interfaces": [
              {
                "address": "192.168.1.10:31000",
                "administrative_state": "UP",
                "interface_id": 10,
                "scion_mtu": 1472,
                "remote": {
                  "address": "192.168.1.20:31000",
                  "interface_id": 100
                }
              }
            ]
```

(continues on next page)

(continued from previous page)


```
          },
          {
            "neighbor_isd_as": "1-ff00:2:1",
            "relationship": "CORE",
            "interfaces": [
              {
                "address": "192.168.1.10:31001",
                "administrative_state": "UP",
                "interface_id": 20,
                "scion_mtu": 1472,
                "remote": {
                  "address": "192.168.1.30:31000",
                  "interface_id": 15
                }
              }
            ]
          }
        ]
      }
    ]
  }
}
```


20.4 Testing a SCION Link

There are multiple ways to test if a SCION link is properly configured. Here, we present the most common options.

20.4.1 Checking the SCION Link Status

The management API provides various debug endpoints to investigate the status of an appliance. To check the status of a SCION link, we can use the GET /debug/scion/interfaces endpoint. This endpoint returns a list of all SCION links that are configured on the appliance including their current status. To filter the list of SCION links `remote_isd_as` and `interface_id` can be used as query parameters.

For example to check the status of the SCION link between `ISP 1` and `ISP 2` from above we use the following command using cURL:

```
curl https://${appliance_ip}/api/v1/debug/scion/interfaces?remote_isd_as=1-ff00:1:10
```

or alternatively, using the `appliance-cli`:

```
appliance-cli get debug/scion/interfaces?remote_isd_as=1-ff00:1:10
```

If the `state` of the SCION link is `UP` the SCION link is properly configured.

20.4.2 Testing with `scion showpaths`

If a SCION link is properly configured, it will be automatically discovered by the SCION control plane and SCION paths using this link will be available. We can use this to test if a SCION link is properly configured by requesting SCION paths to the direct neighbor of the SCION link.

SCION paths can be requested using the `scion showpaths` command available on every Anapaya EDGE and CORE appliance. To verify that the SCION link between `ISP 1` and `ISP 2` is properly configured, we can use the following command:

```
scion showpaths 1-ff00:1:10
```

This command will show all SCION paths from `ISP 1` to `ISP 2`. If the SCION link is properly configured, the output contains a path using the newly added SCION link. Additionally, `scion showpaths` also probes the SCION path and reports its status.

Alternatively, the management API can be used to request SCION paths. Refer to the POST /tools/scion/showpaths documentation for more details.

CHAPTER

TWENTYONE

CONFIGURING A GATE CUSTOMER

In this user guide, we will configure a customer on an Anapaya GATE appliance. The guide is split into two sections: In the first section we update the configuration of the GATE appliance itself. In the second section we update the configuration of the customer EDGE appliance such that the services it exposes are accessible through the GATE.

21.1 Gathering Information

Before starting the configuration, both parties need to communicate and agree on the following parameters:

GATE Operator

- The ISD-AS number of the SCION AS where the GATE appliance is running.

GATE Customer

- The ISD-AS number of the SCION AS where the EDGE appliance is running.
- The IP prefix which should be reachable by the residential customers of the GATE operator. This needs to be a prefix with public IP addresses, which are currently not routed in the Internet.

Both parties

- Specific requirements for path selection policies.

21.1.1 Example Information

In this user guide, we will be using the following information for our example configurations.

GATE Operator

- ISD-AS Number: `1-ff00:0:1`

GATE Customer

- ISD-AS Number: `1-ff00:0:1234`
- IP Prefix: `198.51.100.0/28`

Both parties

There are no specific path selection requirements. Therefore, we will configure the default path filters and traffic matchers in the `scion_tunneling` section.

Default Path Filters and Traffic Matchers

```
"scion_tunneling": {
  "path_filters": [
    {
      "acl": ["+"],
      "name": "default"
    }
  ],
  "traffic_matchers": [
    {
      "condition": "BOOL=true",
      "name": "default"
    }
  ]
},
```

21.2 Configuring the GATE

In this section we assume the GATE appliance has been set up using the guide in *Anapaya GATE* and is already operational.

21.2.1 Remotes Configuration

On the GATE appliance, we need to add the customer ISD-AS number to the list of `scion_tunneling/remotes`.

Tunneling Remotes

```
"remotes": [
  {
    "isd_as": "1-ff00:0:1234"
  }
],
```

21.2.2 Domain Configuration

On the GATE appliance, we need to add a domain configuration to the `scion_tunneling/domains` list. As part of this configuration we have defined the following parameters:

- `name` is an identifier for the domain, here we use `gate-customer-3`,
- `local_isd_ases` is the ISD-AS number of the GATE operator, in this case `1-ff00:0:1`,
- `remote_isd_ases` contains an ISD-AS filter that matches the ISD-AS number (`1-ff00:0:1234`) of the GATE customer,
- `prefixes/announce_filter` contains a wildcard filter, which has the effect that all IP prefixes that are announced by the BGP router of the GATE operator, are re-announced via the *SCION Gateway Routing Protocol (SGRP)* protocol to the EDGE of the GATE customer,

- `prefixes/accept_filter` contains a filter that accepts the IP prefixes defined by the GATE customer (`198.51.100.0/28`),
- `traffic_policies` contains custom routing policies. In this guide, we reference the default path filters and traffic matchers as defined in *Example Information*.

Refer to *Domain Configuration* for more details on domain configuration.

Domain Configuration on GATE appliance

```
{
  "name": "gate-customer-3",
  "local_isd_ases": [
    "1-ff00:0:1"
  ],
  "remote_isd_ases": [
    {
      "action": "ACCEPT",
      "isd_as": "1-ff00:0:1234",
      "sequence_id": 0
    }
  ],
  "prefixes": {
    "announce_filter": [
      {
        "action": "ACCEPT",
        "prefixes": [
          "0.0.0.0/0"
        ],
        "sequence_id": 0
      }
    ],
    "accept_filter": [
      {
        "action": "ACCEPT",
        "prefixes": [
          "198.51.100.0/28"
        ],
        "sequence_id": 0
      }
    ]
  },
  "traffic_policies": [
    {
      "sequence_id": 0,
      "traffic_matcher": "default",
      "failover_sequence": [
        {
          "path_filter": "default",
          "sequence_id": 0
        }
      ]
    }
  ]
```

(continues on next page)

(continued from previous page)

```
},
```

21.2.3 BGP Configuration

The GATE operator then also needs to ensure, that the IP prefix `198.51.100.0/28` is accepted via the BGP session between the ISP's BGP network and the GATEs.

21.3 Configuring the EDGE Appliance

The process to configure the EDGE appliance is very similar to the configuration of the GATE appliance. We assume the EDGE appliance has been set up according to one of the examples in *Anapaya EDGE*.

21.3.1 Remotes Configuration

On the EDGE appliance, we need to add the GATE operator ISD-AS number to the list of `scion_tunneling/remotes`.

Tunneling Remotes

```
"remotes": [
  {
    "isd_as": "1-ff00:0:1"
  }
],
```

21.3.2 Domain Configuration

On the EDGE appliance, we need to add a domain configuration to the `scion_tunneling/domains` list. As part of this configuration we have defined the following parameters:

- `name` is an identifier for the domain, here we use `gate-services`,
- `default` is set to `true`, see *Default Domain* for details,
- `local_isd_ases` is the ISD-AS number of the GATE customer, in this case `1-ff00:0:1234`,
- `remote_isd_ases` contains an ISD-AS filter that matches the ISD-AS number (`1-ff00:0:1`) of the GATE operator,
- `prefixes/announce_filter` contains a filter that accepts the IP prefixes defined by the GATE customer (`198.51.100.0/28`),
- `traffic_policies` contains custom routing policies. In this guide, we reference the default path filters and traffic matchers as defined in *Example Information*.

Refer to *Domain Configuration* for more details on domain configuration.

Domain Configuration on EDGE appliance

```
{
  "name": "gate-services",
  "default": true,
  "local_isd_ases": [
    "1-ff00:0:1234"
  ],
  "remote_isd_ases": [
    {
      "action": "ACCEPT",
      "isd_as": "1-ff00:0:1",
      "sequence_id": 0
    }
  ],
  "prefixes": {
    "announce_filter": [
      {
        "action": "ACCEPT",
        "prefixes": [
          "198.51.100.0/28"
        ],
        "sequence_id": 0
      }
    ]
  },
  "traffic_policies": [
    {
      "sequence_id": 0,
      "traffic_matcher": "default",
      "failover_sequence": [
        {
          "path_filter": "default",
          "sequence_id": 0
        }
      ]
    }
  ]
},
```

Note: In case the Anapaya EDGE appliance has a BGP session to the LAN, the IP prefix `198.51.100.0/28` should be announced via BGP and the EDGE automatically re-announces it as long as it matches the `prefixes/announce_filter`.

When the Anapaya EDGE is connected to the LAN using static routing, the IP prefix `198.51.100.0/28` needs to be configured in the `scion_tunneling/static_announcements` section. Please refer to *Static Announcements* for more details.

Domain Configuration

```
"static_announcements": [
  {
    "prefixes": ["198.51.100.0/28"],
```

(continues on next page)

(continued from previous page)

```
    "sequence_id": 0
  },
],
```

Note: In cases where the Anapaya EDGE is run alongside an Internet gateway, there needs to be a mechanism to determine if the return traffic should use a path going through the Anapaya EDGE or the default Internet gateway. There are several options:

- Use BGP to connect the Anapaya EDGE to the LAN.
- Enable policy based routing on the firewall between the EDGE and the LAN.
- Enable source NAT on the Anapaya EDGE. Refer to *Configuring Source NAT to collect Users from the Anapaya GATE* for detailed information.

21.4 Testing the Connectivity

Once both the GATE and the EDGE have been configured, the connectivity can be tested. There are various ways how to test the new configuration. In the following, we present two possibilities.

21.4.1 Inspecting the Configured Domain

Using the `appliance-cli`, we can check the status of the configured domain on both appliances.

Use the following command to view the summary of a specific routing domain:

```
appliance-cli inspect scion-tunneling summary --domain <domain>
```

When both sides are correctly configured, we see the following example outputs.

GATE Domain Summary

```
$ appliance-cli inspect scion-tunneling summary --domain gate-customer-3
  DOMAIN: gate-customer-3
    PREFIXES:
      198.51.100.0/28
    TRAFFIC MATCHER: default
      FAILOVER SEQUENCE:
        STATE FILTER  HEALTHY LOCAL      REMOTE
        -->   default true    1-ff00:0:1 1-ff00:0:1234,10.4.0.1:30856
              ...
```

EDGE Domain Summary

```
$ appliance-cli inspect scion-tunneling summary --domain gate-services
  DOMAIN: gate-services
    PREFIXES:
      ...
    TRAFFIC MATCHER: default
      FAILOVER SEQUENCE:
        STATE FILTER  HEALTHY LOCAL         REMOTE
        -->   default true    1-ff00:0:1234 1-ff00:0:1,10.4.0.1:30856
              ...
```

If your output does not show the expected result, consult your monitoring setup and the *Troubleshooting & Runbooks* section.

21.4.2 Test End-To-End Connectivity

To test end-to-end connectivity, a residential user of the ISP which operates the previously configured Anapaya GATE should be able to access the services that were made accessible through the GATE. This process depends on the type of service or application that was exposed.

CHAPTER

TWENTYTWO

CONFIGURING OAUTH2 FOR THE APPLIANCE API

The management API of the appliance can be secured with OAuth2/OIDC. This guide explains how to achieve this for Auth0 and Azure AD.

The appliance supports two OAuth2 token grant flows:

OAuth2 Authorization Code Grant Flow

This flow is designed for direct interaction with a web application. In this case, the Anapaya appliance is the web application and the user accesses the appliance management UI with a browser (client). The appliance automatically redirects the user to the configured identity provider to log in and validates the JWT issued by the identity provider.

OAuth2 Client Credentials Grant Flow

This flow is designed for machine-to-machine interactions, e.g., if a daemon process wants to interact with the appliance API. The process authenticates itself at the identity provider with client ID and client secret and then gets an access token. The access token is then passed to the Anapaya appliance and validated. A user first needs to register the application with the identity provider before it can authenticate itself to the appliance API.

Note: It's also possible to configure a different tool, e.g. a CLI program with the Authorization code grant flow, the tool would then do the redirect flow and in the end just send the token to the appliance. For this case, it is enough to configure the token verification keys in the appliance, as described in the client credentials flow sections.

22.1 Auth0

22.1.1 Authorization Code Flow

1. Create an Auth0 application by following the Auth0 docs, use `Regular Web Applications` as the type.
2. Create the `appliance/reader` and `appliance/writer` roles, such that users can be assigned the different roles. For this follow the Auth0 docs to create the following two roles:
   - `appliance/reader`: For read access to the appliance API.
   - `appliance/writer`: For read and write access to the appliance API.

   Tip: `appliance:reader` and `appliance.reader` are aliases for `appliance/reader` and can be used interchangeably. The same applies to `appliance:writer` and `appliance.writer`.

   ---

3. Assign the roles to the users which should have the respective roles, following the Auth0 docs.

4. Create an action to attach the user roles to the issued JWTs, by following the Auth0 docs. Use `Login / Post Login` as the trigger. In the editor, fill in the following code:

```
exports.onExecutePostLogin = async (event, api) => {
    if (event.authorization) {
        api.idToken.setCustomClaim("https://anapaya.net/roles", event.authorization.
↪roles);
    }
};
```

   This will set the roles in the `https://anapaya.net/roles` claim in the ID token.

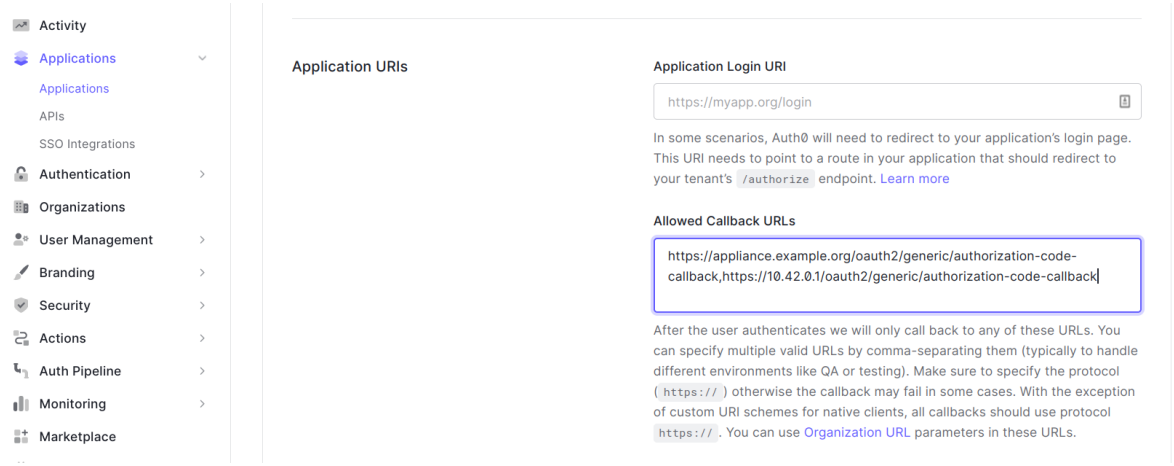
5. Deploy the action.

6. Attach the action to the login flow.

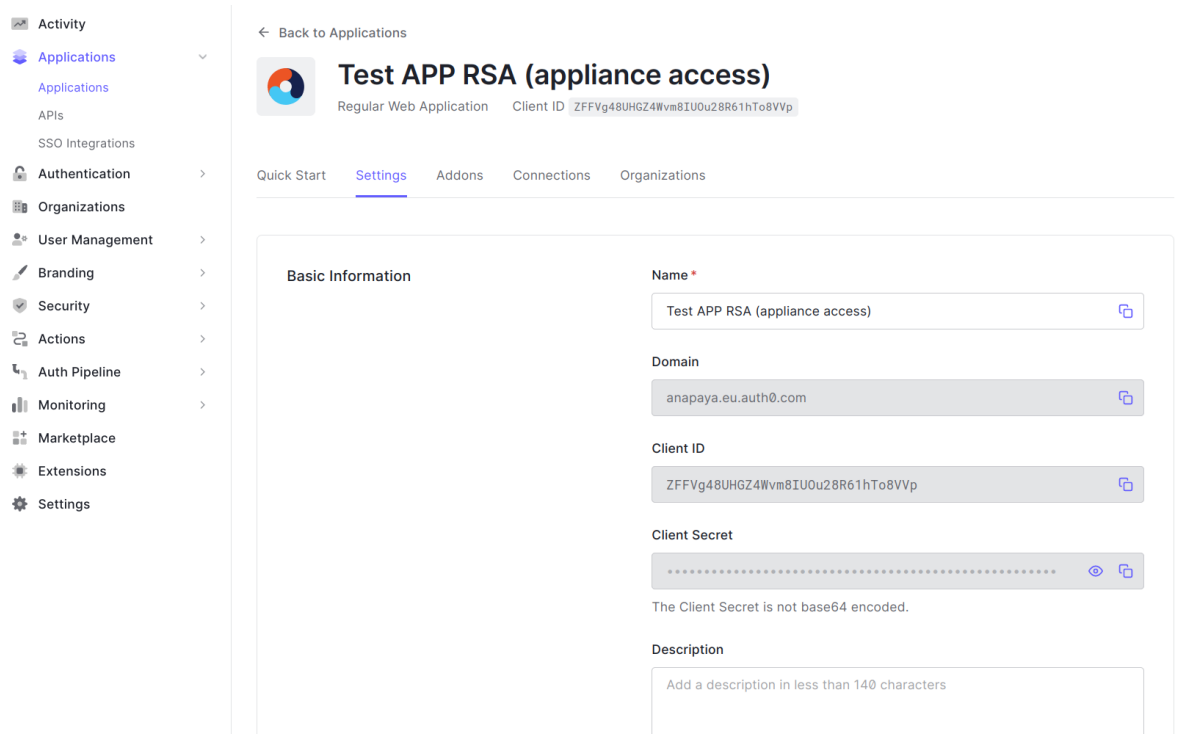
7. Configure the allowed callback URLs on Auth0. To do this, go to the application that was created previously and add the URL of your appliance in the `Allowed Callback URLs` field. The URL should be the following: `https://<appliance_address>/oauth2/generic/authorization-code-callback` where `<appliance_address>` needs to be replaced with the actual address of the appliance. For example, if you have a DNS entry for an appliance with name `appliance.example.org` use the following URL: `https://appliance.example.org/oauth2/generic/authorization-code-callback`. If you don't have a DNS entry you can use the IP address of the appliance, e.g., for an appliance with management IP `10.42.0.1` use the following URL `https://10.42.0.1/oauth2/generic/authorization-code-callback`.

Note: You can specify multiple callback URLs by separating them with `,`. This allows you to reuse the same Auth0 application for multiple appliances.

   The configuration of the Auth0 callback URLs looks as follows for the two examples above:



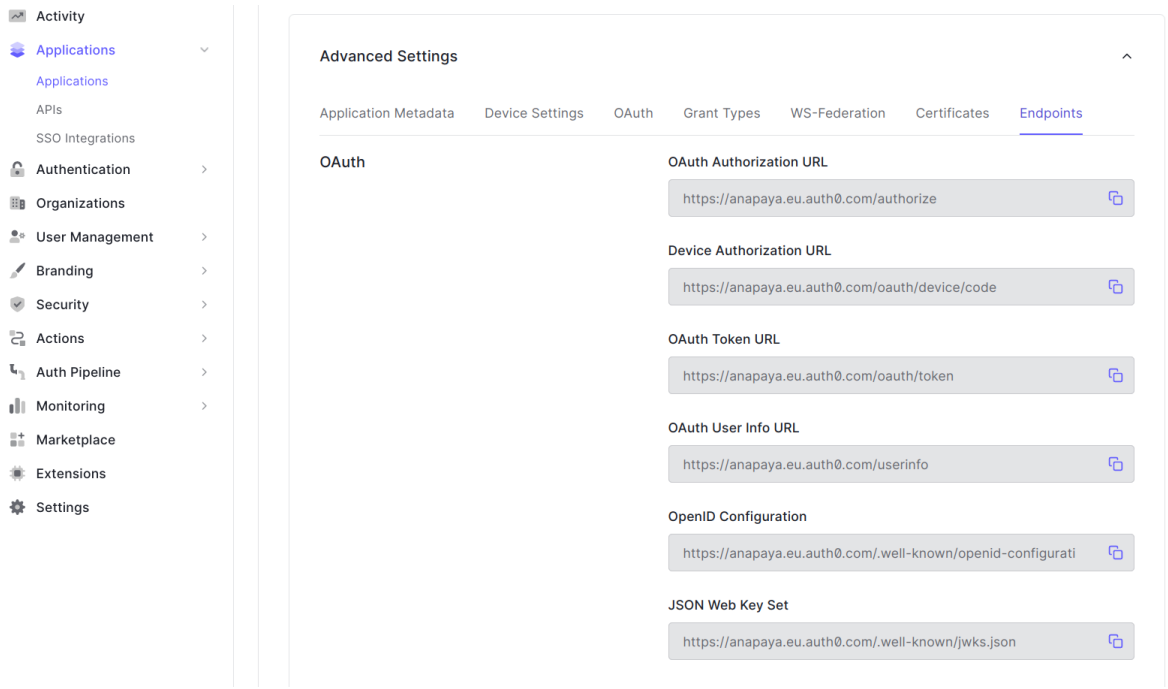
8. Extract the configuration from the Auth0 application created in the previous step. Open the application by clicking on it in the applications list. This should now show the configuration for the application as follows:



From this settings page, you can extract and deduce the following information needed to configure the appliance:

- `base_auth_url`: Extracted from the domain. In the screenshot above it is `https://anapaya.eu.auth0.com/`.
- `client_id`: Copied from the `Client ID` field. In the screenshot above it is `ZFFVg48UHGZ4Wvm8IUOu28R61hTo8VVp`
- `client_secret`: Copied from the `Client Secret` field.
- `id`: Can be chosen freely, e.g. `auth0`.
- `type`: Has to be `GENERIC` for Auth0.

For the value of the `metadata_url`, scroll to the bottom of the settings page and open the `Advanced Settings` section. There open the `Endpoints` tab. The value is then visible in the `OpenID Configuration` field:



9. Fill the appliance configuration with the data extracted in the previous step. The `oauth` part of the appliance configuration should look as follows, for the configuration extract above:

```
{
  "oauth": {
    "enabled": true,
    "identity_providers": [
      {
        "base_auth_url": "https://anapaya.eu.auth0.com/",
        "client_id": "ZFFVg48UHGZ4Wvm8IUOu28R61hTo8VVp",
        "client_secret": "client-secret",
        "id": "auth0",
        "metadata_url": "https://anapaya.eu.auth0.com/.well-known/openid-
↪configuration",
        "type": "GENERIC"
      }
    ]
  }
}
```

10. Deploy the configuration. The API will now only allow logged in users. To verify this, open a browser and type the URL of the appliance you just configured, e.g. `https://appliance.example.org/`.

> **Warning:** Deploying a wrong configuration can potentially lock you out of the API. It is advised to temporarily also configure *basic authorization* as a fallback if the OAuth configuration doesn't work.

22.1.2 Client credentials flow

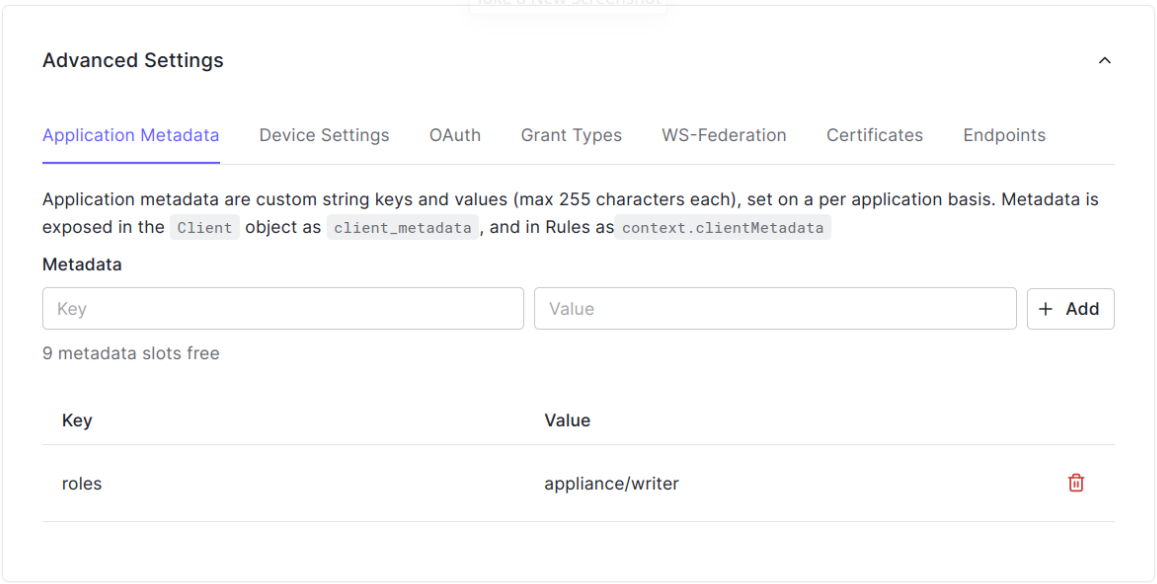
1. Follow the Auth0 docs to create an API for the appliance. As signing algorithm you need to choose `RS256`; `HS256` is not supported by the appliance. An API is needed to be able to create a *Machine to Machine* application.
2. Create an application according to the description on the Auth0 docs, use `Machine to Machine Applications` as the type and connect it to the API created in the previous step.
3. Add the roles of the application to the application metadata in the Advanced Settings section of the application:

Note: The Anapaya appliance supports the following two roles:

- `appliance/reader` for read-only access
- `appliance/writer` for read-write access

Multiple roles can be specified by separating them with a `,`.

Tip: `appliance:reader` and `appliance.reader` are aliases for `appliance/reader` and can be used interchangeably. The same applies to `appliance:writer` and `appliance.writer`.



4. Create an action to add the roles from the metadata into the issued JWT, by following the Auth0 docs. Use `M2M / Client Credentials` as the trigger. Use the following code for the action. It extracts the roles from the metadata and adds them to the JWT as a claim:

```
exports.onExecuteCredentialsExchange = async (event, api) => {
      if ("roles" in event.client.metadata) {
            api.accessToken.setCustomClaim("https://anapaya.net/roles", event.
↪client.metadata["roles"].split(","));
      }
};
```

5. Deploy the action.
6. Attach the action to the Machine to Machine flow.

7. Determine the JWKS endpoint for your Auth0 organization. Find the *JSON Web Key Set* field in the *Advances Settings* section of the Auth0 application under the *Endpoints* tab.
8. Configure the appliance with JWKS URL determined in the previous step. Given the URL `https://anapaya.eu.auth0.com/.well-known/jwks.json` the following appliance configuration should be applied:

```
{
  "oauth": {
    "enabled": true,
    "token_verification_keys": [
      {
        "id": "auth0_keys",
        "jwks_url": "https://anapaya.eu.auth0.com/.well-known/jwks.json"
      }
    ]
  }
}
```

Note: For testing it can be useful to go to the *Quick Start* tab of the Application in Auth0. There you will find an example `cURL` command that you can use to test against the configured appliance.

Warning: Deploying a wrong configuration can potentially lock you out of the API. It is advised to temporarily also configure *basic authorization* as a fallback if the OAuth configuration doesn't work.

9. Configure your application to fetch tokens from Auth0 and supply them in the header when communicating with the Anapaya appliance API.

22.2 Azure AD

22.2.1 Authorization code flow

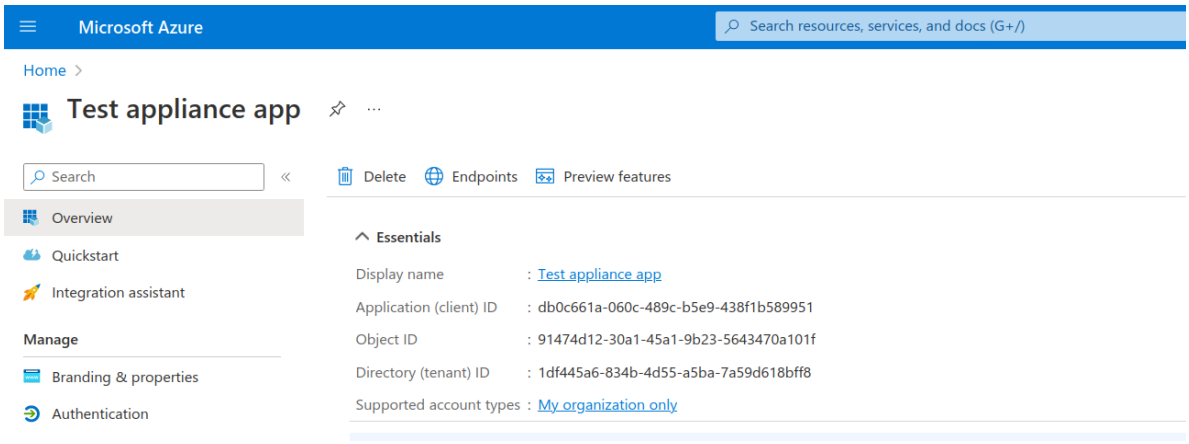
1. Create an application in Azure AD by following the documentation on learn.microsoft. As redirect URL set the redirect URL of your appliance, it must have following form: `https://<appliance_address>/oauth2/azure/authorization-code-callback` where `<appliance_address>` needs to be replaced with the actual address of the appliance. For example, if you have a DNS entry for an appliance with name `appliance.example.org` use the following URL: `https://appliance.example.org/oauth2/azure/authorization-code-callback`. If you don't have a DNS entry you can use the IP address of the appliance, e.g., for an appliance with management IP `10.42.0.1` use the following URL `https://10.42.0.1/oauth2/azure/authorization-code-callback`. As application type select *Web*.

   In the last step of the registration you will create a client secret. Make sure to remember its value as this will be used in the Anapaya appliance configuration as the `client_secret` field.
2. Add app roles according to the learn.microsoft docs. Select between the following two roles for the value:
   - `appliance/reader`: For read-only access to the appliance API.
   - `appliance/writer`: For write-write access to the appliance API.

   Furthermore, ensure that the *Allowed member types* is set to *Users/Groups*.

Tip: `appliance:reader` and `appliance.reader` are aliases for `appliance/reader` and can be used interchangeably. The same applies to `appliance:writer` and `appliance.writer`.

3. Assign roles to users by following the learn.microsoft guide.
4. Extract the required information to configure OAuth on the appliance. For this open the Azure Portal, click on *Azure Active Directory*, on the left under the *Manage* section click on *App registrations*, then select the previously created app on the right side. This should give you an overview similar to this:



From this overview you can deduce the OAuth configuration of the appliance:

- *Application (client) ID* is the `client_id` field in the configuration.
- *Directory (tenant) ID* (if present) is the `tenant_id` field in the configuration.
- The `client_secret` should be filled with secret that you created during the initial registration of the app.

Given the overview from the image above, the following configuration can be deduced:

```
{
  "oauth": {
    "enabled": true,
    "identity_providers": [
      {
        "client_id": "db0c661a-060c-489c-b5e9-438f1b589951",
        "client_secret": "<secret hidden>",
        "tenant_id": "1df445a6-834b-4d55-a5ba-7a59d618bff8",
        "id": "azure",
        "type": "AZURE_AD"
      }
    ]
  }
}
```

5. Deploy the appliance configuration.

Warning: Deploying a wrong configuration can potentially lock you out of the API. It is advised to temporarily also configure *basic authorization* as a fallback if the OAuth configuration doesn't work.

22.2.2 Client credentials flow

1. Register a new application by following the documentation on learn.microsoft. Note that there is no need to set any redirect URL.
2. Add app roles according to the learn.microsoft docs. Select between the following two roles for the value:
   - `appliance/reader`: For read-only access to the appliance API.
   - `appliance/writer`: For write-write access to the appliance API.

   Furthermore, ensure that the *Allowed member types* is set to *Applications*.

   Tip: `appliance:reader` and `appliance.reader` are aliases for `appliance/reader` and can be used interchangeably. The same applies to `appliance:writer` and `appliance.writer`.
3. Assign the role to the application by following the learn.microsoft docs.
4. Validate that you can get JWTs according to the documentation on learn.microsoft.

   Note: The steps above require a scope value. To get it go to *App registration* on the Azure portal, click on *Manage -> Expose an API*, and at the top extract *Application ID URI*. If there is no URI you have to create one.
5. Validate the JWT from the previous step contains the desired roles. You can use jwt.io for this.
6. Determine the JWKS URL for your Azure instance. Open the *App registration* in the Azure portal and select *Overview -> Endpoints*. Copy the value from the field *OpenID Connect metadata document* (it should look similar to `https://login.microsoftonline.com/1df445a6-834b-4d55-a5ba-7a59d618bff8/v2.0/.well-known/openid-configuration`). Open the URL in the browser. This should provide you with a JSON file that contains the `jwks_uri` field. copy the value of this field (e.g., `https://login.microsoftonline.com/1df445a6-834b-4d55-a5ba-7a59d618bff8/discovery/v2.0/keys`). Use this URL to configure the appliance in the next step.
7. Configure the Anapaya appliance by adding the `jwks_uri` value extracted above into the `jwks_url` field under the `token_verification_keys` field. The configuration should look as follows:

```
{
  "oauth": {
    "enabled": true,
    "token_verification_keys": [
      {
        "id": "azure_keys",
        "jwks_url": "https://login.microsoftonline.com/1df445a6-834b-4d55-a5ba-
↪7a59d618bff8/discovery/v2.0/keys"
      }
    ]
  }
}
```

8. Deploy the appliance configuration. The appliance will now successfully validate the JWTs issued by Azure.

   Warning: Deploying a wrong configuration can potentially lock you out of the API. It is advised to temporarily also configure *basic authorization* as a fallback if the OAuth configuration doesn't work.

CHAPTER

TWENTYTHREE

BASIC TROUBLESHOOTING GUIDE

A network might start misfunctioning for a wide range of reasons reaching from hardware problems to software issues. Here, we give some basic guides on how you can troubleshoot some common network issues. We mainly focus on the connectivity failures caused by the misconfiguration of SCION services.

23.1 SCION Connectivity Issues

Issue: You are operating the SCION AS `1-ff00:1:1` and you have been notified that the connectivity (either SCION connectivity or the IP connectivity through an IP-in-SCION tunnel) from the host `EDGE-1` in your AS to the neighboring AS `1-ff00:1:2` is lost.

Note: In practice, you ideally should get informed about such an incident through your alerting system which sits on top of the monitoring system. Therefore, you might be able to extract infromation form the alerts which could lead you to the source of the issue. For this guide, we do not rely on such information as it is dependent on your monitoring and alerting systems.

Note: The steps taken here for troubleshooting should be perceived solely as recommendations. Furthermore, they are meant to assist you with resolving only a small subset of issues you might encounter in practice.

A reasonable first step is to log into `EDGE-1` and check the set of SCION paths to the AS `1-ff00:1:2`.

In *Not All Expected Paths Are Alive*, we consider the case where you do not see the full set of paths you expect and then explain two potential causes and how to resolve them.

In *All Expected Paths Are Alive*, we cover the scenario where all the expected paths are actually alive. We again discuss two possible causes and guide you how to resolve them.

23.1.1 Not All Expected Paths Are Alive

A basic sanity check for SCION connectivity-related issues is to log into `EDGE-1` in the AS `1-ff00:1:1` and run the showpaths command. This command shows us the set of available paths to a particular destination. With the `--refresh` flag, we can force the scion tool to grab fresh paths from the local SCION control service. The command to run showpaths towards the AS `1-ff00:1:2` is the following:

```
scion showpaths 1-ff00:1:2 --refresh
```

If there is no path, the output will look like this:

```
Available paths to 1-ff00:1:2
Error: no path found
```

It is also possible that you do not see the complete set of paths you expect or some of them are in the `timeout` state instead of `alive`.

For example, you expect to see the path `[1-ff00:1:1 2>3 1-ff00:1:2]` which corresponds to the link from interface 2 in `1-ff00:1:1` to interface 3 in `1-ff00:1:2`, but it is not present. A natural next step for troubleshooting here is to run an IP `ping` between `EDGE-1` and the corresponding router in the AS `1-ff00:1:2`. If this works, it means that there is connectivity on the IP underlay connecting `EDGE-1` and the router in `1-ff00:1:2`. In that case, we can guess that the connectivity issue is on the SCION level. Note that if the `ping` command is not working either, then it might be a misconfiguration of the underlay network or an issue with networking hardware. Covering such scenarios is out of scope for this document.

Considering the investigations from above, it seems safe to conclude that we have a SCION connectivity issue. We explain two possible reasons for this.

Scenario 1: Endpoint Misconfiguration

One potential cause is that there is an error in the configuration of `EDGE-1`. This is especially likely if you have just configured `EDGE-1`. Furthermore, if a non-empty subset of the paths is available, the AS certificate issue that we discuss in the next section can be ruled out on our side.

The issue could be simply caused by a typo in an IP address or a missing entry. In the example above, you need to check the configuration of interface 2 in your AS. If this is the problem, then naturally you should fix the misconfiguration, configure the appliance with the new configuration, and then check that you see the set of paths you expect.

Scenario 2: AS Certificate Issue

If there is no valid AS certificate configured on `EDGE-1`, the appliance cannot create valid path segments from the beacons because it cannot create signatures. As a result, the showpaths will not display any path. However, the `ping` commands to the AS `1-ff00:1:2` would work. Thus, the AS certificate might be the source of the problem.

You can see the list of AS certificates that are configured on the appliance by running the command:

```
curl -k https://${mgmt_address}/api/v1/cppki/certificates
```

Alternatively, to inspect the list in a browser, you can navigate to https://\protect\T1\textdollar\{mgmt_address\}/api/v1/cppki/certificates.

Note: `${mgmt_address}` needs to be replaced with the actual management address of the appliance.

If there is no AS certificate configured on `EDGE-1`, the output will look like:

```
{
   "certificate_chains": []
}
```

The cause for a missing AS certificate could be that we have just added `EDGE-1` and forgot to configure an AS certificate, its certificate has been deleted accidentally, or there was an issue with automatic renewal. Automatic renewal can fail, for example, when there has been a prolonged connectivity issue in the order of days.

Thus, to resolve the issue, you need to add a valid AS certificate to `EDGE-1`. In general, an AS certificate needs to be requested from one of the CAs of the local ISD. The initial certificate is requested with an out-of-band mechanism. Afterward, the AS certificates are periodically renewed by the appliance in a fully automatic fashion. See *Certificate/TRC Provisioning* for more details on listing, generating, and installing AS certificates.

23.1.2 All Expected Paths Are Alive

Scenario 1: Domain Misconfiguration

Assume that according to your SCION Gateway Routing Protocol (SGRP) setup, you are expecting that a `ping` command from the end host `Endhost-1` (in the AS `1-ff00:1:1`) to the end host `Endhost-2` (in the AS `1-ff00:1:2`) to work fine, but it does not. On the other hand, running a `showpaths` command towards the AS `1-ff00:1:2` actually displays all the paths you expect to see between these two ASes.

Note: See *Routing Domains* for details on the routing domain configuration.

To investigate this further, a reasonable next step is to check out the network prefixes advertised by the local SCION AS (i.e., `1-ff00:1:1`) and the prefixes learned from the remote SCION ASes (in particular, `1-ff00:1:2`).

These prefixes are exposed by the appliance as an HTTP status page. In fact, `gateway` is the process that configures and processes everything related to IP-in-SCION tunneling. Thus, it publishes the status page. To inspect that, you can run:

```
curl ${mgmt_address}/diagnostics/sgrp
```

Below is an example of how the output could look like:

```
{
   "advertise": {
         "static": [
            "10.8.0.2/32",
            "10.8.0.5/32"
         ],
         "dynamic": []
   },
   "learned": {
         "dynamic": []
   },
   "next-hops": {}
}
```

In this case, no prefix from remote ASes has been learned.

If in your setup, there is no prefix learned or more generally the set of learned and advertised prefixes does not match what you expect, then perhaps the domain is misconfigured. If this is the case, you naturally need to first fix the configuration, configure the appliance with the modified configuration, and then check the HTTP status page to confirm that the changes appear there too. Afterward, the `ping` command from `Endhost-1` to `Endhost-2` should start working.

Scenario 2: TRC Issue

In order for the appliance to join the SCION network and communicate with other nodes, it has to be configured with a set of TRCs. These TRCs build the trust anchors for verifying all of the control plane data that is exchanged in the SCION protocol. Therefore, the lack of a trusted TRC in the appliance will result in the loss of connectivity.

You can see the list of TRCs which are configured on the appliance by running the command:

```
curl -k https://${mgmt_address}/api/v1/cppki/trcs
```

Alternatively, to inspect the list in a browser, you can navigate to https://\protect\T1\textdollar\{mgmt_address\}/api/v1/cppki/trcs.

If there is no TRC configured on the appliance the output will be as follow:

```
{
   "trcs": []
}
```

This indicates that no TRC is configured on the appliance; thus in order to fix the issue, you need to install a valid TRC on this appliance. You can see *Certificate/TRC Provisioning* for more details on generating and installing a TRC.

The cause for a missing TRC could be that we have just added `EDGE-1` and forgotten to configure a TRC or its TRC has been deleted accidentally. Note that in the second case, the showpaths command might keep functioning correctly for some time.

CHAPTER

TWENTYFOUR

SETTING UP A MONITORING HOST

Our monitoring stack is based on Prometheus, Grafana, Loki and AlertManager. They are all open-source tools with plenty of documentation and support online.

In order to set up monitoring of Anapaya appliances, there are a few technical requirements.

1. The monitoring host must be able to reach the management interface of the target appliances.
2. Firewall rules must allow opening an HTTP(S) connection on the monitoring port of the appliance.

24.1 Enabling Telemetry in the Appliance

In order to collect telemetry data, you need to enable telemetry exporting in the appliance.

1. In the appliance configuration of the monitored host, verify that there is an entry for telemetry, as shown in the snippet below. The **address** determines the address where the appliance exposes the metrics and is in the format `ip:port`. By default, the telemetry address is `0.0.0.0:42001`. Prometheus is configured to scrape the metrics at this address.

```
{
  "management": {
    "telemetry": {
      "address": "<address>"
    }
  }
}
```

24.2 Setting up Prometheus

In order to set up Prometheus, follow the official Prometheus instructions. Specifically,

1. Ensure you have the latest version of Prometheus installed. Consult the installation guide for reference.
2. Follow the instructions in Section Configure Prometheus to monitor the sample targets.
   - To monitor each appliance, add the following snippet in the `prometheus.yml` file. For each host that will be monitored, add an entry in the `targets` section. The appliance address is the one configured in the telemetry section of the appliance configuration, as shown in *Enabling Telemetry in the Appliance*. It has the format `host:port`, where `host` can be either an IP address or a hostname.

```
- job_name: 'anapaya-appliance'
  honor_labels: true
  metric_relabel_configs: # Add this config if you are using the Anapaya␣
↪Grafana dashboards.
    - source_labels: ['hostname']
      target_label: 'shortname'
    - source_labels: ['__name__']
      regex: 'target_info'
      action: drop
  static_configs:
    - targets:
        - <appliance address>
      labels:
        product: <product>
```

Note: If you use the *recommended Grafana dashboards*, make sure you add the correct product label as the dashboards require this label to be set accordingly.

The available labels are: `core, edge, gate, ca`

3. Start Prometheus. The exact command depends on the method of installation.

24.2.1 Recording and Alerting Rules

Prometheus allows the configuration of rules for recording data or creating alerts when an event happens. These alerts can later be picked up by AlertManager and be integrated with your alerting system. You can specify the events that trigger an alert, the scope and severity of the alert, and also provide a description and summary of the firing alert. Below, we provide two examples of how to monitor the state of a service and the state of an interface.

```
- alert: SystemServiceDown
  expr: up == 0
  for: 1m

- alert: SCIONInterfaceStateDown
  expr: router_interface_up == 0
  for: 1m
```

You can find more information on which metrics can be scraped by Prometheus in *Telemetry*.

24.2.2 Recommended Alert Rules

Anapaya provides a recommended rule set for alerts that can be used as a starting point. For the published alerts we provide a *Troubleshooting & Runbooks* page in the event that one of the alerts is triggered. The files are accessible in Anapaya's software repository on cloudsmith.io. Depending on the product you are using, there is a predefined set of alert rules, i.e., `anapaya-alerts-core` for CORE, `anapaya-alerts-edge` for EDGE, `anapaya-alerts-gate` for GATE, and `anapaya-alerts-scion-ca` for the Anapaya SCION CA. There is also an `anapaya-alerts-external` package that includes all recommended alerts.

For example, to download the latest version of the EDGE alert rules, run:

```
curl -O https://dl.cloudsmith.io/<access_token>/anapaya/stable/raw/names/anapaya-alerts-
↪edge/versions/latest/anapaya-alerts-edge-latest.yml
```

The <access_token> is provided to you by Anapaya as part of your software license.

Adjust your Prometheus configuration to use the downloaded alert rule file. Follow the official instructions on adding an alert rule file to your Prometheus configuration.

24.3 Setting up Grafana

In order to set up Grafana, follow the official Grafana instructions. Before setting up Grafana, ensure that you have set up and started Prometheus following the instructions in *Setting up Prometheus*. On a high level, you need to:

1. Install Grafana.
2. Log in to Grafana.
3. Configure Prometheus as a datasource for Grafana.
4. Start building queries and dashboards. Grafonnet provides a way to programmatically create dashboards. An explanation of the library can be found in this blogpost.

24.3.1 Recommended Grafana Dashboards

Anapaya provides a recommended set of Grafana dashboards that can be used as a starting point. The archives including the JSON dashboards are accessible in Anapaya's software repository on cloudsmith.io. The JSON files can then be imported to Grafana. Depending on the product you are using, there is a predefined archive including multiple dashboards, i.e., `anapaya-dashboards-core` for CORE, `anapaya-dashboards-edge` for EDGE, `anapaya-dashboards-gate` for GATE, and `anapaya-dashboards-scion-ca` for the Anapaya SCION CA. There is also an `anapaya-dashboards-external` archive that includes all recommended dashboards.

For example, to download the latest version of the EDGE dashboards, run:

```
curl -O https://dl.cloudsmith.io/<access_token>/anapaya/stable/raw/names/anapaya-
↪dashboards-edge/versions/latest/anapaya-dashboards-edge-latest.zip
```

The <access_token> is provided to you by Anapaya as part of your software license.

In order to import a JSON dashboard, follow the official instructions.

24.4 Setting up Loki

24.4.1 Monitoring Host

In order to export the logs from each monitored host, we use Loki. Follow the official instructions on setting up Loki on the monitoring host.

Ensure that Loki is added as a data source, as explained in the instructions.

24.4.2 Monitored Hosts

On the hosts that you want to monitor, add the following snippet in the `management/telemetry` section of the appliance configuration. Make sure to replace the fields in <> with the actual data based on the host.

- **url** is the URL of the Loki instance. The appliance is configured to send the generated logs to that URL. The user is able to also access the logs on this URL.
- **address** was already configured in Section *Setting up Prometheus*.

```
{
  "management": {
    "telemetry": {
      "address": "<address>",
      "logging": {
        "logging_type": "LOKI",
        "loki": {
          "basic_auth": {
            "password": "<password>",
            "username": "<username>"
          },
          "url": "<url>"
        }
      }
    }
  }
}
```

24.5 Setting up AlertManager

In order to export the alerts from each monitored host, we use AlertManager. Follow the official instructions on setting up AlertManager. As stated in the official documentation, AlertManager can be integrated with a wide variety of alert systems, including emails, Slack, and Opsgenie.

CHAPTER

TWENTYFIVE

EXPLORING THE SCION NETWORK

The goal here is to introduce several tools which can be used to explore the SCION network. We explain the functionalities that the **scion** tool provides and talk about traffic interception using TShark.

25.1 Investigating SCION Paths

Here, we focus on one of the fundamental functionalities of the **scion** tool, namely the `scion showpaths` command, which lets network administrators explore the paths in the SCION network. Showpaths takes the destination AS as input, requests paths from the SCION control plane, and displays them in a human-readable format. The showpaths tool also supports command line arguments to display additional information about the path (e.g., path expiration time or path status). For example, the command below can be used to display all the paths to the AS with the ISD-AS number `1-ff00:1:1`:

```
scion showpaths 1-ff00:1:1
```

Here is an example of how the output would look like:

```
Available paths to 1-ff00:1:1
[0] Hops: [1-ff00:1:2 1>1 1-ff00:1:1] MTU: 1472 NextHop: 10.2.0.1:30042 Status: alive␣
↪LocalIP: 10.2.0.2
[1] Hops: [1-ff00:1:2 2>3 1-ff00:1:1] MTU: 1472 NextHop: 10.2.0.2:30042 Status: alive␣
↪LocalIP: 10.2.0.2
```

This indicates that there are two paths to the aforementioned AS. The paths are represented as a sequence of AS hops and interface pairs that are traversed. An interface pair is represented as **eg>in**, where **eg** is the egress interface ID, and **in** is the ingress interface ID. In the second path in the example above, a packet on the path exits the AS `1-ff00:1:2` on the egress interface 2 and enters the AS `1-ff00:1:1` on the ingress interface 3.

The Maximum Transmission Unit (MTU) and the next hop on this path are also displayed. The next hop indicates the internal address of the SCION router a packet has to be forwarded to when using this path.

Due to path caching, sometimes showpaths might show fewer paths than you expect. If that is the case, you can ask showpaths to fetch new paths using the `--refresh` flag.

You can exploit various functionalities of the showpaths command by using different flags. To see a complete list of the provided flags, you can run:

```
scion showpaths --help
```

For example, you can run the following command to see how long it takes until the paths to the AS `1-ff00:1:1` expire:

```
scion showpaths -e 1-ff00:1:1
```

By default, showpaths probes the paths it displays by sending probe packets across each of them and waiting for a response. A path is in one of the following three states:

- **Alive**: The response from the destination AS was received.
- **Timeout**: No response to the probe packet was received from the destination AS.
- **SCMP**: A SCION Control Message Protocol (SCMP) error was received in response to the probe packet. (You will learn more about SCMP in *Network checks using SCMP*.)

If this behavior is not desired, probing can be deactivated by providing the `--no-probe` flag.

25.2 Network checks using SCMP

The SCION Control Message Protocol (SCMP) is analogous to the Internet Control Message Protocol (ICMP) and provides the following functionalities:

- **Network diagnostic**: SCMP is used to implement network debugging tools such as the SCION equivalents of `ping` or `traceroute`.
- **Error messages**: SCMP is used by SCION applications (e.g., routers and dispatchers) to signal problems encountered during packet processing or to inform end hosts about network-layer problems.

The **scion** tool uses SCMP to gather information about the network. To see the command line arguments of **scion**, you can run:

```
scion help
```

Note: The **scion** tool provides two sub-commands that use SCMP to gather information: `traceroute` and `ping`.

- `traceroute` is similar to IP traceroute; it sends multiple SCMP packets and each packet is crafted so that a different router in the path replies.
- `ping` is similar to IP ping; it sends a specified number of packets at a given interval and prints out the round-trip time.

Here is an example `scion ping` command:

```
scion ping -c 1 1-ff00:1:1,[10.8.0.1]
```

This command sends pings toward the AS `1-ff00:1:1` and the host with the IP address `10.8.0.1`. Of course, the corresponding host needs to have a SCION network stack to be able to respond to SCION pings.

Furthermore, when the `-c` flag is set, `scion ping` sends the specified number of SCMP echo packets and reports back the statistics. You can familiarize yourself with different flags supported by the `scion ping` and `scion traceroute` commands by running `scion ping --help` and `scion traceroute --help`.

The `scion` tool gives you the possibility to select the path on which you want to execute your `ping` or `traceroute` command. For this, you need to utilize the `--interactive` flag. Here is an example:

```
scion ping --interactive 1-ff00:1:1,[10.8.0.1]
```

After executing such a command, you will be asked to choose your desired path.

In addition to the flag `--interactive`, the `scion` tool provides the `sequence` option which gives you even more flexibility in the choice of path for the execution of `ping` and `traceroute`. You can read about how it works by running `scion ping --help` or `scion traceroute --help`.

25.3 Traffic interception with TShark

TShark is a network protocol analyzer that can be used to inspect live network traffic (see Wireshark man page for more information about this tool).

To use TShark, you first need to make sure that it is installed on the machine where the TShark commands to capture traffic will be run. Then, you can simply run:

```
tshark
```

Then, TShark starts capturing packets from the default interface. To capture traffic from any interface, you can run:

```
tshark -i any
```

The output of the `tshark` command is always in the following format:

```
<seq-id> <timestamp> <src> -> <dst> <protocol> <protocol specific info>
```

The `seq-id` is an increasing ID that starts at 0 and increments by one for each captured packet. The `timestamp` indicates the time since starting the capture. The `src` and `dst` values represent source and destination IP addresses of the packet. The `protocol` represents the protocol in use; there could be packets with various protocols such as "HTTP", "ICMP", "TCP", "BFD", "UDP", etc. Finally, the last part of the capture line presents information that is specific to the protocol.

TShark supports various packet filtering mechanisms. We already mentioned one common filter, the `-i` flag. This flag can be used to select the desired interface(s). For example, the command `tshark -i eno5` will show only packets that go through the `eno5` interface.

More specific filters can be written in a custom packet filter language (see the wireshark wiki). For instance, to show all traffic that has IP destination address `192.168.7.2`, use the following command:

```
tshark -i any dst 192.168.7.2
```

Multiple filters can be combined with the `and` operator, for example to additionally filter for the port `42001`, one can run the following command:

```
tshark -i any dst 192.168.7.2 and port 42001
```

The default filtering is too limited to inspect the SCION traffic. Therefore, we provide a plugin that allows us to filter on the SCION layer. For example, the filters `scion.src_as` and `scion.dst_as` can be exploited to filter the packets according to their source and destination AS. The following command prints out only the packets whose destination AS is equal to `ff00:1:1`:

```
tshark -i any -Y 'scion.dst_as == "ff00:1:1"'
```

As mentioned above, there are packets with various protocols. We can also filter according to our desired protocols. For example, `scion.next_hdr != BFD` in the command below makes sure that no packet from the Bidirectional Forwarding Detection (BFD) protocol is shown:

```
tshark -i any -Y 'scion.next_hdr != "BFD"'
```

Or, for example, you can filter for the SCMP packets by running:

```
tshark -i any -Y 'scion.next_hdr == "SCMP"'
```

Tip: For more information about TShark SCION filters, see https://github.com/scionproto/scion/blob/master/tools/wireshark/scion.lua. Inspect the filters that were used above, and see what other filters can be applied.

Tip: You can use the `-V` flag in the `tshark` commands to print full packets.

CHAPTER

TWENTYSIX

SCION CPPKI CERTIFICATE AUTHORITY SERVICE

The SCION CPPKI requires some ASes to act as Certificate Authorities (CA) in their local ISD. These ASes need to provide a certificate renewal service to their customer ASes. The certificate renewal request is sent via SCION to the Anapaya appliance in the CA AS, and on successful certificate issuance, a valid SCION CPPKI Certificate Chain consisting of the SCION CPPKI AS and CA certificates is returned. This document describes the different deployment models the Anapaya appliance supports for such a CA AS, and goes into details on the configuration and operations of the standard solution.

26.1 Deployment Models

The Anapaya Appliance supports multiple deployment models for a SCION CPPKI Certificate Authority (CA). The different models are tailored to address different needs and requirements. In the rest of this section, these models are explained in more detail.

26.1.1 External SCION CPPKI Certificate Authority

In high-security networks, customers usually require specific policies on the management of the certificates and keys. These customers can provide their own implementation of a SCION CPPKI Certificate Authority that adheres to these policies. For example, a policy could be to store private keys for the SCION CPPKI CA certificate in an HSM, or to have strict identity checks before issuing an initial SCION CPPKI AS certificate.

The Anapaya appliance is capable of using an external SCION CPPKI Certificate Authority implementation via a REST HTTP API. For this deployment type, the Anapaya appliance needs to be configured with a shared-secret such that it can authenticate itself with the external CA service.

26.1.2 Vault-based SCION CPPKI Certificate Authority

The standard solution is to use HashiCorp Vault as the SCION CPPKI Certificate Authority backend. The Anapaya appliance internally has an adapter that exposes the REST HTTP API and translates them into calls to the Vault.

The required configuration of Vault is described in *Vault-based CA: Configuration*.

The following list defines terms that will be used extensively throughout the document:

- **Secrets Engine**: A secrets engine is a term defined by Vault that describes a component that stores, generates, or encrypts data. They are very flexible and define most of Vault's capabilities. Secrets engines are enabled at a "path" (think URL path). Multiple instances of the same secrets engine can be mounted at different paths. This allows us to mount separate PKI engine instances per SCION AS.

  For more details, see https://www.vaultproject.io/docs/secrets.

- **PKI Engine**: Vault defines a secrets engine for handling X.509 certificates called PKI. To avoid confusion, this secrets engine is called *PKI engine* for the rest of this document.

  For more details, see https://www.vaultproject.io/docs/secrets/pki.

- **Policy**: Various degrees of access to secrets and secrets engine functionality can be defined for individual users and/or applications. The rules for access control are defined in policies.

  For more details, see https://www.vaultproject.io/docs/concepts/policies.

The following diagram highlights the interactions between different Vault clients and the respective Vault endpoints.

- **Certificate Authority Adapter**

  The Certificate Authority (CA) adapter is part of the Anapaya Appliance. It only interacts with a subset of the available secrets engine endpoints. The `sign-verbatim` endpoint is used by the CA adapter to service the certificate renewal requests. This endpoint issues certificates as specified in a CSR. All the validation steps are done in the CA adapter itself.

  To simplify the storage requirements, and ensure consistency between different Anapaya appliances, Vault acts as the storage backend for the necessary TRCs, as well as the blocklist that prevents specific SCION ASes from renewing their certificates.

  The CA adapter requires a token to interact with Vault. Tokens in Vault always have a TTL and need to be renewed periodically, the CA adapter takes care of this internally. It authenticates itself periodically with Vault via the AppRole authentication method and fetches the token. The secrets for this need to be provisioned in the Anapaya appliance configuration.

- **Certificate Authority Provisioning**

  The Vault needs to be provisioned with the right data in order for the SCION AS to provide the correct SCION CPPKI CA functionality. Because of the short lifetime of the SCION CPPKI CA certificates (recommended is 11 days), this task needs to be automated and executed periodically.

- **Manual Intervention**

  Manual intervention is only necessary to adapt the blocklist and to provision new TRCs after a TRC update.

26.1.3 In-process SCION CPPKI Certificate Authority

For testing purposes, the Anapaya appliance supports renewing SCION CPPKI AS certificates in-process. In this mode, no external CA service or Vault is required. However, a valid CA private key and the corresponding SCION CPPKI CA certificate are required and must be stored on the appliance itself.

26.2 Vault-based CA: Configuration

In this section, we describe the necessary configuration of HashiCorp Vault such that it can be used as a SCION CPPKI Certificate Authority backend by the Anapaya appliance. We also list deployment considerations that an operator should be aware of.

26.2.1 Vault cluster

The Vault cluster is an integral part of the CA architecture. It should be deployed with support for individual Vault node failure. This can be achieved by deploying multiple Vault instances in a cluster. (see: High Availability Mode)

Vault can be run on top of various different databases that provide the storage layer. However, for simplicity, we suggest that Vault is run using the integrated storage.

26.2.2 Authentication Setup

The Anapaya appliance (via the CA adapter) needs to authenticate itself with Vault to gain access to the target endpoints. For this, the AppRole authentication method must be enabled, and the appropriate roles need to be provisioned. Each CA adapter should get its own individual identity and `role_id`, such that access to individual Anapaya appliances can be revoked without affecting the remaining appliances.

Each CA adapter should get the right policy assigned, such that it can access the correct Vault *PKI Engine* and the *Key/Value Engine*.

To take advantage of the policies that are listed in *Policies*, the Vault identity for the CA adapter needs to have the `isd_as` with the ISD-AS number in its metadata. To do this, execute the following steps:

1. Create a Vault identity for the CA adapter, one identity per adapter instance.

   The suggested name pattern is of the form `ca-adapter_<isd-as>-<instance-id>`. The metadata should include the key `isd_as` with the ISD-AS number as the value and the identity should have the *CA Adapter Policy* attached to it. Use the /identity/entity endpoint to create the identities.

2. Create a Vault AppRole for each CA adapter instance.

   The suggested name pattern is `ca-adapter_<isd-as>-<instance-id>`. Use the /auth/approle/role/:role_name endpoint.

3. Create a Vault entity alias that links the AppRole and the identity together.

   Create an alias so that once you log in with the AppRole you are automatically assigned the entity. For this, you will need the following information:

   - **role ID** of the CA adapter. Get this with the /auth/approle/role/:role_name/role-id endpoint. (`data.role_id` in the response body)
   - **identity ID** of the CA adapter identity. Get this with the /identity/entity/name/:name endpoint. (`data.id` in the response body)
   - **accessor ID** of the AppRole authentication engine. Get this with the /sys/auth endpoint. (`.["approle/"].accessor` in the response body)

   Then, link the AppRole to the identity by creating an alias using the /identity/entity-alias endpoint. The `name` is set to the role ID, the `canonical_id` is set to the identity ID, and the `mount_accessor` is set to the accessor ID.

4. Extract `role_id` and `secret_id` of the created AppRole(s) and distribute them to provision the Anapaya appliance where the instance of the CA adapter should run. To read the `secret_id`, you can use the /auth/approle/role/:role_name/secret-id endpoint.

26.2.3 Secrets Engine Setup

In this section, we describe the different Vault Secrets Engines that are required for the Vault to act as a CA backend.

PKI Engine

Docs: https://www.vaultproject.io/docs/secrets/pki

The PKI engine is the core of the CA backend. It stores the currently active SCION CPPKI CA certificate and the associated private key, and is used to issue the renewed certificates.

Create one active PKI engine per SCION CA AS that is backed by the Vault backend. Each CA adapter only has access to the PKI engine for their respective ISD-AS. The currently active PKI engine is mounted at `/ca/<isd-as>/pki`. For example, the PKI engine for AS `1-ff00:0:110` is mounted at `/ca/1-ff00:0:110/pki`. The CA adapter will use the `/ca/<isd-as>/sign-verbatim` endpoint to request the certificate issuance based on the CSR that it received from the customer AS.

We suggest that each PKI engine is only ever associated with exactly one CA certificate. This allows for a clean history of what SCION CPPKI AS certificates were issued with any given SCION CPPKI CA certificate. To achieve this, we recommend the *Rolling CA Provision Strategy* that is described below.

Key/Value Engine

Docs: https://www.vaultproject.io/docs/secrets/kv/kv-v2

The key/value engine is used to keep the state for our CA adapter. For verification of the request, the CA adapter needs the currently active TRC (and potentially the TRC that is in the grace period currently). Furthermore, the CA adapter needs access to the blocklist that allows operators to manually block certain ASes, and prevent them from renewing their certificate.

Again, mount one instance per SCION CPPKI CA AS at `/ca/<isd-as>/kv`. For example, the key/value engine for AS `1-ff00:0:110` is mounted at `/ca/1-ff00:0:110/kv`. When enabling the Vault storage mount at `ca/<isd-as>/kv`, version 2 must be selected.

Note: We require version 2 of the key/value engine in vault. Follow the vault setup documentation and ensure you are using the right version.

- `/ca/<isd-as>/kv/data/trcs` hosts the TRCs for the local ISD.

  The TRCs are keyed by the following pattern `ISD<isd>-B<base>-S<serial>`. The value must be a PEM encoded TRC. For example, the TRC ISD1-B1-S1 is expected to be located at `/ca/<isd-as>/kv/data/trcs/ISD1-B1-S1`. Additionally, the latest TRC must be located at `/ca/<isd-as>/kv/data/trcs/latest`.

  The CA adapter uses the TRCs to verify the signatures on the CSRs that are received from the customer AS. It will first look up the latest TRC. If the latest TRC implies that the predecessor TRC is still in the grace period, it will additionally look up that predecessor TRC at the keyed path.

- `/ca/<isd-as>/kv/data/blocklist` hosts the blocklist.

  For each AS that is on the blocklist, an entry in the KV story is required. The data of the entry can be empty, the presence is enough to block an AS. For example to put `1-ff00:0:111` into the blocklist, it suffices to post on the path `ca/<isd-as>/kv/data/blocklist/1-ff00:0:111`. To check if `1-ff00:0:111` is on the blocklist, read from the same path. If the status code is `404` the item is not on the blocklist, if the status code is `200` the item is blocked.

  The CA adapter uses this blocklist every time it gets a CSR to decide whether a certificate should be issued, or whether the request should be denied.

26.2.4 Policies

Each role gets its individual policy. Each policy should grant the role the minimal set of capabilities for proper operation.

CA Adapter Policy

The CA adapter should only get access to the endpoints belonging to **its** own ISD-AS. We can leverage `identity.entity.metadata.isd_as` to ensure this. The metadata is set as part of the AppRole creation for the CA adapter. The CA adapter should get the following policy associated with its role:

```
# The sign-verbatim endpoint is required to sign new AS certificates.
path "ca/{{identity.entity.metadata.isd_as}}/pki/sign-verbatim" {
    capabilities = ["create", "update"]
}

# TRCs are required in the CA adapter to verify requests.
path "ca/{{identity.entity.metadata.isd_as}}/kv/data/trcs/+" {
    capabilities = ["read"]
}

# Blocklist is required in the CA adapter to validate requests.
path "ca/{{identity.entity.metadata.isd_as}}/kv/data/blocklist/+" {
    capabilities = ["read"]
}

# The CA adapter needs to login via approle to get a token for the other endpoints.
path "auth/approle/login" {
    capabilities = ["create", "update"]
}
```

CA Provisioning Policy

The exact policy for the CA provisioning is dependent on its implementation. To support the *Rolling CA Provision Strategy*, the following should be part of the policy:

```
# The CA provision task needs to be able to configure the CA certificate.
path "ca/+/+/config/ca" {
    capabilities = ["create", "update"]
}

# The CA provision task needs to be able to create new mounts.
path "sys/mounts/ca/+/+" {
    capabilities = ["create", "update", "delete"]
}

# The CA provision task needs to be able to remount old CAs.
path "sys/remount" {
    capabilities = ["create", "update", "sudo"]
    allowed_parameters = {
        "from" = ["ca/*"]
        "to" = ["ca/*"]
    }
```

(continues on next page)

(continued from previous page)

```
}

# The CA provision task needs to be able to tune the newly create PKI engine
path "sys/mounts/ca/+/+/tune" {
    capabilities = ["create", "update"]
    allowed_parameters = {
        "default_lease_ttl" = []
        "max_lease_ttl" = []
        "force_no_cache" = [false]
        "options" = []
    }
}
```

Manual intervention

The exact policy for manual intervention depends on the capabilities an operator should have in order to intervene. The following is a suggestion with a potentially useful capability set for an operator.

```
# Provision, list and read TRCs.
patch "ca/+/kv/data/trcs/+" {
    capabilities = ["read", "create", "update"]
}

# Add, list, and delete entities from the blocklist.
patch "ca/+/kv/data/blocklist/+" {
    capabilities = ["read", "create", "update", "delete"]
}

# Manually sign certificates.
path "ca/+/pki/sign-verbatim" {
    capabilities = ["create", "update"]
}

# Manually provision CA certificate.
path "ca/+/pki/config/ca" {
    capabilities = ["create", "update"]
}

# Manually run garbage collection
path "ca/+/pki/tidy" {
    capabilities = ["create", "update"]
}
```

26.3 Vault-based CA: Operations

The following section describes common tasks that are done while operating a Vault-based CA backend.

26.3.1 Upload a new TRC

To add a new TRC to the *Key/Value Engine*, you need to use the /ca/<isd-as>/kv/data/trcs/ISD<isd>-B<base>-S<serial> endpoint. The TRC must be PEM encoded.

Note: Make sure that the TRC is PEM encoded. The raw ASN.1 DER encoding is not supported. You can format any TRC to PEM encoding with the following command:

```
scion-pki trc format <TRC-file>
```

For example, to upload the TRC `ISD1-B1-S4`, you could use the following curl command:

```
curl \
  --header "X-Vault-Token: $VAULT_TOKEN" \
  --request POST \
  --data '{"data": {"trc": "-----BEGIN TRC-----\nMIIJ..."}}' \
  https://<vault-addr>/v1/ca/1-ff00:0:110/kv/data/trcs/ISD1-B1-S4
```

If the uploaded TRC is the new latest TRC, you also need to upload it to /ca/<isd-as>/kv/data/trcs/latest. This could be done with the following curl command:

```
curl \
  --header "X-Vault-Token: $VAULT_TOKEN" \
  --request POST \
  --data '{"data": {"trc": "-----BEGIN TRC-----\nMIIJ..."}}' \
  https://<vault-addr>/v1/ca/1-ff00:0:110/kv/data/trcs/latest
```

Note: In order for the SCION CPPKI AS certificate to be verifiable, the Anapaya appliance should also be provisioned with this TRC. Furthermore, the CA certificate potentially has to be rotated if the issuing Root Certificate has changed in the TRC update.

26.3.2 Add an AS to the blocklist

To add an AS to the blocklist, you need to create an entry in the *Key/Value Engine* with the /ca/<ca-isd-as>/kv/data/blocklist/<blocked-isd-as> endpoint. For example, to block ISD-AS `1-ff00:0:112` in the CA `1-ff00:0:110`, you could use the following curl command:

```
curl \
  --header "X-Vault-Token: $VAULT_TOKEN" \
  --request POST \
  --data '{"data": {"blocked": true}}' \
  https://<vault-addr>/v1/ca/1-ff00:0:110/kv/data/blocklist/1-ff00:0:112
```

Note: The `"blocked": true` is for informational purposes only. The existence of the entry blocks the `1-ff00:0:112` from getting a renewed SCION CPPKI AS certificate. You are free to add any additional informational metadata.

26.3.3 Remove an AS from the blocklist

To remove an AS from the blocklist, you need to delete the entry in the *Key/Value Engine* with the /ca/<ca-isd-as>/kv/metadata/blocklist/<blocked-isd-as> endpoint. For example, to remove the ISD-AS `1-ff00:0:112` in the CA `1-ff00:0:110` from the blocklist, you could use the following curl command:

```
curl \
  --header "X-Vault-Token: $VAULT_TOKEN" \
  --request DELETE \
  https://<vault-addr>/v1/ca/1-ff00:0:110/kv/metadata/blocklist/1-ff00:0:112
```

26.3.4 Rolling CA Provision Strategy

Vault is designed such that only one CA certificate is used per PKI engine. For example, it is not possible to query what CA certificate was used to issue an AS certificate via the /pki/cert endpoint. Together with the short lifetime of the CA certificates, this severely limits the observability and auditability of the system.

To remedy this problem, we suggest a rolling CA certificate provisioning strategy. In this strategy, you do not reconfigure the PKI engine with the updated CA certificate and private key. Instead, the previous PKI engine is archived, and a new PKI engine with the new crypto material is created. This must be transparent to our CA adapter to simplify configuration and business logic. You can achieve this with a sequence of /sys/remount operations:

1. Create the new PKI engine at `ca/<isd-as>/pki-next` and provision it with the fresh CA certificate and private key.
2. Archive the current PKI engine by remounting `ca/<isd-as>/pki` to `ca/<isd-as>/pki_<timestamp>`, where the timestamp is the current date in *yyyymmddHHMMSS* format. E.g., `ca/1-ff00:0:110/pki_20210624090000`.
3. Remount the new PKI engine from `ca/<isd-as>/pki-next` to `ca/<isd-as>/pki`.

This process can be scripted, and thus should cause a very minimal downtime. The advantage here is, that the CA adapter does not need to decide which PKI engine instance to talk to. It is always reachable under the same path at `ca/<isd-as>/pki`.

Note: There is a practical limit on the number of mount points of ~14000. If you rotate the CA certificate every 14 days and run a single CA in Vault, you run out of mount points after more than 250 years. If you run more CAs in the same Vault cluster, this number decreases.

https://www.vaultproject.io/docs/internals/limits#mount-point-limits

The current amount of mounts can be monitored via the /sys/mounts endpoint. Before you exhaust the limits, you can drop old PKI engines that are no longer needed for auditing purposes.

CHAPTER

TWENTYSEVEN

AS NUMBERS AND CERTIFICATES

In this user guide, we explain how to request an AS number and AS certificate. They are necessary for connecting an Anapaya appliance to the SCION network.

27.1 AS Number

Every participant in a SCION network needs their own SCION AS number. Refer to ISD-and-AS-numbering for more details on the numbering scheme.

Currently, Anapaya is the only organization assigning SCION AS numbers for the productive SCION networks. For the SSFN (Secure Swiss Finance Network), SIX is assigning SCION AS numbers on behalf of Anapaya.

Note: The entire 4-byte BGP AS numbering space is reserved for organizations that own the respective BGP AS number. Thus, if your organization already has a BGP AS number assigned by an official BGP numbering authority, it can claim the same SCION AS number.

27.1.1 Request AS Number from Anapaya

Reach out to the Anapaya CSE team to request a SCION AS number from Anapaya. Your request should include

- your organization details,
- your technical contacts responsible for SCION,
- and the *Isolation domains* you want to connect to.

Note: Once assigned, the AS number and your organization will be added to the *Autonomous Systems* list.

27.1.2 Request AS Number from SIX (for SSFN participants)

Follow the instructions on the SIX SSFN website on how to apply for a SSFN certificate. In the process, when required to fill in `AS Number`, leave the field empty and SIX will assign an AS number.

27.2 AS Certificate

A SCION AS needs a SCION AS certificate for every isolation domain it should be part of. Requesting an initial AS certificate is a manual process which is described below. Once the AS certificate is uploaded to the Anapaya appliance, SCION connectivity to the rest of the network can be established. AS certificate renewals are then automated and are performed by the appliance every few days.

27.2.1 Create Certificate Signing Request

As a first step, the Certificate Signing Request (CSR) needs to be created. Refer to *Generating a Certificate Signing Request* for details on how to use the appliance API for this. Make sure you include the relevant subject details.

Note: When creating a CSR for a SCION AS in the SSFN, make sure the values match with the contract values agreed upon with SIX.

27.2.2 Request the AS Certificate

- In case the CSR was created for a public isolation domain, send the CSR to the Anapaya CSE team.
- In case the CSR was created for the SSFN, send the CSR to the SSFN certificate team at SIX.

27.2.3 Install the AS Certificate

Once you receive the AS certificate from the issuing party, you need to install it in the Anapaya appliance. Refer to *Installing AS Certificates* for the necessary commands.

CHAPTER

TWENTYEIGHT

INITIALLY PROVISIONING AN ANAPAYA APPLIANCE

This guide provides information on how to provision an Anapaya appliance for the first time. This guide is targeted at the operators of the appliances. For further information on common EDGE deployment setups and sample configurations, please refer to *Anapaya EDGE*.

28.1 EDGE Initial Provisioning

In the following process, the assumption is that the Sales team from the customer or partner signs the appropriate order forms with Anapaya and returns them to Anapaya's sales team.

28.1.1 Configuration preparation

1. Define the WAN configuration details. Usually, the WAN details are defined by the ISP providing the SCION access.
2. Collect the LAN configuration details. The party which uses the Anapaya EDGE is free to choose the LAN parameters.

28.1.2 EDGE Installation on Customer Site

1. Install the EDGE at the customer location and power on the device.
2. Depending on the internal processes of the organization, the operator might also need to whitelist some SSH keys to get SSH access later.
3. Confirm that the host is accessible through SSH from the operator's network.

Note: In case the EDGE is part of a managed service by the ISP and should be remotely configured, the field technician installing the EDGE might need to manually configure the WAN IP and a static route toward the ISP's network equipment. For more information, please refer to *Connecting to the Appliance*.

28.1.3 Initial Network Configuration

1. Create the initial appliance configuration for the EDGE including the *Network Interfaces*, *System*, *Management* sections.
2. Push the initial appliance configuration to the host following *Applying a new Configuration*.
3. Confirm that there is still SSH access to the host.
4. Upgrade the EDGE to the latest software version following *Updating an Appliance*.

28.1.4 Full SCION Configuration

1. Prepare the appliance configuration for the EDGE including the *SCION* section.
2. Deploy the configuration updates to the EDGE following *Applying a new Configuration*.
3. Check if the SCION interface is up following *Testing a SCION Link*.

Note: For the SCION interface to be up, both the EDGE and the ISP's CORE appliances need to be configured. If the SCION interface is not coming up, check the EDGE configuration for potential errors. If the issue persists, contact the connecting ISP and verify that the CORE side has also been configured correctly.

---

4. Once the configuration details for IP-in-SCION tunneling are clarified, prepare the full appliance configuration with the *IP-in-SCION Tunneling* section.
5. Deploy the configuration updates to the EDGE following *Applying a new Configuration*.

Note: Depending on how fast all the necessary configuration parameters are clarified, this and the previous section can be combined and a full appliance configuration can be pushed to the EDGE appliance from the beginning.

28.1.5 Acquiring an AS Certificate

At this point, the customer needs to acquire an AS certificate for the new EDGE.

1. Create the Certificate Signing Request (CSR) following *Create Certificate Signing Request*.
2. Send the CSR to the organization running the Certificate Authority for the ISD. For more information on the governing entities of the ISDs, please refer to *ISD and AS Assignments* and *Joining a SCION Network*.

Note: For certain ISDs, such as SSFN, the CSR needs to be sent by the party which is legally part of the network (i.e. the end customer). In other cases, such as the public Swiss ISD, the CSR can be sent by either the owner of the EDGE or a managed service operator.

---

3. Once the AS certificate is issued, push it to the host following *Install the AS Certificate*.

CHAPTER

TWENTYNINE

PERFORMANCE OPTIMIZATIONS

The following sections describe how to optimize the performance of the appliance. For now, we only provide recommendations for the dataplane because the dataplane is the most performance-critical part of the appliance as it is responsible for forwarding packets. The dataplane is built on top of VPP and mainly uses DPDK for packet I/O. Therefore, general performance optimizations for VPP and DPDK can often also be applied to the appliance as well.

29.1 System

29.1.1 CPU

The appliance is designed to run on a multi-core CPU with different cores assigned to different tasks. We recommend that core 0 be assigned to Linux and control plane services. Core 1 should be used as the VPP's main core, which handles management functions (CLI, API, stats collection). The rest of the cores should be used as the VPP's worker cores, which perform the packet processing.

These configuration options can be set under the `config.system.vpp.cpu` section of the configuration file. Assuming that the appliance is running on a 4-core CPU, the following configuration is recommended:

```
{
    "system": {
        "vpp": {
            "cpu": {
                "main_core": 1,
                "workers": 2
            }
        }
    }
}
```

If the system only has two cores, we recommend that core 0 is used for Linux and core 1 for VPP. The work performed usually performed by the worker cores is then performed by VPP's main core. This configuration can be achieved by setting the number of workers to 0 (note that the default value is 1):

```
{
    "system": {
        "vpp": {
            "cpu": {
                "main_core": 1,
                "workers": 0
            }
```

(continues on next page)

(continued from previous page)

```
        }
    }
}
```

It is also possible to run the appliance on a single-core CPU and have VPP share the core with Linux.

Note: In case the packet processing is not performed on a separate core, the performance of the appliance will be significantly reduced. In particular, jitter and latency will be negatively affected.

29.1.2 Hugepages

The dataplane makes use of hugepages (see Linux Kernel documentation) for the packet buffers. Hence, it is important that there are enough hugepages available. The size of the hugepages and number of hugepages allocated by the appliance can be configured in the `config.system.kernel` section of the appliance configuration:

```
{
    "system": {
        "kernel": {
            "hugepage_size": "2M",
            "hugepages": 256
        }
    }
}
```

By default, 256 2MB hugepages are allocated. This is sufficient for most deployments. If there is a large number of fast interfaces (e.g., 25 Gbps), it might be necessary to increase the number of hugepages.

Note: If there are not enough hugepages available, the dataplane will not start or log an error message.

Note: Currently, the appliance only supports 2MB hugepages.

29.1.3 Buffers

By default, the appliance uses a fixed portion of the hugepages for packet buffers. In case more buffers are required, the number of buffers can be increased by increasing the number of hugepages. This is only needed in case there is a large number of fast interfaces (e.g., 25 Gbps).

Note: The number of buffers currently cannot be independently configured through the appliance API. This will be added in a future release.

29.1.4 RX/TX Queues and Descriptors

By default, the appliance configures each interface with one 1 RX queue per worker core. The default RSS hash function over the 5 tuple of the packet is used to distribute the incoming traffic among the workers.

Note: The number of RX/TX queues cannot be independently configured through the appliance API. This will be added in a future release.

The number of descriptors per RX/TX queue is by default set to 1024, which is sufficient for most deployments. If there is a large number of fast interfaces (e.g., 25 Gbps), it might be necessary to increase the number of descriptors. We recommend 2048 descriptors for a 25 Gbps interface and 4096 descriptors for a 100 Gbps interface.

Check out the VPP documentation for more information on how to configure RX/TX queues.

Note: The number of queues and descriptors currently cannot be independently configured through the appliance API. This will be added in a future release.

29.2 SCION

29.2.1 SCION RSS

With SCION RSS, our appliances are capable of utilizing the full potential of the available computational resources. The SCION RSS feature significantly enhances the throughput on multi-core systems by enabling receive side scaling (RSS) for SCION traffic. SCION RSS operates by leveraging source port entropy on the UDP underlay. For the feature to work properly, both sides of a link must support it, i.e., must be running at least release v0.34. In the following, we describe how to configure SCION RSS on links between *neighbor appliances* and *sibling appliances*.

Note: A **neighbor appliance** is an appliance that is directly connected to the appliance but located in a different AS.

A **sibling appliance** is an appliance that is located in the same AS as the appliance.

Traffic between Neighbor Appliances

To enable SCION RSS for traffic forwarded to a particular neighbor, enable it on all interfaces connected to that neighbor. This can be done by setting the `enable_scion_rss` option to `true` in the `config.scion.ases.neighbors.interfaces` sections:

```
{
    "scion": {
        "ases": [
            {
                "neighbors": [
                    {
                        "interfaces": [
                            {
                                "interface_id": 1,
                                "address": "[fd02:e8a2:c9e2:03e6::2]:30100",
                                "remote": {
```

(continues on next page)

(continued from previous page)

```
                                    "address": "[fd02:e8a2:c9e2:03e6::1]:30100",
                                    "interface_id": 201
                                }
                                "enable_scion_rss": true
                            }
                        ]
                    }
                ]
            }
        ]
```

Warning: Only enable SCION RSS for neighbor appliances that support it, i.e., that are running at least release v0.34. This usually requires some information exchange with the operator of the neighbor appliance.

Traffic between Sibling Appliances

With topology synchronization, the appliance automatically detects sibling appliances that support SCION RSS and enables the feature accordingly.

Without topology synchronization, it can statically be defined whether a sibling appliance supports SCION RSS or not, and the feature will be enabled accordingly. This can be done by setting `scion_rss` to true or false in the `config.cluster.peers.features` section:

```
{
    "cluster": {
        "peers": [
            {
                "features": {
                    "scion_rss": true
                }
            }
        ]
    }
}
```

Warning: Only enable SCION RSS for sibling appliances that support it, i.e., that are running at least release v0.34.

By default, SCION RSS is not enabled for traffic sent from a gateway to a sibling router on a different appliance. To enable SCION RSS set `scion_tunneling.endpoint.enable_scion_rss = true`.

Warning: Only enable SCION RSS for sibling appliances that support it, i.e., that are running at least release v0.34.

CHAPTER

THIRTY

RELEASE NOTES

To receive the most up-to-date release notes by email, sign up for our release notes newsletter.

30.1 Appliance Release v0.35

This page contains the release notes for the v0.35 Anapaya appliance software release. The appliance software release is applicable for the following Anapaya products:

- Anapaya CORE
- Anapaya EDGE
- Anapaya GATE

We recommend always upgrading to the latest available patch release. Please refer to **Upgrade Notes** (if any) of each release if there are any special steps to be taken when upgrading. For general information on how to upgrade your appliance, please refer to *Updating an Appliance*.

Known Issues

- There is a bug in version v0.35.0 that prevents the appliance from configuring the system kernel settings. This can result in the dataplane process not starting up properly. This is fixed in v0.35.1.
- In version v0.35.0 metrics of the IP-in-SCION tunneling service were sometimes incomplete. This is fixed in v0.35.1.
- Starting with version v0.35.0, for Mellanox Connectx-5/6 interfaces (including VFs), custom interface names are not supported anymore. If you have custom interface names for Mellanox Connectx-5/6 interfaces, you need to rename them to the default interface names before upgrading to v0.35.0.

  Often the name of a Mellanox Connectx-5/6 interface is `ensXnp0` where `X` is a number. The default name for a VF is `ensXvY` where `X` is the number of the physical interface and `Y` is the number of the VF.

  You can check the list of interfaces with their current names and description using `lshw -class network -businfo`.
- Appliance configurations that contain IP prefixes in non-canonical form can lead to IP-in-SCION tunneling service crashes in releases prior to v0.35.9 or on releases from v0.36.0 to v0.36.3.

  On releases v0.36.8 and older, the following accept filter could lead to a crash of a software component and loss of IP-in-SCION tunneling connectivity:

```
"accept_filter": [
    {
      "action": "ACCEPT",
```

(continues on next page)

(continued from previous page)

```
      "prefixes": [
        "198.51.100.210/28"
      ],
      "sequence_id": 0
    }
  ],
]
```

Instead, the accept filter should be expressed in its canonical form:

```
"accept_filter": [
    {
      "action": "ACCEPT",
      "prefixes": [
        "198.51.100.208/28"
      ],
      "sequence_id": 0
    }
  ],
]
```

30.1.1 Upgrade Notes

Warning: Release v0.34.0 and newer require Ubuntu 22.04 as the system and at least the anapaya-system package with version 2.8.0. Please refer to *Upgrade Notes* for more information.

Warning: In release v0.35.9, the appliance controller has been extended with additional validation rules. If you are upgrading from a previous verison and your appliance configuration contains IP prefixes in non-canonical form, the appliance controller will start and wait for the configuration to be fixed.

To ensure that your appliance is healthy, check the appliance controller logs after the update:

```
journalctl -u appliance-controller.service -e
```

Verify that the logs do not contain any log events with the following message:

```
"validating latest configuration","cause":{"msg":"invalid configuration"...
```

30.1.2 v0.35.9 (2024-07-19)

Fixes

- The appliance controller now requires all IP prefixes to be provided in canonical form. In prior releases, a non-canonical IP prefix in the accept filter could lead to crashing of the IP-in-SCION tunneling service.

30.1.3 v0.35.8 (2024-06-19)

- Fix a bug in the IP-in-SCION tunneling path monitoring that could lead to incorrect path statistics and only a subset of desirable paths being available

30.1.4 v0.35.7 (2024-06-14)

Improvements

- The IP-in-SCION tunneling compmonent now monitors maximum 100 (configurable) paths per remote AS. This is a measure to prevent high load on topologies with very high path diversity.

Fixes

- Fix an issue when monitoring a high number of paths in the IP-in-SCION tunneling component. Previously, path probes could be dropped due to an internal queue not having enough space.

30.1.5 v0.35.6 (2024-05-16)

Improvements

- Reduce resource consumption of the IP-in-SCION tunneling gateway.

30.1.6 v0.35.5 (2024-05-10)

Fixes

- Improve failover of IP-in-SCION tunneling under certain setups.

30.1.7 v0.35.4 (2024-03-08)

Fixes

- Fix setup of explicit MAC addresses on VFs on top of i40e NICs.
- Fix race condition in setup of Mellanox 5 NICs.
- Fix a rare race condition in the appliance controller if interfaces have multiple addresses configured.

30.1.8 v0.35.3 (2024-02-19)

Fixes

The gateway uses less SIG streams for the IP-in-SCION tunneling service. This fixes a compatibility issue with releases older than v0.34, that could lead to traffic being dropped under certain circumstances.

30.1.9 v0.35.2 (2024-02-14)

Improvements

VRRP can now be configured to run in unicast mode. In the unicast mode, the VRRP packets are sent to all physical routers in the VRRP group using the unicast addresses. By default, VRRP uses the multicast mode where the VRRP packets are broadcasted to all routers using the multicast address. The unicast mode is useful when something in the network does not support the standard VRRP multicast mode.

The unicast mode can be enabled by providing the unicast addresses of the routers in the VRRP group. The unicast addresses can be provided using the `peers` option in the VRRP configuration.

The following example shows how to configure unicast VRRP on an ethernet interface:

```
{
  "interfaces": {
    "ethernets": [
      {
        "name": "eno7",
        "vrrp": [
          {
            "vrid": 1,
            "addresses": [
              "10.100.1.1"
            ],
            "priority": 100,
            "peers": [
              "10.2.1.2",
              "10.2.1.3"
            ]
          }
        ]
      },
    ]
  }
}
```

Fixes

- Static peer appliance SCION interfaces are now correctly generated in auto-allowed mode. Previously, this was not always the case and this may have led to subtle problems in certain cases. In particular, the discovery service announced the gateways without allowed interfaces, which lead to allowed interfaces not being respected. Instead, all SCION interfaces were treated as allowed.
- The appliance now correctly configures the telemetry listen endpoint if no IP is specified.
- The dataplane is now always stopped before the appliance removes virtual functions (VFs). This prevents the appliance getting stuck in a state where it cannot remove VFs because the dataplane is still using them.

30.1.10 v0.35.1 (2024-01-09)

Fixes

- Fix a bug that was introduced in v0.35.0 that caused kernel configurations to not be applied.
- Fix inconsistent metrics in the IP-in-SCION tunneling service.
- Fix the IP-in-SCION tunneling service not properly updating remote gateway monitors when the remote control address or the allowed interfaces change.
- Omit unnecessary reconfigurations of the prefix server in the IP-in-SCION tunneling service.

30.1.11 v0.35.0 (2023-12-21)

Features

New IP-in-SCION Tunneling Gateway Service

The IP-in-SCION tunneling gateway service has been redesigned and rewritten from the ground up. Instead of wiping and recreating the entire gateway control and dataplane state on every change of inputs (configuration, remote tunneling endpoints, IP prefixes, etc.), the new gateway now calculates the minimal set of changes required to go from the current state to the desired state. This minimizes disruption to the dataplane and solves a longstanding issue where the gateway would not be able to process configuration and state changes fast enough under highly dynamic network conditions. While this rewrite does not change any user facing behavior, it ensures that the gateway service is now much more scalable and can handle much more dynamic network conditions.

Dataplane Performance Optimizations

The VPP dataplane has been upgraded to the latest stable release 23.10. This brings with it a number of improvements to reliability and forwarding performance. Furthermore, we optimized the AES computation needed for SCION packet forwarding to improve the forwarding performance of CORE/EDGE/GATE appliances up to 2.5x! Finally, the appliance now uses an improved hugepage and RX/TX queue allocation strategy depending on the available memory of the system and the port speed of the NICs in use.

Improvements

Improved support for Mellanox Connectx-5/6 NICs

The appliance has now improved support for Mellanox Connectx-5/6 NICs by using the VPP native RDMA driver. This improves forwarding performance for these NICs and allows users to configure Mellanox Connectx-5/6 devices directly as physical interfaces. Previously, Mellanox Connectx-5/6 devices had to be configured as virtual functions (VFs) and were only supported through the VPP DPDK driver.

Appliance API improvements

The appliance API includes several new endpoints:

- `GET /debug/vpp/bond` to list information about VPP bond interfaces.
- `GET /debug/vpp/lacp` to list VPP LACP information.
- `GET /debug/cluster/status` to provide information about the cluster synchronization status of various peers.

Additionally, the appliance API now validates that all tunneling endpoints that are configured on an appliance, either as local endpoints or as peer endpoints, are unique.

Finally, the API documentation now lists for every API the earliest release in which it is available.

Support for EBGP multihop

The appliance now supports EBGP multihop. This allows to configure EBGP sessions between appliances that are not directly connected. The number of hops can be configured via the `bgp.neighbor.ebgp_multihop` property in the appliance configuration.

Various other improvements

- If the appliance is part of a an AS acting as a CA and the CA service_type is set, the appliance validates that the address and necessary credentials are set.
- The `appliance-cli info` command now contains information about the overall appliance health and SCION interface states.
- The appliance configuration now includes an `experiments` section. This section can be used to configure experimental features that are not yet officially supported. Possible experiments will be documented in the relevant documentation sections.
- The routing daemon (FRR) and telemetry exporters (OpenTelemetry, Node Exporter, PromTail) have been updated to their latest minor versions.

Fixes

- The appliance now properly configures the IPv4 and IPv6 gateway addresses defined on the network interface. Previously, gateway addresses were only set for Linux interfaces.
- The appliance now properly propagates changes to the MAC address done on a VLAN's parent interface to the VLAN interface itself.
- The appliance now properly configures bond interfaces even if the configuration was interrupted through any means.
- Interactions between controllers and the dataplane cannot fail anymore with a "all channel IDs are in use" error. This sometimes caused a disconnection between controllers and dataplane, which could lead to a brief loss of connectivity in rare cases.

30.2 Appliance Release v0.34

This page contains the release notes for the v0.34 Anapaya appliance software release. The appliance software release is applicable for the following Anapaya products:

- Anapaya CORE
- Anapaya EDGE
- Anapaya GATE

We recommend always upgrading to the latest available patch release. Please refer to **Upgrade Notes** (if any) of each release if there are any special steps to be taken when upgrading. For general information on how to upgrade your appliance, please refer to *Updating an Appliance*.

Known Issues

- Changing the log level in the appliance-controller configuration will also change the log level in all managed services, thus leading to a restart of all services.
- Removing a VLAN interface on top of an interface with LINUX driver will not get removed from the system. It needs to be removed manually. This has been fixed in release v0.34.1.
- Adding a VLAN on a VF is not working correctly in all instances. This has been fixed in release v0.34.1.

30.2.1 Upgrade Notes

Warning: Release v0.34.0 and newer require Ubuntu 22.04 as the system and at least the anapaya-system package with version 2.8.0.

To check the current Ubuntu version use:

```
lsb_release -a
```

If that does not show Ubuntu 22.04, please install the base image with Ubuntu 22.04. Following the docs: *Installing the Appliance Base Image*, use version `sys_v2.8.0-scion_v0.34.0-1`.

To check the current version of the anapaya-system package either use the `appliance-cli`:

```
appliance-cli get software/system/installed
```

or `curl` (replace the `$APPLIANCE_API` with the correct address of the appliance):

```
curl -k https://$APPLIANCE_API/api/v1/software/system/installed
```

To upgrade an appliance that already is running Ubuntu 22.04 follow the following steps:

1. Install the anapaya-system package version v2.8.0 or newer if needed. Follow the the guide: *Update the anapaya-system package*
2. Restart the appliance-installer by running `systemctl restart appliance-installer` on the appliance.
3. Install the anapaya-scion package v0.34.0 or newer by following the guide: *Updating an Appliance*.
4. By default the appliance will not configure the new firewall rules. Enabling the firewall management through the appliance, will delete existing iptables rules. To enable the firewall in auto-mode, simply drop the firewall section from the appliance configuration (`appliance-cli edit config -i`), or follow the *Firewall* documentation to create a custom configuration.

30.2.2 v0.34.2 (2023-11-07)

Fixes

- The appliance now correctly validates configuration options for bond interfaces. This includes rejecting virtual functions as members for bond interfaces. Furthermore, VLAN interfaces are now correctly configured on top of bond interfaces.
- The appliance now removes the default DHCP configuration from the base image, when the appliance is first configured. Previously, the DHCP configuration was not removed, which could lead to unexpected behavior in certain circumstances.
- A bug in the IP-in-SCION tunneling component was fixed which would lead it to restart in certain circumstances when multiple local ISD-ASes were configured.

Improvements

- The appliance now supports configuring a telemetry address that is not available at API gateway start time. Previously, the appliance would fail to start if the telemetry address was not available at start time. Now, the appliance will retry to connect to the telemetry address.

30.2.3 v0.34.1 (2023-10-20)

Fixes

- The appliance now correctly uses `VPP_DPDK` as a default driver for an empty driver field in the `interfaces` section.
- VLANs configured on virtual functions that were running on interfaces with the `igbvf` driver are now correctly configured, however, on such interfaces only a single VLAN can be configured. The appliance configuration validator will reject a config that adds multiple VLANs on such a virtual function.
- The appliance now correctly deconfigures Linux VLAN interfaces when they are removed from the configuration.
- The appliance now correctly deconfigures existing virtual functions when they are removed from the configuration.

- The appliance now correctly unbinds a network interface from `vfio` in all circumstances when the driver is change from `VPP_DPDK` to `LINUX`.
- The appliance now correctly sets MTU on devices that use `ice` or `iavf` drivers.
- The appliance management API now correctly handles requests to the `/api/v1/network/*` endpoints again. In v0.34.0, they were returning 502.

Improvements

- The appliance now supports interfaces using the `ice` driver, i.e., Intel E810 Ethernet network adapters.
- The version and checksum of the installed `anapaya-system` package is now included in the output of `appliance-cli info`.
- `appliance-cli inspect scion-tunneling` is now an alias for `appliance-cli inspect tunneling`.

30.2.4 v0.34.0 (2023-09-28)

Features

SCION RSS

Starting with release v0.34, the Anapaya appliance supports SCION RSS. This feature significantly enhances the throughput on multi-core systems by enabling receive side scaling (RSS) for SCION traffic. SCION RSS operates by leveraging source port entropy on the UDP underlay and can only be enabled if both sides of a link support it. The appliance automatically detects if a sibling appliance supports SCION RSS via topology synchronization, and uses the feature accordingly. For non-sibling appliances, SCION RSS must be manually enabled in the appliance configuration for each neighbor link individually. By default, SCION RSS is not used for IP-in-SCION tunneling traffic. To benefit from increased performance of IP-in-SCION tunneling, enable SCION RSS by setting `scion_tunneling.endpoint.enable_scion_rss = true`. For more information, please refer to the documentation *here*.

Firewall management

The appliance now supports firewall management via the appliance management API. By default the appliance configures a sensible set of firewall rules to protect from undesired access to the appliance. The firewall is implemented using nftables. For further details, see the documentation available: *Firewall*.

New Appliance API endpoints

- The HTTP POST endpoint `/debug/notifications` triggers a system reconfiguration with the latest stored version of the configuration. The endpoint can be used to manually trigger a reconfiguration even when the notifications are turned off.
- The HTTP POST endpoint `/config/validate` can be used to validate an appliance configuration without applying it.
- The HTTP POST endpoint `POST /cppki/certificates/request` enables the manual request of a SCION CPPKI AS certificate chain for a given CSR using the regular certificate renewal mechanism. The endpoint expects a CSR and uses that to request a certificate renewal. The certificate renewal request is signed by an active key/certificate of the appliance such that the CA will be able to authenticate the renewal request and issue the certificate. This is useful if one appliance has been disconnected from the SCION network for several days and thus has no valid AS certificate anymore that could be used for certificate renewal. In such a case, one can

generate a new CSR on the appliance that was offline and use this endpoint on an appliance that still has a valid AS certificate to request a new certificate on behalf of the sibling. The returned certificate can then be deployed to the offline appliance using the regular `POST /cppki/certificates` endpoint.

Improvements

Merged appliance-host & appliance-controller

The appliance-host and appliance-controller services have been merged into the appliance-controller. As part of this change, the appliance-controller now runs as a systemd service and no longer as a docker container. As part of the upgrade the old services are automatically stopped and removed.

Improved configuration validation

There are several new configuration validation checks that are performed when pushing an appliance configuration. This should further minimize the risk of misconfiguration.

Interface pinning by default

Interface pinning of gateway endpoints (i.e. configuration of `allowed_interfaces`) is now enabled by default. When the `allowed_interfaces` property is not set, the local gateway endpoint will automatically use the locally configured SCION interfaces as `allowed_interfaces`. This feature is enabled by default, but can be disabled by setting the new API property `disable_auto_allowed_interfaces` to `true`. This feature is also disabled, when the `cluster.synchronization.address` property is set and therefore topology synchronization is enabled.

The `allowed_interfaces` are also automatically filled in for gateway endpoints on peer appliances, when they are listed in the `cluster.peers` section. This can also be disabled for peer appliances by setting `disable_auto_allowed_interfaces` on the peer tunneling endpoint to `true`.

Serving the appliance API via SCION

The appliance API can now be served via SCION. This is useful to manage the appliance via the SCION network.

Example SCION listener address: `[1-ff00:0:110,127.0.0.1]:443`

Various other improvements

- A set of BGP and FRR metrics are now available on the appliance metrics endpoint. The new metrics are prefixed with `frr_`. The documentation for the metrics can be found under *Telemetry*.

  Two alerts related to the FRR daemon state and the BGP peer state are added to all products. The relevant runbooks can be found under *Troubleshooting & Runbooks*.

- Add two new alerts related to the SCION dispatcher (`SCIONApplicationReadError` and `SCIONApplicationRegistrationError`). The alert applies to all appliance products. Check the corresponding runbooks under *Troubleshooting & Runbooks*.

- The appliance now exposes the `RootDistanceMaxSec` configuration for NTP. The configuration option can be found on `system.ntp.root_distance_max`.

  For more information refer to: systemd-timesyncd docs

- The appliance now strictly parses the configuration file, and a PUT request will fail if the configuration file is invalid. This behavior can be disabled by setting the `disable_strict_parsing` query parameter.
- The driver for a ethernet network interface can now be changed from `VPP_DPDK` and `VPP_VMXNET3` to `LINUX` without having to reboot the appliance.
- VLAN filters can now be configured on a virtual function (VF), meaning the NIC adds and removes the VLAN tag for outgoing and incoming traffic respectively.

  This configuration is required when the VF configures a single VLAN which is also configured on the PF.
- The TLS configuration of an appliance-cli context can now be updated after the context has been created.
- It is now possible to configure static neighbor entries for interfaces in the appliance configuration. A static neighbor defines a mapping of IP to MAC address. The configuration can be set via the `interface.ethernets.neighbors` property. For more details see the full configuration specification.

Fixes

- The appliance cron daemon now properly handles updates to the configuration of individual cron tasks. Under certain circumstances, an updated configuration was not properly picked up by the daemon.
- The appliance now correctly applies a change of the telemetry address. Previously, the appliance ignored the change of the telemetry address if it was already configured.
- The appliance will now properly ensure that all services are started on boot even if the periodic configuration notifications are disabled.
- The appliance configures a fixed value of 25 seconds as persistent keepalive value for all wireguard tunnels (this value is claimed to be working for most setups according to WireGuard docs). Previously, it was disabled, due to a bug.
- On reboot, the appliance ensures that network interfaces are bound to VFIO before handing control to the VPP dataplane.
- The appliance now properly configures wireguard interfaces and brings them up. Previously, the appliance might fail to bring the wireguard interfaces up if there was no change in the configuration.
- The appliance sets context deadline on notification updates. This ensures that the managers eventually finish handling a notification and do not keep hanging.

  The private `appliance_controller_notifications_queued_total` metric was added to track how many notifications are queued and/or dropped.
- The prefix discovery of the remote gateway now checks multiple paths to the remote AS. By checking one path for each interface in the remote AS, prefix discovery is successful even if two appliances in the remote AS are not internally connected.
- The following metrics were missing the `remote_address` label which meant that multiple gateways in an remote AS were overwriting each others values:
  - gateway_prefixes_advertised
  - gateway_prefixes_fetched
  - gateway_prefix_fetch_errors_total
- The router now properly cleans up the internal interface UDP socket, in case it is interrupted while configuring the dataplane. Previously, cleanup after the router received an interrupt could fail, because the UDP socket still existed when deleting the mirror and scion interface.
- The dataplane-control now properly handles all netlink messages. Previously, it could miss netlink messages if there were many of them in a short period of time.

- The appliance-cli now supports context names containing a `..`

Breaking Changes

- In the appliance configuration, the following names can no longer contain the pipes (`|`) character.
  - domain names
  - traffic matcher names
  - path policy names

  When migrating to the new version the appliance will automatically replace pipes in these names by slashes.
- The `appliance-cli inspect tunneling` commands and their output were restructured.

  `appliance-cli inspect tunneling paths`:

```
FINGERPRINT  STATE LATENCY JITTER DROPS EXPIRY   HOPS
2d1dfa220f6b alive 4.81ms  0.54ms 0.00% 3h14m29s 1-ff00:0:110 2>2 1-ff00:0:111
2dfc890de2ca alive 2.77ms  0.50ms 0.00% 3h14m29s 1-ff00:0:110 3>3 1-ff00:0:112
80fc02efd428 alive 4.77ms  0.90ms 0.00% 3h14m29s 1-ff00:0:110 1>1 1-ff00:0:111
b950e977ed92 alive 4.73ms  0.61ms 0.00% 3h14m29s 1-ff00:0:110 4>4 1-ff00:0:113
```

  `appliance-cli inspect tunneling summary`:

```
DOMAIN: donkeycorp
  PREFIXES: 10.0.1.0/24
    TRAFFIC MATCHER: normal
      PATH FILTER: default
        REMOTE: 1-ff00:0:111,172.20.2.3:30856
              STATE LATENCY JITTER DROPS EXPIRY   PATH
          --> alive 8.29ms  3.12ms 0.00% 3h14m29s 1-ff00:0:110 1>1 1-ff00:0:111
        REMOTE: 1-ff00:0:111,172.20.2.4:30856
              STATE LATENCY JITTER DROPS EXPIRY   PATH
            > alive 8.29ms  3.12ms 0.00% 3h14m29s 1-ff00:0:110 1>1 1-ff00:0:111
```

  - `-->` Indicates the active path for a traffic matcher within its domain.
  - `>` Indicates the candidate path for a currently unused remote.

  Multiple command line options were introduced to customize the output of the `summary` command.

30.3 Appliance Release v0.33

This page contains the release notes for the v0.33 Anapaya appliance software release. The appliance software release is applicable for the following Anapaya products:

- Anapaya CORE
- Anapaya EDGE
- Anapaya GATE

We recommend always upgrading to the latest available patch release. Please refer to **Upgrade Notes** (if any) of each release if there are any special steps to be taken when upgrading. For general information on how to upgrade your appliance, please refer to *Updating an Appliance*.

Important Upgrade Notes

- When an interface has virtual functions defined on it that are owned by VPP, it is necessary to adapt the appliance configuration and change the driver of the parent interface to LINUX.

Known Issues

- Certain DPDK drivers with missing features are not supported in v0.33.0 and v0.33.1. Support is added in v0.33.2.
- The algorithm for incremental changes to the network stack which was introduced in v0.33.0 has some issues. They are addressed in v0.33.2.
- Multiple entries in `system/vpp/tun/prefixes` are not correctly handled in versions v0.33.0 to v0.33.2. Starting from v0.33.3 all of the prefixes are correctly installed again.
- Configuring static BGP announcements in `bgp/global/networks` does not work in release v0.33.3. If you rely on this feature, upgrade to v0.33.4.
- Changing the `management/telemetry/address` is not working in all current releases. As a workaround, remove the address field and add it again with the new value in two separate configuration updates.
- Changing the log level in the appliance-controller configuration will also change the log level in all managed services, thus leading to a restart of all services.

30.3.1 v0.33.0 (2023-04-06)

Features

Alerts, Runbooks, and Monitoring Dashboards

Starting with release v0.33, we publish alerts, troubleshooting runbooks, and a set of prebuilt Grafana dashboards that can be used to monitor Anapaya EDGE, CORE, and GATE appliances. Check out the *runbooks* and the *documentation* on how to set up a monitoring stack and use the dashboards.

Appliance CLI

The appliance CLI is a new command line interface to interact with the appliance management API and replaces the previous reliance on the `curl` command. The appliance CLI can be used to configure the appliance, extract information, or upgrade the appliance software. It can be used on the appliance host itself or on a remote machine. For further details, refer to the *appliance CLI documentation*.

New Anapaya Appliance Base Image Installer

The Anapaya base image installer streamlines the installation of the Anapaya appliance on bare-metal devices. The installation can either be guided interactively or preseeded using a configuration file. Please refer to the *base image installation guide* for more details. Furthermore, the new base images are now based on Ubuntu 22.04 LTS.

Improvements

Improved Path Selection Algorithm

Anapaya EDGE and GATE appliances use an improved path selection algorithm. The algorithm now also properly takes droprate of the paths into account when selecting the best path. Furthermore, the heuristic for scoring paths based on latency and jitter has been improved. Finally, the algorithm generally prefers paths with fewer hops when other performance metrics are similar. The new path selection algorithm now also reports the reasoning for path switches in the log.

Appliance Management API Improvements

The newly introduced */management/api/listeners* configuration option allows specifying multiple listening addresses for the appliance management API. This is useful if the appliance management API should be reachable from multiple networks.

Furthermore, the appliance management API exposes new endpoints to provide additional insights:

- A new /debug/service-health endpoint provides the health status of all services running on the appliance. /debug/services/{service_name}/health can be used to fetch the health status of a single service.
- A new /debug/bgp/neighbors endpoint shows the state of BGP sessions to individual neighbors

Various Other Improvements

- When network configuration changes *dataplane-control* does incremental changes to the network stack instead of wiping everything and recreating it from scratch. This enables most network configuration changes without service disruption.
- The `bgpd` daemon that is running as part of the FRR module now always configures `soft-reconfiguration inbound` for all BGP neighbors. This allows BGP sessions to be reconfigured without having to reset the BGP session.
- The number of hugepages can now be configured without the need for a reboot.
- The appliance configuration validator has been extended to validate that the /scion/ases/forwarding-key is properly base64 encoded.
- The appliance API now supports configuration of strict CSR validation. For this, the `/scion/ases/ca_service/anapaya_vault/validation/subject` property has been added to the API. It can be set to one of the following values:
  - `MATCHING_ISD_AS` (default): The Anapaya Vault backend validates that the ISD-AS field of the CSR subject is the same as in the certificate that was used to sign the request.
  - `EXACT_MATCH`: The Anapaya Vault backend validates that the complete CSR subject is exactly the same as in the certificate that was used to sign the request.

  This setting is only relevant if the appliance is part of a CA AS that uses the Anapaya SCION CA service.

Fixes

- The appliance management API now stays responsive even after prolonged (around two weeks) inactivity. Previously, the appliance API would hang because it failed to issue the self-signed certificates due to an internal mismatch between the cache and the actual files on disk.
- Routing updates on the Linux routing table are now always properly mirrored to the VPP-based forwarding plane. In rare circumstances, the VPP-based forwarding plane was missing some routing updates leading to a loss of connectivity.
- Network interfaces based on Mellanox Connectx-5 NICs are now properly detected and configured.
- Appliance configuration changes are now atomically written to the file system. Previously, aborting the process (e.g., by pulling the plug) could lead to corrupted files.
- The gateway_session_is_healthy metric is now updated correctly. Previously, it would only be updated when the session state changed, leaving the metric uninitialized for a long time in certain cases.

Breaking Changes

- The `/management/api/address` has been removed in favor of the newly introduced */management/api/listeners* list, which allows specifying multiple listening addresses at the same time. The appliance configuration needs to be adopted accordingly.

  E.g., if your previous `api` configuration looked like this:

  ```
  "address": "127.0.0.1:443"
  ```

  It must be migrated to the following:

  ```
  "listeners": [
      {
          "address": "127.0.0.1:443"
      }
  ]
  ```

  The appliance will automatically migrate the configuration on startup.
- The `/cluster/synchronization/legacy_address` configuration option has been removed. This option was used to configure cluster synchronization with releases prior to v0.30. This release does not support cluster synchronization with appliances running releases prior to v0.30 anymore.

30.3.2 v0.33.1 (2023-04-12)

Fixes

- Fixes a bug in v0.33.0, where the appliance was unable to configure VLANs.

30.3.3 v0.33.2 (2023-05-08)

Improvements

- The initial password for the default `anapaya` user on the appliance API is changed to `anapaya`. This simplifies initial setups of the appliance.

  Note: This only affects new installations. Existing installations are not affected by this change.

  Warning: Make sure to change the default password before exposing the appliance API to the network.

Fixes

- Ensure that `localhost` is always configured on an appliance. For clean installations from the base image, it was missing in some cases and would lead to the API gateway not being able to reach the internal processes.
- Fallback to a legacy method for DPDK drivers that do not support a specific multicast ethernet address setup operation. In prior releases, adding a multicast MAC address would fail for such drivers.
- Properly send neighbor probes. Previously, these probes were not sent and neighbors would be consistently considered not reachable. This resulted in short interruptions of reachability, and some dropped packets depending on the load.
- Changes to the MTU of a VLAN interface and its parent interface are now applied in the correct order. Previously, MTU changes to VLAN interfaces could fail.
- Correctly handle multiple IP routes to the same prefix.
- If no MAC address is defined for a bond interface, a new random MAC address is used instead of the MAC address of the first added interface. This prevents both interfaces from using the same MAC address in case the first added interface is removed from the bond.
- The IPv6 link local address is no longer set on an interface in DOWN state. Because Linux never applies the address on a interface in DOWN state, setting the address would previously fail.

30.3.4 v0.33.3 (2023-06-08)

Improvements

- Add support for the VPP native driver for VMXNet3 interfaces. This driver works better than the DPDK driver for VMXNet3 and fixes an issues with multicast MAC setup on interfaces with VMXNet3 driver.
- The appliance now only redistributes routes to the BGP peers that were learned over SGRP. Routes that are learned via BGP are not redistributed to peers.
- The appliance API now validates the OAuth metadata and JWKS URLs when OAuth is enabled. If the appliance cannot reach the URLs, the configuration is rejected.
- Use NAT Endpoint Dependent (ED) mode instead of Endpoint Independent to track TCP connection state and improve NAT session recycling for TCP connections.
- Improve IP-in-SCION tunneling flow reporting. Flows are now always exported even in case of no client traffic, to ensure that the flow exporting mechanism is properly running. Old flows are correctly deleted.

Fixes

- The IP-in-SCION tunneling component no longer produces gaps in the frame number. This fixes an issues where the receiving side would report frame drops, although, in fact, no data was lost.
- Fix setting up multiple prefixes in the system/vpp/tun/prefixes list. This was broken with the state synchronization introduced in release 0.33.0.
- Properly support disabling OAuth. Previously disabling OAuth could result in a wrongly configured API Gateway.
- Ensure that the API Gateway starts after a reboot. Previously it sometimes failed to start because the network interfaces were not ready.

30.3.5 v0.33.4 (2023-06-14)

Fixes

- The appliance correctly announces statically defined prefixes defined in *bgp/global/networks* to its BGP peers after it was broken in release v0.33.3.
- The gateway service now properly serializes configuration and state updates. In rare cases, having multiple concurrent configuration updates and state changes would lead to a crash of the gateway service.

30.3.6 v0.33.5 (2023-07-07)

Fixes

- The IP-in-SCION tunneling module can no longer end up in a deadlocked state when configuring the forwarding plane. Certain (rare) interleavings of reconfiguration events could lead to a state where the module that configured the forwarding plane could not make progress anymore.

30.3.7 v0.33.6 (2023-08-30)

Fixes

- Fetching the topology from other appliances now correctly works even if one appliance is not reachable. Previously, the fetch could fail even on appliances that are alive if one appliance was not reachable.
- Fix a race condition between the certificate cleanup job and the CSR to certificate promotion. In rare cases, the cleanup job would delete the key of a newly posted certificate, this will now no longer happen. So far we only saw this in testing environments.
- Fix an issue in determining the difference between two IP-in-SCION tunneling configurations. If the length of the failover_sequence was decreased it could lead to a crash of the IP-in-SCION tunneling component and if it was increased it could be that it wouldn't be applied. The difference is now correctly determined.
- IPv6 link-local addresses are now reliably configured. In rare instances, the configuration of a IPv6 link-local was not completed in time, which could lead the network dataplane to be incorrectly configured.

Improvements

- The appliance API now also accepts `appliance.reader` and `appliance:reader` in addition to `appliance/reader` and `appliance.writer` and `appliance:writer` in addition to `appliance/writer` for the `roles` parameter of an OAuth2 token.
- The appliance-cli now properly formats PEM encoded content, such as certificates, keys, and CSRs.
- Logs of all services are now sent to journald. This simplifies ingestion to centralized log management systems, such as Loki.

30.4 Appliance Release v0.32

This page contains the release notes for the v0.32 Anapaya appliance software release. The appliance software release is applicable for the following Anapaya products:

- Anapaya CORE
- Anapaya EDGE
- Anapaya GATE

We recommend always upgrading to the latest available patch release. Please refer to **Upgrade Notes** (if any) of each release if there are any special steps to be taken when upgrading. For general information on how to upgrade your appliance, please refer to *Software Updates*.

Warning: Known issues

- When an interface with VPP driver has an **IPv6 link-local address** configured and the interface is down (L1, e.g. there is no physical connection) when the EDGE is booted, the interfaces with VPP driver do not work. The problem is resolved as soon as the physical connection is established (e.g. by plugging in the cable) and the device is rebooted.
- Devices with Mellanox network cards don't work correctly with the v0.32.0 release.
- There is a race condition when setting up SCION interfaces with IPv6 LL addresses. If a SCION interface with IPv6 LL address doesn't come up, remove it and re-add it. If that doesn't make it working please contact Anapaya support.
- On releases before v0.32.3 there is a rare bug in the IP-in-SCION tunneling component that could make the component be hung up. A restart of the gateway helps. To restart use the restart API (use `gateway` as the `{service_name}`). It is recommended to upgrade to v0.32.3.

30.4.1 v0.32.0 (2022-11-30)

Features

OAuth2/OIDC support in Appliance API

The appliance API now supports authentication based on OAuth2/OIDC. Users can authenticate with their existing identity provider (either Azure AD or Auth0) via the appliance web UI or register an application to use the API. Furthermore, each user or application can be granted read-only or read-write permissions. For further details, see the *configuration documentation* and the *setup guide*.

Source NAT

It is now possible to set up source NAT on the appliance.

The feature is useful when replies to the packets coming out from a IP-in-SCION tunnel are supposed to be routed back to the tunnel while other packets can still be routed in an arbitrary user-defined way.

For further details consult the documentation: *Network Address Translation (NAT)*.

Debug endpoints

The appliance API is extended with various debug endpoints. Those endpoints provide insights during the debugging of a problem with the appliance. Note that those endpoints are experimental and can change in the future. Check the API documentation for more details.

Fixes

- The appliance no longer automatically migrates old configurations if the previous version was older than v0.30. This is to prevent issues that happened during those upgrades. In *Upgrade Notes* it is explained how to properly upgrade from older versions.
- The router correctly handles an error during the creation of a SCION interface. Previously, it could occasionally fail to create the internal interface properly.
- The router no longer configures sibling interfaces if no local interfaces are defined. A router with no local links would not be configured on peer routers and therefore one-sided BFD session would be created, leading to a confusing state.
- Address a memory leak in the IPFIX feature.
- The IP-in-SCION tunneling component no longer fails if there is an ISD-AS configured on the appliance which for which no TRC is present.
- The IP-in-SCION tunneling component correctly tunnels small packets. There was an issue where very small packets were incorrectly encapsulated under some circumstances.

Breaking Changes

- The `/management/telemetry/logging/loki/external_labels` is removed from the appliance configuration. Instead there is a new field `/management/telemetry/labels` that can be used to set labels on metrics and logging at the same time. The appliance will automatically migrate old configurations.
- The `public_key` field is removed from the `wireguard` interfaces section. The field was read-only and was not meant to be set by an appliance operator. To retrieve wireguard keys use the new `/network/wireguards` API endpoint. For further information, check out the documentation for the API endpoint.

Upgrade Notes

Warning: When upgrading from a release older than v0.30.0 it is recommended to first create the new configuration for the current release and store it on the appliance under `/etc/anapaya/appliance/migrations/v0_32/config.json`. The file needs to be owned by the scion user and should only have read-write permissions for this user (`0600`). This will make sure that the appliance is correctly configured when it starts in the new version.

30.4.2 v0.32.1 (2023-01-03)

Fixes

- The appliance now correctly validates the *system/vpp/tun* section of the configuration.
- The dataplane no longer crashes when an interface with traffic on it is removed.

30.4.3 v0.32.2 (2023-01-06)

Fixes

- Dataplane reconfiguration with NAT enabled no longer fails.

30.4.4 v0.32.3 (2023-02-06)

Fixes

- Fix renewing of self-signed certificate in the management API, if the listening address is `127.0.0.1`.
- Fix a rare bug that leaves the IP-in-SCION tunneling component in a hanging (non-working) state after an error occurred.
- Fix crash loop on the router component that happened sometimes after a restart.

30.4.5 v0.32.4 (2023-02-22)

Fixes

- The gateway dataplane reconfiguration logic was improved to better handle large number of changes in IP prefixes.

30.4.6 v0.32.5 (2023-04-28)

Improvements

- The default password of the *anapaya* ` use on the appliance API is changed to *anapaya*.

30.4.7 v0.32.6 (2023-05-04)

Fixes

- The new default password in *v0.32.5* ` did not work due to an internal issue.

30.5 Appliance Release v0.31

This page contains the release notes for the v0.31 Anapaya appliance software release. The appliance software release is applicable for the following Anapaya products:

- Anapaya CORE
- Anapaya EDGE
- Anapaya GATE

We recommend always upgrading to the latest available patch release. Please refer to **Upgrade Notes** (if any) of each release if there are any special steps to be taken when upgrading. For general information, on how to upgrade your appliance, please refer to *Software Updates*.

Known Issues

- Metrics exported by the appliance are missing labels for releases v0.31.0 - v0.31.2.

30.5.1 v0.31.0 (2022-10-06)

Features

Cryptographically Signed Releases

To verify that an Anapaya software release has not been tampered with from the time it is released until the user installs it on an appliance, we now cryptographically sign our software releases of the `anapaya-scion` and `anapaya-system` packages. This enables the appliance installer as well as third parties to verify the authenticity and integrity of our software releases against a root of trust.

To sign our releases, we generate an ECDSA-P256 key pair. Using the private key, our signing tool computes the signature over the SHA256 hash of an Anapaya software package, which will serve as the signature of the corresponding package. The private key is stored in a highly-secure, access-controlled location to ensure that it is not compromised.

The public keys and the signatures of each release is published on releases.anapaya.net.

For more information on how to verify the authenticity and integrity of Anapaya software release, please refer to the *Signed Releases* section of the documentation.

Automatic Topology Synchronization

Anapaya appliances can now be configured to automatically exchange network topology information with each other. This greatly simplifies the configuration of additional SCION links to other SCION ASes. Previously, the existence of a link needed to be manually configured in each appliance. With automatic topology synchronization, only the appliance that owns the SCION link needs to be configured and every other appliance will automatically learn about it dynamically.

For more information on how to configure automatic topology synchronization, please refer to *Topology Synchronization*.

Flow Metrics

Appliances can now be configured to export metrics for IP-in-SCION tunneling flows via HTTP, HTTPs, or gRPC to a flow collector. This allows for the collection of statistics on the number of flows, their duration, and the amount of traffic they have generated per source and destination. For more information, please refer to the *telemetry configuration*.

Improvements

- The appliance now has an API gateway in front of the HTTP APIs. The `/metrics` telemetry endpoint is now also available on the management API address, the telemetry address is now optional. On installation, the appliance is configured to listen on all IPs on port 443 with a self-signed certificate for HTTPs and a default `anapaya` account for basic authentication. This should be changed with the initial configuration. The default configuration of the management API address is now `:443`. When upgrading from a previous release, we recommended to change the management API address port to 443.

Fixes

- The appliance controller process no longer aborts execution if it encounters a missing SCION section.
- Notification disabling is now working as expected. If `disabled` is set and the deadline is in the future or not set, notifications are disabled otherwise notifications are enabled.

Breaking Changes

- The `/scion_tunneling/path_filters/sequence` is removed from the appliance configuration. This has already previously been renamed to `/scion_tunneling/path_filters/hop_pattern`. Please migrate your configuration to use the new name. There is no auto-migration support for this.

30.5.2 v0.31.1 (2022-10-18)

Fixes

- Flow metrics are now correctly exported via HTTP(s).

30.5.3 v0.31.2 (2022-10-21)

Improvements

- Added the option to configure the source address for the IPFIX exporter.

30.5.4 v0.31.3 (2022-11-04)

Improvements

- Topology synchronization can now be deployed and enabled gradually. Previously, all appliances were required to be on release v0.31 to enable automatic topology synchronization.
- The``anapaya-scion`` package can now be installed even if the API gateway is not available on the appliance yet. This is crucial for upgrades from previous releases.

Fixes

- Metric labels are now correctly exported again.
- The ports of the exported flow metrics are now correctly displayed. Previously, the endianness of the port numbers was inverted.

30.5.5 v0.31.4 (2022-11-15)

Fixes

- The `anapaya-scion` package can now be correctly installed on appliances even when an API gateway has already been installed.

30.6 Base Image Release sys_v2.8.0-scion_v0.34.1-1

This page contains the release notes for the sys_v2.8.0-scion_v0.34.1-1 Anapaya Appliance Base Image. The base image release is applicable for initial provisioning of the following Anapaya products:

- Anapaya CORE
- Anapaya EDGE
- Anapaya GATE

We recommend always installing the latest available base image when provisioning appliances for the first time. For general information on how to install a base image, please refer to *Installing the Appliance Base Image*.

30.6.1 Anapaya-system package

The anapaya-system package v2.8.0 is used.

- Includes the latest version of the `appliance-installer`, `caddy`, `anapaya-system-config` packages, as well as security updates to base packages.

30.6.2 Anapaya-scion package

The anapaya-scion package v0.34.1 is used. For more information, please refer to the release notes *Appliance Release v0.34*

30.6.3 Checksum

The checksums of this base image version are:

- ISO - Checksum (SHA-256): `d92e979acaf232218c5abef67fa5c1a83ab7702fc748afd16207d7d72ed83bb9`
- qcow2 (BIOS) - Checksum (SHA-256): `7e6dbda8f9e11f32def8ec9f4418a3c951114c70a136f345f0d4aacc9f076ea9`
- qcow2 (UEFI) - Checksum (SHA-256): `8342da1238c4b1a1d76e6252f18ae4dde59f775cf26015225bd0f3713293a8ae`
- VMDK (UEFI) - Checksum (SHA-256): `c609c36e080a82d20873d17d55b352e56c13e832358e0fb6bf95aec7c196cdad`

30.7 Base Image Release sys_v2.3.0-scion_v0.32.6-1

This page contains the release notes for the sys_v2.3.0-scion_v0.32.6-1 Anapaya Appliance Base Image. The base image release is applicable for initial provisioning of the following Anapaya products:

- Anapaya CORE
- Anapaya EDGE
- Anapaya GATE

We recommend always installing the latest available base image when provisioning appliances for the first time. For general information on how to install a base image, please refer to *Installing the Appliance Base Image*.

30.7.1 Anapaya-system package

The anapaya-system package v2.3.0 is used.

- Adds `wireguard-tools`, `nftables` and `jq` packages to the anapaya-system package.
- Performs security updates of the included packages.

30.7.2 Anapaya-scion package

The anapaya-scion package v0.32.6 is used. For more information, please refer to the release notes *Appliance Release v0.32*

30.7.3 Checksum

The checksums of this base image version are:

- ISO - Checksum (SHA-256): `bcccd7224a3970e9cc44daf0cefb943b677ed018f790041c5953c56ef40c6b18`
- qcow2 (BIOS) - Checksum (SHA-256): `687f5d48800a3d68770e61c505a2eeedeb4e82a4a4905cd91ad4c89ff020ac5d`
- qcow2 (UEFI) - Checksum (SHA-256): `3944cc9421ea6e38739e31a58e1cf087c2cb8e9d01ff579104f1271b59ac42db`

30.8 Base Image Release sys_v2.2.0-scion_v0.32.3-1

This page contains the release notes for the sys_v2.2.0-scion_v0.32.3-1 Anapaya Appliance Base Image. The base image release is applicable for initial provisioning of the following Anapaya products:

- Anapaya CORE
- Anapaya EDGE
- Anapaya GATE

We recommend always installing the latest available base image when provisioning appliances for the first time. For general information on how to install a base image, please refer to *Installing the Appliance Base Image*.

30.8.1 Anapaya-system package

The anapaya-system package v2.2.0 is used.

- Adds package `wireguard-tools` to the anapaya-system package.
- Performs security updates of the included packages.

30.8.2 Anapaya-scion package

The anapaya-scion package v0.32.3 is used. For more information, please refer to the release notes *Appliance Release v0.32*

30.8.3 Improvements

- Releases the base image as a `qcow2` and an `iso` image.
- Adds a base image installer that can install the base image with no user interaction.
- The base image is based on Ubuntu 22.04.

30.8.4 Checksum

The checksum of this base image version is:

- ISO - Checksum (SHA-256): `11f1a1e6ecb1d9992da436f71c330253c887336384da4de86af52ad604c56213`

CHAPTER

THIRTYONE

ISD AND AS ASSIGNMENTS

The following isolation domains (ISD) and autonomous systems (AS) are a assigned by Anapaya.

31.1 Isolation domains

ISD	Name	Description	TRC Bundle
64	Switzerland	*Swiss ISD - public ISD for all SCION network participants in Switzerland*	`ISD64`
65	Europe	*Europe ISD - public ISD for all SCION network participants in Europe*	`ISD65`
66	Asia	*Asia ISD - public ISD for all SCION network participants in Asia*	`ISD66`
67	North America	*North America ISD - public ISD for all SCION network participants in North America*	`ISD67`
68	Reserved		
69	Reserved		
70	SSFN	Secure Swiss Finance Network	`ISD70`
71	SCION Education Network		`ISD71`
72	HVR	HIN Vertrauensraum	`ISD72`

31.2 Autonomous Systems

Reach out to cse@anapaya.net if you want to get your AS publicly listed.

AS	ISDs	Name	Description
88	71	Princeton University - CITP	Princeton University Test AS
225	71	University of Virginia	University of Virginia SCION AS
114	71	SIDN Labs	SIDN Labs SCION AS
254	71	NCSR Demokritos	NCSR Demokritos SCION AS
333	64, 70, 72	Swisscom	Swisscom SCION AS
156	64, 70, 72	Cyberlink	Cyberlink SCION AS
209	71	GEANT	GEANT SCION AS
2:0:	64	Anapaya Zurich	Collecting SCION links via the Internet
2:0:	64	SIX Integration Environment	Environment for Connectivity and Process verifications for dedicated test infrastructure
2:0:	65	Anapaya CONNECT Frankfurt	Anapaya PoP in Frankfurt
2:0:	66	Anapaya CONNECT Singapore	Anapaya PoP in Singapore - SCION AS Certificate Issuer for ISD 66
2:0:	66	Anapaya CONNECT Hong Kong	Anapaya PoP in Hongkong
2:0:	64	Anapaya Zurich	Anapaya PoP in Zurich @ SwissIX - SCION AS Certificate Issuer for ISD 64
2:0:	65	Anapaya Frankfurt	Anapaya Test AS - SCION AS Certificate Issuer for ISD 65
2:0:	64	Anapaya Hardturm	Anapaya Test AS - announcing 193.29.21.224/28
2:0:	67	Swisscom US	Swisscom SCION AS Los Angeles
2:0:	71	BRIDGES	BRIDGES SCION AS - Connecting US and European research institutions across the Atlantic
2:0:	64	Mysten Labs	Mysten Labs Switzerland
2:0:	65	Mysten Labs	Mysten Labs Paris
2:0:	67	Mysten Labs	Mysten Labs Ashburn
2:0:	71	CybExer Technologies	CybExer Technologies SCION AS
2:0:	71	Otto-von-Guericke-Universität Magdeburg	Otto-von-Guericke-Universität Magdeburg SCION AS

CHAPTER

THIRTYTWO

DEFAULT PORT ALLOCATIONS

The Anapaya appliance exposes multiple APIs and component communication endpoints at different ports.

There are two **classes** of network endpoints:

- **Underlay endpoints**: These are always UDP/IP.
- **Service endpoints**: These can be TCP/IP, UDP/SCION or QUIC/SCION depending on how the corresponding service endpoint should be reachable.

The port mapping is provided to the appliance through the appliance configuration file. There exist default values for each network endpoint. However, an operator can manually specify different ports for the network endpoints as long as there is no port overlap.

32.1 Default Port Tables

Here, we provide a table for each of our applications which includes the default port numbers. Note that if more than one AS is configured on the appliance or if some of the default port numbers are used by the operator for a different purpose, then the allocated ports might not match the default values provided in the tables below.

32.1.1 Control

Endpoint	Protocol	Class	Default Port
IntraAS	TCP/IP & UDP/SCION	Service	40000
InterAS	QUIC/SCION	Service	chosen by service
Cluster	UDP/IP & TCP/IP	Service	40001

32.1.2 Router

Endpoint	Protocol	Class	Default Port
Internal Interface	UDP/IP	Underlay	30100
External Interface	UDP/IP	Underlay	31000-39999

Note: The External Interface ports are set by the operator. Any ports from the range 31000-39999 can be used. By default, you can assign port 31000 to an external interface and increase monotonically from there if multiple external

interfaces are on the same IP address. Otherwise, port 31000 can be reused.

32.1.3 Gateway

Endpoint	Protocol	Class	Default Port
Data	UDP/SCION	Service	40200
Control	QUIC/SCION	Service	40201
Probe	UDP/SCION	Service	40202

32.1.4 Dispatcher

Endpoint	Protocol	Class	Default Port
Data	UDP/IP	Underlay	30041

32.1.5 Appliance Controller

Endpoint	Protocol	Class	Default Port
Appliance Mgmt API	TCP/IP	Service	42000
Telemetry	TCP/IP	Service	42001
Synchronization API	gRPC (TCP/IP)	Service	42003

32.2 L3 Communication Matrices

The following port ranges are used to access the management and telemetry APIs of the Anapaya appliance and for the Anapaya appliances to communicate with each other. Firewall rules need to be configured accordingly to allow communication on these ports.

32.2.1 Appliance Management

Endpoint	Protocol	Port
Management API	TCP/IP	42000
Telemetry	TCP/IP	42001

32.2.2 Appliance Intra-AS

Endpoint	Protocol	Port	Comment
Dispatcher	UDP/IP	30041	
Internal SCION Interface	UDP/IP	30100-30199	Required ports depend on the number of internal interfaces.
Control Plane	TCP/IP	40000-40099	Required ports depend on the number of ISD-ASes (two ports per ISD-AS).
Appliance Cluster	TCP/IP	42002	
Appliance Cluster	UDP/IP	42002	
Appliance Topology Synchronization	TCP/IP	42003	

32.2.3 SCION Inter-AS Links

Endpoint	Protocol	Port	Comment
External SCION Interface	UDP/IP	31000-39999	Required ports depend on the number of external interfaces. Note that these are external facing ports and usually outside of any firewall infrastructure.

CHAPTER

THIRTYTHREE

IMPACT ANALYSIS

This document describes the impact of various changes to the system. The impact is described for configuration changes and manual container restarts and is split into the following categories:

- Impact on SCION Data Traffic
- Impact on SCION Control Plane Traffic
- Impact on IP-in-SCION Tunneling Traffic

33.1 Configuration Changes

Config Section	Config Change	SCION Traffic	Control Plane Traffic	IP-in-SCION Tunneling Traffic
BGP *(config.bgp)*	Any	No	No	Potentially Interrupted
Cluster *(config.cluster)*	Any	Potentially Interrupted	Potentially Interrupted	Potentially Interrupted
Interfaces *(config.interfaces)*	Modification of physical interface	Interrupted	Interrupted	Interrupted
	Any other interface change	No	No	No
Management *(config.management)*	Any	No	No	No
NAT *(config.nat)*	Any	No	No	Interrupted
SCION *(config.scion)*	Any	No	Potentially Interrupted	No
SCION Tunneling *(config.scion_tunneling)*	Any	No	No	Potentially Interrupted
System *(config.system)*	DNS, NTP	No	No	No
	VPP, Kernel	Interrupted	Interrupted	Interrupted

33.2 Container Restarts

Container	SCION Traffic	Control Plane Traffic	IP-in-SCION Tunneling Traffic
Dataplane	Interrupted	Interrupted	Interrupted
Dataplane-control	No	No	No
Router	No	No	No
Dispatcher	No	Interrupted	Interrupted
Control	No	Interrupted	No
Daemon	No	No	No
Gateway	No	No	Interrupted
FRR	No	No	Interrupted
Appliance-controller	No	No	No
Appliance-cron	No	No	No

CHAPTER

THIRTYFOUR

FREQUENTLY ASKED QUESTIONS

34.1 Joining a SCION Network

34.1.1 How can my organization join the Secure Swiss Finance Network (SSFN)?

To join the SSFN the following process must be followed:

1. The interested organization needs to order SCION access from a SCIONabled provider. For more information on SCIONabled providers that are also part of the SSFN, please refer to our official page.
2. The interested organization needs to be SSFN certified by SIX. For further information on the certification process, please refer to the SIX's official SSFN page.
3. Once the contracts are ready, the providers can schedule the installation and configuration of the Anapaya appliances on the customer premises.

34.1.2 How can my organization join the HIN Vertrauensraum (HVR)?

To join the HVR the following process must be followed:

1. The interested organization needs to order SCION access from a SCIONabled provider. For more information on SCIONabled providers, please refer to our official page.
2. The interested organization needs to be HVR certified by HIN. For further information on the certification process, please contact HIN.
3. Once the contracts are ready, the providers can schedule the installation and configuration of the Anapaya appliances on the customer premises.

34.1.3 How can my organization join the Swiss ISD?

To join the SSFN the following process must be followed:

1. The interested organization needs to order SCION access from a SCIONabled provider. For more information on SCIONabled providers, please refer to our official page.
2. Once the contracts are ready, the providers can schedule the installation and configuration of the Anapaya appliances on the customer premises.

34.2 Anapaya Products

34.2.1 What is Anapaya EDGE?

Anapaya EDGE is the gateway that connects users to the SCION Internet. The Anapaya EDGE is installed physically or in a virtual environment at the edge of a customer's network. The Anapaya EDGE is connected to the SCION network through a SCIONabled ISP. Through Anapaya's IP-in-SCION tunneling mechanism, the EDGE can tunnel IP traffic from the customer's LAN and communicate with remote destinations through the SCION network.

For information on Anapaya EDGEs and common setups, please refer to *Overview* and *Anapaya EDGE*.

34.2.2 What is Anapaya CORE?

Anapaya CORE is a router that is part of the SCION backbone. It is placed at the border of the network of Internet Service Providers and connects to other SCIONabled ISPs and customers.

For information on Anapaya COREs and common setups, please refer to *Overview* and *Anapaya CORE*.

34.2.3 What is Anapaya GATE?

Anapaya GATE is the gateway that connects end-users to the SCION Internet. It allows users of SCIONabled ISPs to connect to remote locations, such as a company's headquarter network using the SCION Internet even if the user does not have an EDGE at home. It is located at the edge of SCIONabled ISPs and advertises the ISP user prefixes to SCIONabled organizations and vice versa. This solution protects remote access, e.g., VPN connections, and access to other workloads by an organization's remote workforce.

34.3 Processes

34.3.1 How can my organization get a SCION AS number?

Depending on the ISD, there are different entities that are responsible for assigning the AS numbers. Please refer to *ISD and AS Assignments* for further information.

34.4 IP-in-SCION Tunneling

34.4.1 What is the IP-in-SCION Tunneling mechanism?

IP-in-SCION tunneling is a mechanism that allows tunneling IP packets between two IP-in-SCION tunneling endpoints, i.e., Anapaya EDGE appliances. This mechanism enables any type of application and communication to take advantage of SCION-based networking without the need to update any application, client, or server.

For more information, please refer to *IP-in-SCION Tunneling*.

34.4.2 What kind of prefixes can I configure to announce through IP-in-SCION tunneling?

In most setups, Autonomous Systems can announce private or public IP prefix ranges depending on their setup. There are special cases such as the SSFN, where participants are required to only announce public IP prefix ranges, which are not routed through the public Internet.

In general, IP-in-SCION tunneling is used to enable communication between the local site and a remote destination. Our implementation supports tunneling with IPv4 and IPv6 prefixes.

34.4.3 Can a prefix range be advertised by multiple ASes?

Yes, it is possible for a prefix range to be advertised by multiple ASes. In such cases, the appliances in the AS receiving the prefix announcement need to configure policies for which remote AS to prioritize when sending packets to that prefix range.

34.5 Control Plane

34.5.1 How often does the SCION control plane create new paths (beaconing)?

The beaconing process is initiated by core ASes every 30 seconds by default. This enables the network to detect topology changes fast enough, without adding a serious overhead to the network. The frequency with which the Beaconing process happens can be configured to other values if desired.

34.5.2 For how long are SCION paths valid?

By default, SCION paths have a validity period of six hours. In other words, when a sender, e.g., an Anapaya EDGE, requests a path from the network, the path can be used for six hours until it needs to be renewed. In practice, paths are updated every few minutes when the beaconing process leads to the creation of new path segments.

It is worth noting that the default validity period of paths is configurable.

34.6 Data Plane

34.6.1 How are paths chosen by the sender?

A sender selects a path toward its destination depending on the available paths, the static policies configured and dynamic criteria such as latency and jitter.

The frequency with which the chosen path changes depends on the user configured policies and the application.

In the Anapaya EDGE, the sender will use the same path toward the same destination for as long as the path remains alive and the performance is not degraded. Additionally, by default, a single path is used to reach a destination. However, the sender can configure smart policies to differentiate traffic and use different paths.

34.7 Setup

34.7.1 On what kind of platform can I run the Anapaya software?

The Anapaya software can run both on physical hardware and virtual environments. For details on computing requirements, please refer to our Computing resources guide.

34.7.2 How can an Anapaya EDGE connect to a SCIONabled provider?

The recommended way to connect to a SCIONabled provider is through the ISP's access network. Please contact your SCIONabled access provider for further details.

34.7.3 How can an Anapaya appliance be connected to the internal network of an organization?

An Anapaya appliance can connect to the internal network of the organization it belongs to through a variety of connection types. Specifically, supported setups include static routing, eBGP, VRRP (in case of redundant setups). For more information, refer to *Anapaya EDGE* and *Anapaya CORE*.

34.7.4 What is the difference between MTU and SCION MTU?

Due to the fact that every SCION packet uses an IP/UDP underlay, a SCION packet has an overhead of 24 bytes for IPv4 and 48 bytes for IPv6. For this reason, the SCION protocol defines the SCION MTU which is equal to the difference of the underlay MTU minus the overhead bytes. This means that effectively slightly fewer bytes can be communicated through a single SCION packet compared to a non-SCION packet. Note also that the size of the payload additionally depends on the length of the chosen SCION path since the latter is included in the packet's SCION header.

34.8 Security guarantees

34.8.1 How can SCION and Anapaya appliances offer protection against DDoS attacks?

There are a few different scenarios in which Anapaya appliances and SCION can protect against DDoS attacks.

Scenario 1: The attacker is not part of the SCION network and tries to attack the target through the public Internet. The target uses prefixes that are not routed in the public Internet, but only in the SCION network.

Protection: Since the target's prefix range is not accessible through the public Internet, the target is basically "invisible" to the attacker. In other words, the attacker cannot carry out the attack as it has no way of reaching the target's network.

Scenario 2: The attacker is not part of the SCION network and tries to attack the target through the public Internet. The target uses prefixes that are routed through the public and the SCION Internet.

Protection: The attacker can reach the target's network only through the public Internet. This means that users connecting through the public Internet to the target can suffer from availability problems of the service in case of an attack. However, users connecting through the SCION network use a different entry point to the service which will remain unaffected.

Scenario 3: The attacker is part of the SCION network.

Protection: In this case, the user needs to resort to other defense mechanisms such as SCION Hidden Paths. With Hidden Paths, an organization can control which other SCION ASes can retrieve SCION paths leading to the organization's network. That way, the organization is in control of which entities can send it traffic.

To learn more about the possibilities of using SCION Hidden Paths for additional DDoS protection within a SCION network, please contact customer-support@anapaya.net.

Scenario 4: The attacker has a target and due to the network topology, there are other organizations affected as collateral damage (e.g. an ISP is under attack).

Protection: If the organizations affected by the attack as collateral damage are part of the SCION network, traffic will automatically be rerouted to avoid congested areas under attack. This is one of the benefits of the SCION protocol, as path selection depends on static data, such as the user-defined policies, and real-time performance data, such as latency and jitter. In other words, in case of an attack on an ISP, paths through that ISP will be de-prioritized due to their bad performance.

34.8.2 Do Anapaya appliances encrypt the payload information?

Anapaya appliances do not encrypt the payload information. SCION is a network protocol so the primary focus is protecting the network and topology information. If the users desire to have their payload encrypted, then they must configure it in the applications they use. Please note that in the future, payload encryption might be added as a feature.

34.8.3 Why is the GATE solution secure and preferable to using a normal VPN connection?

Security has three aspects: Confidentiality, Integrity and Availability (CIA). Current VPN solutions can provide confidentiality and integrity as they encrypt traffic and perform integrity checks. However, they cannot guarantee availability because traffic is still routed through the public Internet. The Anapaya GATE solution guarantees availability by hiding the VPN connection from the Internet. In other words, the combination of VPN and the Anapaya GATE can provide full CIA security.

Another aspect in which the Anapaya GATE fortifies the system is routing security. As VPN traffic goes through the public Internet, it is susceptible to BGP hijacking attacks. With the Anapaya GATE solution, traffic goes from an ISP's internal network to the SCION network and vice versa, which offers BGP hijack resilience by design.

For further information, you can also refer to the How to protect company data while working from home and VPN connections need protection blog posts.

34.9 SCION Questions

34.9.1 How can I communicate with entities in ISDs my organization is not part of and I do not have trust?

For a host within an AS to be able and get paths to a remote ISD, it needs to have the valid Trust Root Configuration (TRC) for that ISD (and the ISDs it needs to traverse until it reaches the destination). The TRC of an ISD defines the trusted entities within an ISD, such as the core ASes and the Certificate Authorities. Based on this, the ASes can then fetch the necessary information to validate the received paths.

34.9.2 What happens if a link fails along the path that a packet is being forwarded through?

If there is a link failure along a path in use, the router that detects the link failure sends a SCION Control Message Protocol (SCMP) message to the sender, stating the location of the link failure.

Once the sender receives the SCMP packet, it will immediately switch to a path that is functioning.

For further information on how SCMP works, please refer to the official SCION SCMP documentation.

Note: For general questions about the SCION protocol, please refer to the official SCION documentation.

CHAPTER

THIRTYFIVE

GLOSSARY

Note: For terms related to the SCION protocol we use the glossary of The Complete Guide to SCION.

Anapaya Appliance
: The Anapaya Appliance is a software package implementing the SCION protocol. It can be configured as a CORE, EDGE or GATE.

Anapaya CORE
: *Description of Anapaya CORE product*

Anapaya EDGE
: *Description of Anapaya EDGE product*

Anapaya GATE
: *Description of Anapaya GATE product*

Autonomous System (AS).
: An Autonomous System is a network under a common administrative control. For example, the network of an Internet service provider, company, or university can constitute an AS. If an organizational entity operates multiple networks that are not directly connected through a local area network, then the different networks are considered different ASes in SCION.

Certificate Authority (CA)
: A Certificate Authority (CA) is a trusted entity that issues digital certificates. These certificates typically bind a domain name to a public key. In SCION these certificates are used to assure the authenticity of ASes.

Control Plane
: The SCION control plane is responsible for the propagation and discovery of network paths, i.e., for the exchange of routing information between network nodes. The control plane thus determines where traffic can be sent and deals with questions such as how routes are established, which paths exist, what quality individual links offer, etc. Within a SCION AS, such functionalities are carried out by the control service. Packet forwarding is instead a task pertaining to the data plane.

Core AS
: In SCION networks, a core AS is an AS which has some additional duties such as initiating the beaconing for path discovery. Furthermore, a core AS needs to manage and distribute the ISD's trust related information (TRC).

Data Plane
: The data plane (sometimes also referred to as the forwarding plane) is responsible for forwarding data packets that end hosts have injected into the network. After routing information has been disseminated by the control plane, packets are forwarded according to such information by the data plane.

IP-in-SCION-Tunneling
: The IP-in-SCION-Tunneling mechanism enables common IP traffic to be transported over SCION. A service

provider and service consumer can communicate via standard IP traffic that is tunneled through the SCION network.

Isolation Domain (ISD)
In SCION, autonomous systems (ASes) are organized into logical groups called isolation domains or ISDs. Each ISD consists of ASes that span an area with a uniform trust environment (i.e., a common jurisdiction). A possible model is for ISDs to be formed along national boundaries or federations of nations.

Path
In SCION, a network path is a defined route that a packet traverses. The path consists of a sequence of hops. Each hop consists of a SCION ISD-AS and an ingress and egress interface. Two parties that want to communicate use an end-to-end path, that is constructed based on a set of up to three path segments.

Path Segment
Path Segments are generated through the beaconing process, which is initiated by the core ASes. There exist three types of path segments: core, up and down path segments. A core path segment is a path between cores, while an up path segment is a path segment from a non-core AS to a core AS. Down path segments are the same as up path segments but in reverse direction. These path segments are used to form an end-to-end path which consist of maximum an up, a core and a down path segment.

Path-Segment Construction Beacon (PCB)
Core ASes generate PCBs to explore paths within their isolation domain (ISD) and among the different ISDs. ASes further propagate selected PCBs to their neighboring ASes. As a PCB traverses the network, it carries path segments, which can subsequently be used for traffic forwarding.

Public Key Infrastructure (PKI)
PKI enables parties to authenticate themselves to other parties without having to exchange secrets bilaterally. A certificate authority which all parties have to trust digitally signs certificates, which are used to prove authenticity to other parties. In SCION, PKI is used to authenticate ASes.

SCION
Scalability Control and Isolation on Next-Generation Networks (SCION) is the name of the inter network routing protocol.

Trust Root Configuration (TRC)
In SCION, the Trust Root Configuration is the anchor of trust in an ISD. It consists of a collection of signed certificates and policies. The TRC further specifies the core ASes, the CAs and the voting parties within the ISD.

Voting Quorum
The voting quorum is a trust root configuration (TRC) field that indicates the number of votes needed on a successor TRC for it to be verifiable. A voting quorum greater than one will thus prevent a single entity from creating a malicious TRC update.

INDEX