YANGcore User's Guide

Watsen Networks

April 24, 2025

Abstract

This documentation is for the YANGcore product by Watsen Networks.

This documentation is still a work-in-progress. Some sections clearly indicate when material is pending, but there are also some missing sections, and other sections may not have enough detail.

YANGcore is currently in its "pre-alpha" stage and details captured in this document may change.

Contents

1 Introduction **3**

2 API Introduction **3**
2.1 Northbound Interface Introduction . . . 3
2.2 Outbound Interface Introduction . . . 4

3 Getting Started **4**
3.1 Fetching Host-meta . . . 4
3.2 Fetching the RESTCONF Root Resource . . . 5
3.3 Fetch the YANG Library . . . 5
3.4 Get the Default Configuration . . . 8
3.5 Configure Users . . . 9
3.6 Delete the Second User . . . 11
3.7 Prep for TLS . . . 11
3.8 Add an HTTPS Endpoint . . . 12
3.9 Delete the HTTP Endpoint . . . 13
3.10 Get the New/Current Configuration . . . 14
3.11 Again, with a Single PUT . . . 15
3.12 View to Whole Audit Log . . . 17
3.13 View Last Three Audit Log Records . . . 18

4 Special Topics **19**
4.1 HTTP Headers . . . 19
4.2 Ordered Lists . . . 19
4.3 Special Characters . . . 19
4.4 List Pagination . . . 20
4.4.1 The "limit" Query Parameter . . . 20
4.4.2 The "offset" Query Parameter . . . 20
4.4.3 The "direction" Query Parameter . . . 20
4.5 Triggers . . . 22
4.5.1 Configuration-based . . . 22
4.5.2 Clock-based . . . 22
4.6 Plugins . . . 22
4.6.1 Installation . . . 22
4.6.2 Example . . . 22
4.6.3 Hot Reloading . . . 23

5 Northbound API Details **24**
5.1 YANG Library . . . 24
5.2 YANG Modules . . . 26
5.3 Complete Tree Diagram . . . 26
5.4 NBI Top-level Nodes . . . 28
5.4.1 /ietf-keystore:keystore . . . 29
5.4.2 /ietf-restconf-server:restconf-server . . . 29
5.4.3 /ietf-truststore:truststore . . . 30
5.4.4 /yangcore:audit-log . . . 30
5.4.5 /yangcore:dynamic-callouts . . . 31
5.4.6 /yangcore:plugins . . . 31
5.4.7 /yangcore:preferences . . . 31
5.4.8 /yangcore:users . . . 32

6 Outbound Interface Details **33**
6.1 Notifications . . . 33
6.1.1 The "user-password-aging" Notification . . . 33
6.2 Dynamic Callouts . . . 33
6.2.1 The "relay-notification-log-record" Dynamic Callout . . . 34
6.2.2 The "relay-audit-log-record" Dynamic Callout . . . 35

1 Introduction

YANGcore is an application server that uses YANG to define an API contract and RESTCONF to present a REST API.

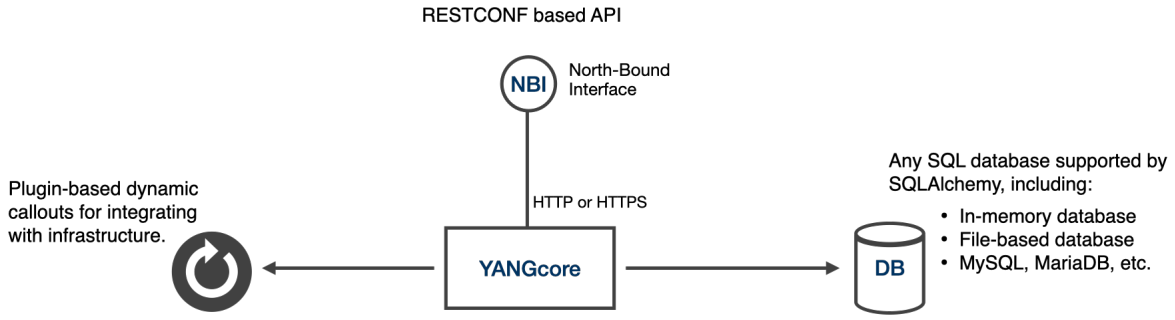
YANGcore alone, without any application level logic added, will present an RESTCONF-base API for the entire YANGcore data tree, as described in this document. Whilst YANGcore may run alone, it makes no sense to do so outside of familiarizing oneself with the YANGcore layer before familiarizing oneself with an application added on top of YANGcore.

Specifically, all the information provided in YANGcore-specific documentation is not expected to be repeated in application-specific documentation. Instead, it is expected that the application-specific documentation will point users to first understand how to install/use YANGcore.

This documentation is in preparation for a "1.0.0" release, using the common "major.minor.patch" semantic versioning convention. The use of versions like "0.0.N", where 'N' is an integer, should be read as "the Nth Pre-Alpha, with no statement about if it contains"major" or "minor" changes.

2 API Introduction

The following picture illustrates the API interfaces implemented by YANGcore.



The various aspects of this picture are discussed in the following sections.

2.1 Northbound Interface Introduction

YANGcore's Northbound interface (NBI) enables:

- the setting of configuration
- the viewing of operational state
- the execution of RPCs/actions

The NBI applies to the entire data model provided by YANGcore itself and any applications sitting on top of YANGcore. This guide only regards YANGcore's data model, defined by the YANG Library.

The best way to quickly understand the YANGcore NBI's data model (i.e., as defined by the YANG Library) is to look at the Complete Tree Diagram.

The NBI is accessed using the RESTCONF (RFC 8040) protocol. The NBI implements the Network Management Datastore Architecture (NMDA) (RFC 8342) or, more specifically, the RESTCONF Extensions to Support the Network Management Datastore Architecture (RFC 8527).

The maximum message size that a client can send is set 32MB.

See the Northbound API Details section for a description of each part of the northbound interface.

2.2 Outbound Interface Introduction

The Outbound Interface is for integration into backend system. The Outbound interface does not present an API to those system, but rather consumes the interfaces presented by those system.

Examples of Outbound interface include:

- YANGcore using an external database for persistent storage.
- YANGcore sending log records to an external logging system.
- YANGcore requesting an external authenticator to authenticate a user (soon to be implemented).
- YANGcore sending email via an external SMTP server (soon to be implemented).
- etc.

See the Outbound Interface Details section for more information about the Outbound interface.

3 Getting Started

This section presents a few examples illustrating some "startup" interactions with YANGcore. These examples assume YANGcore's NBI is listening on its default address and port: localhost:8080.

These examples use the ubiquitous 'curl' command line utility program for illustration purposes only. It is expected that production code would use a programming language to achieve the same result.

These examples assume a freshly installed YANGcore, i.e., using the default values described in "Defaults" section in the Installation Guide. To run a fresh instance of YANGcore, in one window, run the following command:

```
$ yangcore sqlite:///:memory:
```

After answering "Yes" to accept the "Non-production Use" contract, there will be no more output to the screen. Press ^C, or send any signal other than SIGHUP to the process, to cause YANGcore to grcefully exit.

The following subsections illustrate commands run from another window.

```
Warning! All of the following commands and responses were excuted/captured for this build of the
documentation using the current YANGcore server.  Copy/pasting is expected, but please be advised
that the hyphen and the single- and double- quote characters are incorrectly converted from plain
ASCII to their "fancy" UTF-8 equivalents.  They must be converted back to their ASCII forms for
these SHELL scripts to run.
```

3.1 Fetching Host-meta

RFC 8040 defines a discovery protocol for determining a RESTCONF server's root resource location. Note that YANGcore uses '/restconf' as the default root resource location. No "Content-Type" is required for this resource.

Request:

```
#!/bin/sh

# get server's response
curl -isf http://localhost:8080/.well-known/host-meta > output/get_host_meta.out

exit_code=$?
if [[ $exit_code != 0 ]]; then
  echo "curl returned an error"
  exit 1
fi
```


Output:

```
HTTP/1.1 200 OK
Content-Type: application/xrd+xml; charset=utf-8
Content-Length: 104
Date: Fri, 25 Apr 2025 01:50:49 GMT
Server: <redacted>

<XRD xmlns="http://docs.oasis-open.org/ns/xri/xrd-1.0">
  <Link rel="restconf" href="/restconf"/>
</XRD>
```

3.2 Fetching the RESTCONF Root Resource

RFC 8040 defines an ability to query the RESTCONF server's root resource to determine, for instance, what YANG-library version it is using.

Request:

```
================================== NOTE: '\' line wrapping per RFC 8792 ================================

#!/bin/sh

# get server's response
curl -isf -H "Accept:application/yang-data+json" http://localhost:8080/restconf > output/get_restconf_root.\
out

exit_code=$?
if [[ $exit_code != 0 ]]; then
  echo "curl returned an error"
  exit 1
fi
```

Output:

```
HTTP/1.1 200 OK
Content-Type: application/yang-data+json; charset=utf-8
Content-Length: 137
Date: Fri, 25 Apr 2025 01:50:49 GMT
Server: <redacted>

{
    "ietf-restconf:restconf" : {
        "data" : {},
        "operations" : {},
        "yang-library-version" : "2019-01-04"
    }
}
```

3.3 Fetch the YANG Library

The YANG Library is described in RFC 8525. RESTCONF clients may use it to understand what schema the server implements, which is critical to fully understanding the API.

Request:

```
================================== NOTE: '\' line wrapping per RFC 8792 ================================

#!/bin/sh

# get server's response
curl -isf -H "Accept:application/yang-data+json" http://localhost:8080/restconf/ds/ietf-datastores:operatio\
nal/ietf-yang-library:yang-library > output/get_yang_library.out
exit_code=$?
if [[ $exit_code != 0 ]]; then
  echo "curl returned an error"
  exit 1
fi
```


Output:

```
HTTP/1.1 200 OK
Content-Type: application/yang-data+json; charset=utf-8
Content-Length: 5295
Date: Fri, 25 Apr 2025 01:50:49 GMT
Server: <redacted>

{
  "ietf-yang-library:yang-library": {
    "module-set": [
      {
        "name": "shared-module-set",
        "module": [
          {
            "name": "ietf-datastores",
            "revision": "2018-02-14",
            "namespace": "urn:ietf:params:xml:ns:yang:ietf-yang-datastores"
          },
          {
            "name": "ietf-yang-library",
            "revision": "2019-01-04",
            "namespace": "urn:ietf:params:xml:ns:yang:ietf-yang-library"
          },
          {
            "name": "ietf-crypto-types",
            "revision": "2024-10-10",
            "feature": [
              "cleartext-private-keys"
            ],
            "namespace": "urn:ietf:params:xml:ns:yang:ietf-crypto-types"
          },
          {
            "name": "ietf-truststore",
            "revision": "2024-10-10",
            "feature": [
              "certificates",
              "central-truststore-supported"
            ],
            "namespace": "urn:ietf:params:xml:ns:yang:ietf-truststore"
          },
          {
            "name": "ietf-keystore",
            "revision": "2024-10-10",
            "feature": [
              "asymmetric-keys",
              "central-keystore-supported"
            ],
            "namespace": "urn:ietf:params:xml:ns:yang:ietf-keystore"
          },
          {
            "name": "ietf-tcp-common",
            "revision": "2024-10-10",
            "namespace": "urn:ietf:params:xml:ns:yang:ietf-tcp-common"
          },
          {
            "name": "ietf-tcp-client",
            "revision": "2024-10-10",
            "namespace": "urn:ietf:params:xml:ns:yang:ietf-tcp-client"
          },
          {
            "name": "ietf-tcp-server",
            "revision": "2024-10-10",
            "namespace": "urn:ietf:params:xml:ns:yang:ietf-tcp-server"
          },
          {
            "name": "iana-tls-cipher-suite-algs",
            "revision": "2024-10-16",
            "namespace": "urn:ietf:params:xml:ns:yang:iana-tls-cipher-suite-algs"
          },
          {
            "name": "ietf-tls-common",
            "revision": "2024-10-10",
            "namespace": "urn:ietf:params:xml:ns:yang:ietf-tls-common"
          },
          {
            "name": "ietf-tls-server",
            "revision": "2024-10-10",
            "feature": [
              "server-ident-x509-cert",
              "client-auth-supported",
              "client-auth-x509-cert"
            ],
            "namespace": "urn:ietf:params:xml:ns:yang:ietf-tls-server"
          },
          {
            "name": "ietf-udp-server",
```



```
          "revision": "2024-10-15",
          "feature": [],
          "namespace": "urn:ietf:params:xml:ns:yang:ietf-udp-server"
        },
        {
          "name": "ietf-http-server",
          "revision": "2025-04-24",
          "feature": [
            "client-auth-supported"
          ],
          "namespace": "urn:ietf:params:xml:ns:yang:ietf-http-server"
        },
        {
          "name": "ietf-restconf-server",
          "revision": "2025-04-24",
          "feature": [
            "listen",
            "tcp-listen",
            "tls-listen",
            "central-restconf-server-supported"
          ],
          "namespace": "urn:ietf:params:xml:ns:yang:ietf-restconf-server"
        },
        {
          "name": "yangcore",
          "revision": "2025-04-24",
          "namespace": "https://watsen.net/yangcore"
        },
        {
          "name": "yangcore-yl",
          "revision": "2025-04-24",
          "namespace": "https://watsen.net/yangcore"
        }
      ],
      "import-only-module": [
        {
          "name": "ietf-yang-types",
          "revision": "2013-07-15",
          "namespace": "urn:ietf:params:xml:ns:yang:ietf-yang-types"
        },
        {
          "name": "ietf-inet-types",
          "revision": "2013-07-15",
          "namespace": "urn:ietf:params:xml:ns:yang:ietf-inet-types"
        },
        {
          "name": "iana-crypt-hash",
          "revision": "2014-08-06",
          "namespace": "urn:ietf:params:xml:ns:yang:iana-crypt-hash"
        },
        {
          "name": "ietf-x509-cert-to-name",
          "revision": "2014-12-10",
          "namespace": "urn:ietf:params:xml:ns:yang:ietf-x509-cert-to-name"
        },
        {
          "name": "ietf-restconf",
          "revision": "2017-01-26",
          "namespace": "urn:ietf:params:xml:ns:yang:ietf-restconf"
        },
        {
          "name": "ietf-netconf-acm",
          "revision": "2018-02-14",
          "namespace": "urn:ietf:params:xml:ns:yang:ietf-netconf-acm"
        },
        {
          "name": "yangcore-common",
          "revision": "2025-04-24",
          "namespace": "https://watsen.net/yangcore-rpcs"
        }
      ]
    }
  ],
  "schema": [
    {
      "name": "shared-schema",
      "module-set": [
        "shared-module-set"
      ]
    }
  ],
  "datastore": [
    {
      "name": "ietf-datastores:operational",
      "schema": "shared-schema"
    },
    {
```



```
        "name": "ietf-datastores:running",
        "schema": "shared-schema"
      }
    ],
    "content-id": "TBD"
  }
}
```

3.4 Get the Default Configuration

The following example returns the entirety of the default configuration. Noteworthy points include:

- there is a single dual-stack port that listens on localhost:8080 for the "native-interface".
- the preferences allows the internal authenticator to use passwords.
- there are no preconfigured users.

This default is set because:

- Security best-practice advises against default accounts.
- Forcing the initial access over the loopback address asserts that installer was able to log into the host operating system, passing authentication and authorization policies used by the operating system..
- A user must be configured in the first write-request to the server (POST, PUT, PATCH, DELETE, but not GET or HEAD).
- The default allowing passwords primarily simplifies new user getting started with YANGcore. Specifically, it allows the first write-request to be a POST request to [Configure a User]. A seasoned user will use a single PUT request to replace the entire configuration in the >running< datastore. This PUT will fully replace the default configuration.

Request:

```
================================ NOTE: '\' line wrapping per RFC 8792 ================================

#!/bin/sh

# get server's response
curl -isf -H "Accept:application/yang-data+json" http://localhost:8080/restconf/ds/ietf-datastores:running \
> output/get_whole_config.out

exit_code=$?
if [[ $exit_code != 0 ]]; then
  echo "curl returned an error"
  exit 1
fi
```

Output:

```
HTTP/1.1 200 OK
Content-Type: application/yang-data+json; charset=utf-8
Content-Length: 800
Date: Fri, 25 Apr 2025 01:50:49 GMT
Server: <redacted>

{
  "ietf-restconf-server:restconf-server": {
    "listen": {
      "endpoints": {
        "endpoint": [
          {
            "name": "default startup endpoint",
            "yangcore:use-for": "native-interface",
            "http-over-tcp": {
              "tcp-server-parameters": {
                "local-bind": [
                  {
                    "local-address": "127.0.0.1",
                    "local-port": 8080
                  },
                  {
                    "local-address": "::1",
                    "local-port": 8080
                  }
                ]
              }
            }
```



```
          }
        ]
      }
    }
  },
  "yangcore:preferences": {
    "authentication": {
      "internal-authenticator": {
        "passwords-allowed": {}
      }
    }
  }
}
```

3.5 Configure Users

As was mentioned previously, a freshly installed YANGcore has no users configured. And yet YANGcore requires that a user always be configured. Special logic allows the assertion to be overlooked until the first write operation. Thus the first write operation to a freshly installed YANGcore instance must configure at least one user.

The following illustrates configuring two users in a POST request. It isn't necessary to configure more than one user, but the example does so to illustrate that the passwords may be pre-hashed (see the password for user "my-admin2").

Both users have "unrestricted" authorization, which is givan as it is the only authorization type currently supported by YANGcore. Be mindful that the "[null]" that appears in JSON files would be "[None]" in Python, in case the YANGcore-client is written in Python. Equivalent mapping must exist for other programming languages.

Request:

```
================================== NOTE: '\' line wrapping per RFC 8792 ================================

#!/bin/sh

TEMPDIR=`mktemp -d`

# initialize body for the PUT request
cat << EOM > $TEMPDIR/users.json
{
    "yangcore:users": {
        "user": [
            {
                "login": "my-admin",
                "email-address": "my-admin@example.com",
                "authentication": {
                    "password-based": {
                        "password": "\$0\$my-secret"
                    }
                },
                "authorization": {
                    "unrestricted": [null]
                }
            },
            {
                "login": "my-admin2",
                "email-address": "my-admin2@example.com",
                "authentication": {
                    "password-based": {
                        "password": "\$5\$rounds=1000\$xWJ2aHFewf.Q9foR\$/2HkIPNUgm2GWE0QStzHX.un1S03NWMeHR\
HxrWgN/w2"
                    }
                },
                "authorization": {
                    "unrestricted": [null]
                }
            }
        ]
    }
}
EOM

curl -isf -X POST --data @$TEMPDIR/users.json -H "Content-Type:application/yang-data+json" --user my-admin:\
my-secret http://localhost:8080/restconf/ds/ietf-datastores:running > output/post_two_users.out

exit_code=$?
if [[ $exit_code != 0 ]]; then
  echo "curl returned an error"
  exit 1
```



```
fi

rm -rf $TEMPDIR
```


Output:

```
HTTP/1.1 201 Created
Content-Length: 0
Date: Fri, 25 Apr 2025 01:50:49 GMT
Server: <redacted>
```

3.6 Delete the Second User

Assuming the second user wasn't needed, we can delete it as follows.

Note that it is now necessary to supply authenticaiton credentials (i.e. –user my-admin:my-secret). This because users were configured in the previous step.

Request:

```
================================ NOTE: '\' line wrapping per RFC 8792 ================================

#!/bin/sh

curl -isf -X DELETE --user my-admin:my-secret http://localhost:8080/restconf/ds/ietf-datastores:running/yan\
gcore:users/user=my-admin2 > output/delete_second_user.out

exit_code=$?
if [[ $exit_code != 0 ]]; then
  echo "curl returned an error"
  exit 1
fi
```

Output:

```
HTTP/1.1 204 No Content
Date: Fri, 25 Apr 2025 01:50:49 GMT
Server: <redacted>
```

3.7 Prep for TLS

This example adds a private asymmetric key and a matching certificate to the keystore. These two nodes are referenced in the next example.

Please note that the "pki" directory referenced in these examples is effectively the same as the "pki" directory found in the "SZTPD Simulator" package for the SZTPD application. SZTPD Simulator tarballs can be found here.

Please note that the binary values are the base64 of the DER (not the PEM) encodings. That is, the header and footer (e.g., '=====') lines are missing, and all the B64 characters are on one line.

The private and public key values are the native OpenSSL values for private and public keys. The certificate value is the degenerate form of a CMS (PKCS #7) commonly used to communicate a chain of certificates.

Request:

```
================================ NOTE: '\' line wrapping per RFC 8792 ================================

#!/bin/sh

# get a temp dir
TEMPDIR=`mktemp -d`

# The private and public keys
NBI_PRI_KEY_B64=`openssl enc -base64 -A -in pki/sztpd1/nbi/end-entity/private_key.der`
NBI_PUB_KEY_B64=`openssl enc -base64 -A -in pki/sztpd1/nbi/end-entity/public_key.der`

# The matching signed certificate
cat pki/sztpd1/nbi/end-entity/my_cert.pem pki/sztpd1/nbi/intermediate2/my_cert.pem > $TEMPDIR/cert_chain.pem
openssl crl2pkcs7 -nocrl -certfile $TEMPDIR/cert_chain.pem -outform DER -out $TEMPDIR/cert_chain.cms
NBI_EE_CERT_B64=`openssl enc -base64 -A -in $TEMPDIR/cert_chain.cms`

# initialize document for the POST request
cat << EOM > $TEMPDIR/keystore.json
{
    "ietf-keystore:keystore": {
        "asymmetric-keys": {
            "asymmetric-key": [
```



```
                {
                    "name": "nbi-server-end-entity-key",
                    "public-key-format": "ietf-crypto-types:subject-public-key-info-format",
                    "public-key": "$NBI_PUB_KEY_B64",
                    "private-key-format": "ietf-crypto-types:ec-private-key-format",
                    "cleartext-private-key": "$NBI_PRI_KEY_B64",
                    "certificates": {
                        "certificate": [
                            {
                                "name": "nbi-server-end-entity-cert",
                                "cert-data": "$NBI_EE_CERT_B64"
                            }
                        ]
                    }
                }
            ]
        }
    }
}
EOM

# POST to running
curl -isf -X POST --data @$TEMPDIR/keystore.json -H "Content-Type:application/yang-data+json" --user my-adm\
in:my-secret http://localhost:8080/restconf/ds/ietf-datastores:running > output/post_keystore.out
exit_code=$?
if [[ $exit_code != 0 ]]; then
  echo "curl returned an error"
  exit 1
fi

# remove temp dir
rm -rf $TEMPDIR
```

Output:

```
HTTP/1.1 201 Created
Content-Length: 0
Date: Fri, 25 Apr 2025 01:50:49 GMT
Server: <redacted>
```

3.8 Add an HTTPS Endpoint

This example illustrates telling YANGcore to open a new interface to listen on port 8443 for HTTPS connections. Be mindful that adding a new interface does not remove the only interface, the one listening on port 8080.

After invoking this command, the server sends a SIGHUP signal to itself to restart, as it must whenever any aspect of the server's interfaces change. It takes less than five seconds for the server to become available again.

Request:

```
================================ NOTE: '\' line wrapping per RFC 8792 ================================

#!/bin/sh

OPENSSL="openssl"
TEMPDIR=`mktemp -d`

# NBI Port
NBI_PORT="8443"

# initialize body for the PUT request
cat << EOM > $TEMPDIR/endpoint.json
{
    "ietf-restconf-server:endpoint": [
        {
            "name": "my native interface",
            "yangcore:use-for": "yangcore:native-interface",
            "http-over-tls": {
                "tcp-server-parameters": {
                    "local-bind": [
                        {
                            "local-address": "127.0.0.1",
                            "local-port": $NBI_PORT
                        },
                        {
                            "local-address": "::1",
                            "local-port": $NBI_PORT
                        }
```



```
                    ]
                },
                "tls-server-parameters": {
                    "server-identity": {
                        "certificate" : {
                            "central-keystore-reference": {
                                "asymmetric-key": "nbi-server-end-entity-key",
                                "certificate": "nbi-server-end-entity-cert"
                            }
                        }
                    }
                }
            }
        }
    ]
}
EOM

# POST to running/ietf-restconf-server:restconf-server/listen/endpoints/endpoint
curl -isf -X POST --data @$TEMPDIR/endpoint.json -H "Content-Type:application/yang-data+json" --user my-adm\
in:my-secret http://localhost:8080/restconf/ds/ietf-datastores:running/ietf-restconf-server:restconf-server\
/listen/endpoints > output/post_new_https_endpoint.out

exit_code=$?
if [[ $exit_code != 0 ]]; then
  echo "curl returned an error"
  exit 1
fi

rm -rf "$TEMPDIR"
```

Output:

```
HTTP/1.1 201 Created
Content-Length: 0
Date: Fri, 25 Apr 2025 01:50:49 GMT
Server: <redacted>
```

3.9 Delete the HTTP Endpoint

After having configured HTTPS, a common next action would be to test that the HTTPS interface is active and then, presumably, delete the no longer needed HTTP interface. This example rolls these two actions together into a single action of 1) using the HTTPS interface to 2) delete the old HTTP interface.

Note that, in order to use the HTTPS interface:

- the "--cacert nbi_trust_chain.pem" parameter is passed so that curl can authenticate the server's end-entity certficate. The file "nbi_trust_chain.pem" contains the certificate for the CA that signed the server's certificate.
- the HTTP "scheme" is now "https" (previously "http).
- the port is now "8443" (previously 8080)

This example also illustrates the need to percent-encode parts of the URL, as described in Section 3.5.3 of RFC 8040.

Again be mindful that, because this request modifies the server's interfaces, the server will be momentarily unavailable while it restarts.

Request:

```
================================ NOTE: '\' line wrapping per RFC 8792 ================================

#!/bin/sh

curl -isf -X DELETE --cacert nbi_trust_chain.pem --user my-admin:my-secret https://localhost:8443/restconf/\
ds/ietf-datastores:running/ietf-restconf-server:restconf-server/listen/endpoints/endpoint=default%20startup\
%20endpoint > output/delete_old_http_endpoint.out

exit_code=$?
if [[ $exit_code != 0 ]]; then
  echo "curl returned an error"
  exit 1
fi
```

Output:


```
HTTP/1.1 204 No Content
Date: Fri, 25 Apr 2025 01:50:54 GMT
Server: <redacted>
```

3.10 Get the New/Current Configuration

At this point, the YANGcore has been initialized well enough to allow remote connections, as the only way to access the server is over TLS with authenticated creditials.

It is a good time to take stock of the configuration. This example fetches the new whole configuration (i.e., the entirety of the >running< datastore).

Request:

```
================================ NOTE: '\' line wrapping per RFC 8792 ================================

#!/bin/sh

# get server's response
curl -isf -H "Accept:application/yang-data+json" --cacert nbi_trust_chain.pem --user my-admin:my-secret htt\
ps://localhost:8443/restconf/ds/ietf-datastores:running > output/get_whole_config_again.out

exit_code=$?
if [[ $exit_code != 0 ]]; then
  echo "curl returned an error"
  exit 1
fi
```

Output:

```
================================ NOTE: '\' line wrapping per RFC 8792 ================================

HTTP/1.1 200 OK
Content-Type: application/yang-data+json; charset=utf-8
Content-Length: 4356
Date: Fri, 25 Apr 2025 01:50:59 GMT
Server: <redacted>

{
  "ietf-restconf-server:restconf-server": {
    "listen": {
      "endpoints": {
        "endpoint": [
          {
            "name": "my native interface",
            "yangcore:use-for": "yangcore:native-interface",
            "http-over-tls": {
              "tcp-server-parameters": {
                "local-bind": [
                  {
                    "local-address": "127.0.0.1",
                    "local-port": 8443
                  },
                  {
                    "local-address": "::1",
                    "local-port": 8443
                  }
                ]
              },
              "tls-server-parameters": {
                "server-identity": {
                  "certificate": {
                    "central-keystore-reference": {
                      "asymmetric-key": "nbi-server-end-entity-key",
                      "certificate": "nbi-server-end-entity-cert"
                    }
                  }
                }
              }
            }
          }
        ]
      }
    }
  },
  "yangcore:preferences": {
    "authentication": {
      "internal-authenticator": {
        "passwords-allowed": {}
      }
    }
```



```
  },
  "yangcore:users": {
    "user": [
      {
        "login": "my-admin",
        "email-address": "my-admin@example.com",
        "authentication": {
          "password-based": {
            "password": "$5$rounds=1000$T7G7PnY1zSAAIo28$jKSDNT2Yq6GZz9jaN2iq/w7UqJSfWuPIDo7jLBigpj."
          }
        },
        "authorization": {
          "unrestricted": [
            null
          ]
        }
      }
    ]
  },
  "ietf-keystore:keystore": {
    "asymmetric-keys": {
      "asymmetric-key": [
        {
          "name": "nbi-server-end-entity-key",
          "public-key-format": "ietf-crypto-types:subject-public-key-info-format",
          "public-key": "MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEQXWtlzido8F4vSBcelKktLKthr4/cPeqx9/VY4MbJbGYdR\
XIVq6UiaKqPl5rWzXD0eR/k5YQkxFEdlWGDwZZTQ==",
          "private-key-format": "ietf-crypto-types:ec-private-key-format",
          "cleartext-private-key": "MHcCAQEEINAYdRRZonezDfVYob/NWVaA3Fj+1Kioq5gVBH0gIYs7oAoGCCqGSM49AwEHoUQ\
DQgAEQXWtlzido8F4vSBcelKktLKthr4/cPeqx9/VY4MbJbGYdRXIVq6UiaKqPl5rWzXD0eR/k5YQkxFEdlWGDwZZTQ==",
          "certificates": {
            "certificate": [
              {
                "name": "nbi-server-end-entity-cert",
                "cert-data": "MIIFeQYJKoZIhvcNAQcCoIIFajCCBWYCAQExADALBgkqhkiG9w0BBwGgggVOMIICoTCCAkegAwIBA\
gIBATAKBggqhkjOPQQDAjCBkTELMAkGA1UEBhMCWFgxHTAbBgNVBAgMFE15IFN0YXRlIG9yIFByb3ZpbmNlMRQwEgYDVQQHDAtNeSBMb2Nh\
bGl0eTEYMBYGA1UECgwPTXkgT3JnYW5penF0aW9uMRAwDgYDVQQLDAdNeSBVbml0MSEwHwYDVQQDDBhuYmkvc2VydmVyL2ludGVybWVkaWF\
0ZTIwIBcNMjUwNDI1MDE1MDQ0WhgPOTk5OTEyMzEyMzU5NTlaMG4xCzAJBgNVBAYTAlhYMR0wGwYDVQQIDBRNeSBTdGF0ZSBvciBQcm92aW\
5jZTEUMBIGA1UEBwwLTXkgTG9jYWxpdHkxGDAWBgNVBAoMD015IE9yZ2FuaXphdGlvbjEQMA4GA1UECwwHTXkgVW5pdDBZMBMGByqGSM49A\
gEGCCqGSM49AwEHA0IABEF1rZc4naPBeL0gXHpSpLSyrYa+P3D3qsff1WODGyWxmHUVyFaulImiqj5ea1s1w9Hkf5OWEJMRRHJVhg8GWU2j\
ga8wgawwHQYDVR0OBBYEFHi9hKtzuoAGO2w8JK55l0GoF3wLMB8GA1UdIwQYMBaAFIa5d9umr6nb2lwrNz8CIuMtPhZCMAkGA1UdEwQCMAA\
wQwYDVR0fBDwwOjA4oDagNIYyaHR0cDovL2NybC5leGFtcGxlLmNvbT9jYT1uYmk6c2VydmVyOmludGVybWVkaWF0ZTIwGgYDVR0RBBMwEY\
cEfwAAAYIJbG9jYWxob3N0MAoGCCqGSM49BAMCA0gAMEUCIQDbJd7JuYbYWdVMg1U7+0QUtkN9ks4+06B9871jc5e73gIgUJpOy3QT3e62a\
6+2jQLhgjVrpFEKCh/1gfpz9xxuMAQwggKlMIICS6ADAgECAgEBMAoGCCqGSM49BAMCMHsxCzAJBgNVBAYTAlhYMR0wGwYDVQQIDBRNeSBT\
dGF0ZSBvciBQcm92aW5jZTEYMBYGA1UECgwPTXkgT3JnYW5penF0aW9uMRAwDgYDVQQLDAdNeSBVbml0MSEwHwYDVQQDDBhuYmkvc2VydmV\
yL2ludGVybWVkaWF0ZTEwIBcNMjUwNDI1MDE1MDQ0WhgPOTk5OTEyMzEyMzU5NTlaMIGRMQswCQYDVQQGEwJYWDEdMBsGA1UECAwUTXkgU3\
RhdGUgb3IgUHJvdmluY2UxFDASBgNVBAcMC015IExvY2FsaXR5MRgwFgYDVQQKDA9NeSBPcmdhbml6YXRpb24xEDAOBgNVBAsMB015IFVua\
XQxDTAfBgNVBAMMGG5iaS9zZXJ2ZXIvaW50ZXJtZWRpYXRlMjBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABAfaCRGMzhwaU5GXHog8nhCM\
3oWhocHSB6U9DGHdyeawr1gT+6ka5WN4ecEt4J+oig7KlsDuTYcol0azlRR1QeSjgaYwgaMwHQYDVR0OBBYEFIa5d9umr6nb2lwrNz8CIuM\
tPhZCMB8GA1UdIwQYMBaAFPVMxTB8RQMy7CVAPxYs8wq4NREqMAwGA1UdEwQFMAMBAf8wDgYDVR0PAQH/BAQDAgEGMFMGA1UdHwQMMDowOK\
A2oDSGMmh0dHA6Ly9jcmwuZXhhbXBsZS5jb20/Y2E9bmJpOnNlcnZlcjppbnRlcm1lZGlhdGUxMAoGCCqGSM49BAMCA0gAMEUCIHFXQ8/uC\
xvKfVLE55LAfKTiHVqQuN3je9gVaZg6DrnAAiEAgJWGF9FR+P7W52SYrhWtEfJSjWi+28Ev+rwOGQKXq60xAA=="
              }
            ]
          }
        }
      ]
    }
  }
}
```

3.11 Again, with a Single PUT

When initializing a fresh system, rather then send a sequence of POST and DELETE messages, it is easiest to replace the entire contents of the >running< configuration datastore with a single PUT request to the server. In fact, this non-transactional behavior is one of the things that RESTCONF is famous for. The following single PUT achieves the same result as all of the above commands combined.

Note the following PUT document is identical to the output from the previous example. Note also that the system's defaults are included, as otherwise the system would complain that they are missing.

Again be mindful that, because this request modifies the server's interfaces, the server will be momentarily unavailable while it restarts.

Request:

```
================================ NOTE: '\' line wrapping per RFC 8792 ================================

#!/bin/sh

TEMPDIR=`mktemp -d`
```



```
# NBI: private key, public key, and cert
NBI_PRI_KEY_B64=`openssl enc -base64 -A -in pki/sztpd1/nbi/end-entity/private_key.der`
NBI_PUB_KEY_B64=`openssl enc -base64 -A -in pki/sztpd1/nbi/end-entity/public_key.der`
cat pki/sztpd1/nbi/end-entity/my_cert.pem pki/sztpd1/nbi/intermediate2/my_cert.pem > $TEMPDIR/cert_chain.pem
openssl crl2pkcs7 -nocrl -certfile $TEMPDIR/cert_chain.pem -outform DER -out $TEMPDIR/cert_chain.cms
NBI_EE_CERT_B64=`openssl enc -base64 -A -in $TEMPDIR/cert_chain.cms`

# initialize body for the PUT request
cat << EOM > $TEMPDIR/running.json
{
    "ietf-restconf-server:restconf-server": {
        "listen": {
            "endpoints": {
                "endpoint": [
                    {
                        "name": "my native interface",
                        "yangcore:use-for": "yangcore:native-interface",
                        "http-over-tls": {
                            "tcp-server-parameters": {
                                "local-bind": [
                                    {
                                        "local-address": "127.0.0.1",
                                        "local-port": 8443
                                    },
                                    {
                                        "local-address": "::1",
                                        "local-port": 8443
                                    }
                                ]
                            },
                            "tls-server-parameters": {
                                "server-identity": {
                                    "certificate" : {
                                        "central-keystore-reference": {
                                            "asymmetric-key": "nbi-server-end-entity-key",
                                            "certificate": "nbi-server-end-entity-cert"
                                        }
                                    }
                                }
                            }
                        }
                    }
                ]
            }
        }
    },
    "yangcore:preferences": {
        "authentication": {
            "internal-authenticator": {
                "passwords-allowed": {}
            }
        }
    },
    "yangcore:users": {
        "user": [
            {
                "login": "my-admin",
                "email-address": "my-admin@example.com",
                "authentication": {
                    "password-based": {
                        "password": "\$5\$rounds=1000\$NqyUnxlg0qbwXNbO\$eDiLDpJ9KyjXyuvu8iq3dZP0h.0tUQobgy\
ZWoTlJJs0"
                    }
                },
                "authorization": {
                    "unrestricted": [null]
                }
            }
        ]
    },
    "ietf-keystore:keystore": {
        "asymmetric-keys": {
            "asymmetric-key": [
                {
                    "name": "nbi-server-end-entity-key",
                    "public-key-format": "ietf-crypto-types:subject-public-key-info-format",
                    "public-key": "$NBI_PUB_KEY_B64",
                    "private-key-format": "ietf-crypto-types:ec-private-key-format",
                    "cleartext-private-key": "$NBI_PRI_KEY_B64",
                    "certificates": {
                        "certificate": [
                            {
                                "name": "nbi-server-end-entity-cert",
                                "cert-data": "$NBI_EE_CERT_B64"
                            }
                        ]
```



```
                    }
                }
            ]
        }
    }
}
EOM

# PUT running
curl -if -X PUT --data @$TEMPDIR/running.json -H "Content-Type:application/yang-data+json" --cacert nbi_tru\
st_chain.pem --user my-admin:my-secret https://localhost:8443/restconf/ds/ietf-datastores:running > output/\
put_whole_config.out

exit_code=$?
if [[ $exit_code != 0 ]]; then
  echo "curl returned an error"
  exit 1
fi

# cleanup temp dir
rm -rf $TEMPDIR
```

Output:

```
HTTP/1.1 204 No Content
Date: Fri, 25 Apr 2025 01:50:59 GMT
Server: <redacted>
```

3.12 View to Whole Audit Log

This example illustrates the fetching of the whole Audit Log. Notice how all the commands run above appear. The audit log captures all interactions on the NBI interface.

Please be mindful that the Audit Log currently does not record read-only commands (i.e., GET and HEAD) requests. It is planned to add a tunable knob to toggle if read-only commands are recorded also.

```
================================ NOTE: '\' line wrapping per RFC 8792 ================================

#!/bin/sh

curl -isf -X GET -H "Accept:application/yang-data+json" --cacert nbi_trust_chain.pem --user my-admin:my-se\
cret https://localhost:8443/restconf/ds/ietf-datastores:operational/yangcore:audit-log > output/get_audit_l\
og.out

exit_code=$?
if [[ $exit_code != 0 ]]; then
  echo "curl returned an error"
  exit 1
fi
```

Output:

```
================================ NOTE: '\' line wrapping per RFC 8792 ================================

HTTP/1.1 200 OK
Content-Type: application/yang-data+json; charset=utf-8
Content-Length: 1717
Date: Fri, 25 Apr 2025 01:51:04 GMT
Server: <redacted>

{
  "yangcore:audit-log": {
    "audit-log-record": [
      {
        "timestamp": "2025-04-25T01:50:49Z",
        "source-ip": "::1",
        "host": "localhost:8080",
        "method": "POST",
        "path": "/restconf/ds/ietf-datastores:running",
        "outcome": "success",
        "comment": "No authorization required for fresh installs."
      },
      {
        "timestamp": "2025-04-25T01:50:49Z",
        "source-ip": "::1",
        "host": "localhost:8080",
        "method": "DELETE",
        "path": "/restconf/ds/ietf-datastores:running/yangcore:users/user=my-admin2",
        "outcome": "success"
      },
```



```
      {
        "timestamp": "2025-04-25T01:50:49Z",
        "source-ip": "::1",
        "host": "localhost:8080",
        "method": "POST",
        "path": "/restconf/ds/ietf-datastores:running",
        "outcome": "success"
      },
      {
        "timestamp": "2025-04-25T01:50:49Z",
        "source-ip": "::1",
        "host": "localhost:8080",
        "method": "POST",
        "path": "/restconf/ds/ietf-datastores:running/ietf-restconf-server:restconf-server/listen/endpoints\
",
        "outcome": "success"
      },
      {
        "timestamp": "2025-04-25T01:50:54Z",
        "source-ip": "::1",
        "host": "localhost:8443",
        "method": "DELETE",
        "path": "/restconf/ds/ietf-datastores:running/ietf-restconf-server:restconf-server/listen/endpoints\
/endpoint=default startup endpoint",
        "outcome": "success"
      },
      {
        "timestamp": "2025-04-25T01:50:59Z",
        "source-ip": "::1",
        "host": "localhost:8443",
        "method": "PUT",
        "path": "/restconf/ds/ietf-datastores:running",
        "outcome": "success"
      }
    ]
  }
}
```

3.13 View Last Three Audit Log Records

This closing example is the same as the previous example, but it illustrates the use of some query parameters to return just the last three audit log records.

Note that the Audit Log is traversed in the "backwards" direction and then limited to showing the first three records.

```
================================== NOTE: '\' line wrapping per RFC 8792 ================================

#!/bin/sh

curl -isf -X GET -H "Accept:application/yang-data+json" --cacert nbi_trust_chain.pem --user my-admin:my-se\
cret https://localhost:8443/restconf/ds/ietf-datastores:operational/yangcore:audit-log/audit-log-record?dir\
ection=backwards\&limit=3 > output/get_last_three_audit_log_records.out

exit_code=$?
if [[ $exit_code != 0 ]]; then
  echo "curl returned an error"
  exit 1
fi
```

Output:

```
================================== NOTE: '\' line wrapping per RFC 8792 ================================

HTTP/1.1 200 OK
Content-Type: application/yang-data+json; charset=utf-8
Content-Length: 839
Date: Fri, 25 Apr 2025 01:51:04 GMT
Server: <redacted>

{
  "yangcore:audit-log-record": [
    {
      "timestamp": "2025-04-25T01:50:59Z",
      "source-ip": "::1",
      "host": "localhost:8443",
      "method": "PUT",
      "path": "/restconf/ds/ietf-datastores:running",
      "outcome": "success"
    },
    {
```



```
      "timestamp": "2025-04-25T01:50:54Z",
      "source-ip": "::1",
      "host": "localhost:8443",
      "method": "DELETE",
      "path": "/restconf/ds/ietf-datastores:running/ietf-restconf-server:restconf-server/listen/endpoints/e\
ndpoint=default startup endpoint",
      "outcome": "success"
    },
    {
      "timestamp": "2025-04-25T01:50:49Z",
      "source-ip": "::1",
      "host": "localhost:8080",
      "method": "POST",
      "path": "/restconf/ds/ietf-datastores:running/ietf-restconf-server:restconf-server/listen/endpoints",
      "outcome": "success"
    }
  ]
}
```

4 Special Topics

This section goes over standards-based behavior that may not be immediately obvious to some readers.

4.1 HTTP Headers

YANGcore attempts to not require HTTP headers (e.g., "Accept", "Content-Type", etc.), but it strictly enforces that the correct headers are present when needed.

When an "Accept" header is not passed, an implied value will be derived from the "Content-Type" header, if it is present. Errors are returned using the implied value, or "text/plain" otherwise.

YANGcore does not have a default encoding, and so wildcard "Accept" header values, such as "/" or similar, are not supported. When the request demands a response (e.g., GET), an "Accept" header must be provided.

YANGcore is not able to auto-detect the encoding of request bodies, and thus a "Content-Type" header must be provided when a request body is included in a request.

4.2 Ordered Lists

An ordered list is one in which the order matters. In YANG, the 'ordered-by user' statement is used to indicate when the order matters for a 'list' or 'leaf-list'. Please see Section 7.7.1 in the YANG specification for details.

There are currently no ordered lists in YANGcore itself, but ordered lists exist in applications that are built on top of YANGcore.

The ordering of a user-ordered list is initiailly specified when the list is created, assuming more than one entry is defined at that time, and maintained thereafter whenever new entries are added to the list (e.g., using the POST method) and/or replaced (i.e., using the PUT method). In both cases, the 'insert' and 'point' query parameters are used, as described in Section 4.8.5 and Section 4.8.6 of the RESTCONF specification.

4.3 Special Characters

Special characters are herein defined as those that must be escaped in order to be passed into or out of the server.

According to Section 3.5.3 of RFC 8040, any reserved characters in a resource identifier (i.e., key values) MUST be percent-encoded.

This applies to any URL that encodes a 'list' or 'leaf-list' instance, and the 'key' value contains reserved characters (e.g., SPACE), they must be escaped when placed into the URL.

FWIW, a worst-case scenario arises when trying the insert or move an entry in a list whose keys are themselves URLs. This scenario is tested for.

4.4 List Pagination

Some lists, especially lists representing time-series data (i.e., logs), can grow to be large in size. Client applications wishing to browse thru the list entries may only wish to do so in chunks or pages.

There is an effort, spearheaded by Watsen Networks, to define a standard for list pagination: https://datatracker.ietf.org/doc/draft-ietf-netconf-list-pagination-rc.

YANGcore supports three query parameters:

Name	Methods	Description
limit	GET, HEAD	Limits the number of entries returned. If not specified, the number of entries that may be returned in unbounded.
offset	GET, HEAD	Indicates the number of entries in the result set that should the skipped over when preparing the response. If not specified, then no entries in the result set are skipped.
direction	GET, HEAD	Indicates the direction that the result set is to be traversed. If not specified, then the result set is traversed in the "forward" direction.

4.4.1 The "limit" Query Parameter

- The "limit" query parameter limits the number of entries from the working result set (i.e., after the "offset" parameter has been applied) that are returned.
- The allowed values are positive integers.
- This parameter is only allowed for the GET and HEAD methods on "list" and "leaf-list" data resources. A "400 Bad Request" status-line is returned if used for other methods or on other resource types.
- If this query parameter is not present, the number of entries that may be returned is unbounded.
- This query parameter MUST be supported for all lists and leaf-lists.

4.4.2 The "offset" Query Parameter

- The "offset" query parameter indicates the number of entries in the working result set (i.e., after the "direction" parameter has been applied) that should be skipped.
- The allowed values are positive integers. If the skip value exceeds the number of entries in the working result set, then "416 Range Not Satisfiable" status-line is returned.
- This parameter is only allowed for the GET and HEAD methods on
- This parameter is only allowed for the GET and HEAD methods on "list" and "leaf-list" data resources. A "400 Bad Request" status-line is returned if used for other methods or on other resource types.
- If this query parameter is not present, the default value is to not skip over any values from the working result set.
- This query parameter MUST be supported for all lists and leaf-lists.

4.4.3 The "direction" Query Parameter

- The "direction" query parameter indicates how the entries in the working result set should be traversed.
- The allowed values are:
  - forwards
    - Return entries in the "forward" direction. Also known as the "default" or "ascending" direction.
  - backwards
    - Return entries in the "backward" direction. Also known as the "reversed" or "descending" direction.

- This parameter is only allowed for the GET and HEAD methods on "list" and "leaf-list" data resources. A "400 Bad Request" status-line is returned if used for other methods or on other resource types.
- If this query parameter is not present, the default value is "forwards".
- This query parameter MUST be supported for all lists and leaf-lists.

4.5 Triggers

YANGcore reacts to some triggers automatically.

4.5.1 Configuration-based

YANGcore reacts to some configurations changes. Simple examples include:

- automatically hash client passwords.
- automatically SIGHUP when 'restconf-server' is configured.
- incrementing/decrementing opstate counters.

4.5.2 Clock-based

YANGcore reacts to some events based on time. An example is:

- expiring a password after a set amount of time.

4.6 Plugins

YANGcore uses plugins to enable custom logic for Dynamic Callouts.

4.6.1 Installation

Configuring YANGcore to use a plugin is a two-step process:

1. Place the Python module (i.e. file) in the "plugins" directory on the YANGcore server[1]. The location of this directory can be determined via the following command[2]:

```
python -c 'import importlib.resources as resources; print(resources.files("yangcore") / "plugins")'
```

2. Configure a "plugin" under "/plugins"[3].

4.6.2 Example

Assuming a plugin is installed as follows:

```
PLUGIN_DIR=`python -c 'import importlib.resources as resources; print(resources.files("yangcore") / "plugins")'`

cat << EOM > $PLUGIN_DIR/my-plugin.py
import sys
async def relay_audit_log_record_function(input, opaque):
    print("Inside relay_audit_log_record_function()...")
    print("input = " + str(input))
    print("opaque = " + str(opaque))
    print("sys.version = " + sys.version)
EOM
```

And the configuration contains the following, in addition to other configurations:

```
"yangcore:preferences": {
    "outbound-interactions": {
        "relay-audit-log-record-callout": "my-relay-audit-log-record-callout"
    }
},
"yangcore:plugins": {
    "plugin": [
        {
            "name": "my-plugin",
            "functions": {
                "function": [
                    {
                        "name": "relay_audit_log_record_function"
                    }
                ]
            }
```

[1] This step entails file-system level access to the YANGcore server.

[2] Ensure that you run the command using the same version of Python and/or Python virtual environment that YANGcore uses.

[3] In a multi-tenant deployment, only host-level users have access to this configuration; tenants can only use plugins that have been shared by host-level users.


```
        }
    ]
},
"yangcore:dynamic-callouts": {
    "dynamic-callout": [
        {
            "name": "my-relay-audit-log-record-callout",
            "rpc-supported": "yangcore-rpcs:relay-audit-log-record",
            "call-function": {
                "plugin": "my-plugin",
                "function": "relay_audit_log_record_function"
            },
            "opaque": {
                "empty": [None],
                "boolean": True,
                "string": "foobar",
                "list": [ 1, 2, 3 ],
                "dict": {
                    "a": 1,
                    "b": 2
                }
            }
        }
    ]
}
```

Then, when a device sends a progress report, the following may appear in the console where YANGcore is running[4]:

```
============================== Note: '\' line wrapping per RFC 8792] ================================

Inside relay_audit_log_record_function()...
input = { "audit-log-record": { "timestamp": "2025-01-22T00:09:46Z", "source-ip": "127.0.0.1", "host\
": "localhost:8080", "method": "DELETE", "path": "/restconf/ds/ietf-datastores:running/yangcore:user\
s/user=my-admin", "outcome": "success" } }
opaque = {'empty': [None], 'boolean': True, 'string': 'foobar', 'list': [1, 2, 3], 'dict': {'a': 1, \
'b': 2}}
sys.version = 3.8.2 (default, Nov 11 2020, 16:34:19)
[Clang 12.0.0 (clang-1200.0.32.21)]
```

4.6.3 Hot Reloading

It is possible to reload a plugin without restarting the server by replacing the contents on the Python file. The server will reload the plugin in a few seconds.

Warning! Be sure to test the plugin on a non-production server before reloading it. Because the reload happens outside of an NBI transaction, the normal verification tests cannot prevent a bad plugin from being used, potentially leading to loss of data.

[4]The "BASE64VALUE=" values are not real; they are used in this example only for readability.

5 Northbound API Details

5.1 YANG Library

For any RESTCONF server, its YANG Library is the heart of the API, an it unambiguously identifies what API the server implements.

YANGcore's YANG Library, presented below, indicates that two datastores are implemented (>running< and >operational<), each implementing the same set of YANG Modules. Per the Network Management Datastore Architecture (NMDA) (RFC 8342), the >operational< datastore presents the "operational" values for configuration, in case there may be a discrepancy.

Currently, in YANGcore and its applications, no discrepancy can arise, and so the >operational< datastore is primarily used to get operational state values (i.e., the "config false" nodes in the YANG modules) and to get a complete dump of the database's contents, useful for backups that need to contain all data.

Note: The YANG Library presented below is the same as that presented in the section Fetch the YANG Library.


```
{
    "ietf-yang-library:yang-library": {
        "module-set": [
            {
                "name": "shared-module-set",
                "module": [
                    {
                        "name": "ietf-datastores",
                        "revision": "2018-02-14",
                        "namespace": "urn:ietf:params:xml:ns:yang:ietf-yang-datastores"
                    },
                    {
                        "name": "ietf-yang-library",
                        "revision": "2019-01-04",
                        "namespace": "urn:ietf:params:xml:ns:yang:ietf-yang-library"
                    },
                    {
                        "name": "ietf-crypto-types",
                        "revision": "2024-10-10",
                        "feature": [
                            "cleartext-private-keys"
                        ],
                        "namespace": "urn:ietf:params:xml:ns:yang:ietf-crypto-types"
                    },
                    {
                        "name": "ietf-truststore",
                        "revision": "2024-10-10",
                        "feature": [
                            "certificates",
                            "central-truststore-supported"
                        ],
                        "namespace": "urn:ietf:params:xml:ns:yang:ietf-truststore"
                    },
                    {
                        "name": "ietf-keystore",
                        "revision": "2024-10-10",
                        "feature": [
                            "asymmetric-keys",
                            "central-keystore-supported"
                        ],
                        "namespace": "urn:ietf:params:xml:ns:yang:ietf-keystore"
                    },
                    {
                        "name": "ietf-tcp-common",
                        "revision": "2024-10-10",
                        "namespace": "urn:ietf:params:xml:ns:yang:ietf-tcp-common"
                    },
                    {
                        "name": "ietf-tcp-client",
                        "revision": "2024-10-10",
                        "namespace": "urn:ietf:params:xml:ns:yang:ietf-tcp-client"
                    },
                    {
                        "name": "ietf-tcp-server",
                        "revision": "2024-10-10",
                        "namespace": "urn:ietf:params:xml:ns:yang:ietf-tcp-server"
                    },
                    {
                        "name": "iana-tls-cipher-suite-algs",
                        "revision": "2024-10-16",
                        "namespace": "urn:ietf:params:xml:ns:yang:iana-tls-cipher-suite-algs"
                    },
```



```
                {
                    "name": "ietf-tls-common",
                    "revision": "2024-10-10",
                    "namespace": "urn:ietf:params:xml:ns:yang:ietf-tls-common"
                },
                {
                    "name": "ietf-tls-server",
                    "revision": "2024-10-10",
                    "feature": [
                        "server-ident-x509-cert",
                        "client-auth-supported",
                        "client-auth-x509-cert"
                    ],
                    "namespace": "urn:ietf:params:xml:ns:yang:ietf-tls-server"
                },
                {
                    "name": "ietf-udp-server",
                    "revision": "2024-10-15",
                    "feature": [],
                    "namespace": "urn:ietf:params:xml:ns:yang:ietf-udp-server"
                },
                {
                    "name": "ietf-http-server",
                    "revision": "2025-04-24",
                    "feature": [
                        "client-auth-supported"
                    ],
                    "namespace": "urn:ietf:params:xml:ns:yang:ietf-http-server"
                },
                {
                    "name": "ietf-restconf-server",
                    "revision": "2025-04-24",
                    "feature": [
                        "listen",
                        "tcp-listen",
                        "tls-listen",
                        "central-restconf-server-supported"
                    ],
                    "namespace": "urn:ietf:params:xml:ns:yang:ietf-restconf-server"
                },
                {
                    "name": "yangcore",
                    "revision": "2025-04-24",
                    "namespace": "https://watsen.net/yangcore"
                },
                {
                    "name": "yangcore-yl",
                    "revision": "2025-04-24",
                    "namespace": "https://watsen.net/yangcore"
                }
            ],
            "import-only-module": [
                {
                    "name": "ietf-yang-types",
                    "revision": "2013-07-15",
                    "namespace": "urn:ietf:params:xml:ns:yang:ietf-yang-types"
                },
                {
                    "name": "ietf-inet-types",
                    "revision": "2013-07-15",
                    "namespace": "urn:ietf:params:xml:ns:yang:ietf-inet-types"
                },
                {
                    "name": "iana-crypt-hash",
                    "revision": "2014-08-06",
                    "namespace": "urn:ietf:params:xml:ns:yang:iana-crypt-hash"
                },
                {
                    "name": "ietf-x509-cert-to-name",
                    "revision": "2014-12-10",
                    "namespace": "urn:ietf:params:xml:ns:yang:ietf-x509-cert-to-name"
                },
                {
                    "name": "ietf-restconf",
                    "revision": "2017-01-26",
                    "namespace": "urn:ietf:params:xml:ns:yang:ietf-restconf"
                },
                {
                    "name": "ietf-netconf-acm",
                    "revision": "2018-02-14",
                    "namespace": "urn:ietf:params:xml:ns:yang:ietf-netconf-acm"
                },
                {
                    "name": "yangcore-common",
                    "revision": "2025-04-24",
                    "namespace": "https://watsen.net/yangcore-rpcs"
                }
```



```
                ]
            }
        ],
        "schema": [
            {
                "name": "shared-schema",
                "module-set": [
                    "shared-module-set"
                ]
            }
        ],
        "datastore": [
            {
                "name": "ietf-datastores:operational",
                "schema": "shared-schema"
            },
            {
                "name": "ietf-datastores:running",
                "schema": "shared-schema"
            }
        ],
        "content-id": "TBD"
    }
}
```

5.2 YANG Modules

The YANG Library defines the API as a collection of YANG modules. There are twenty-three YANG modules in YANGcore. The most exact description of the API is had by reviewing these YANG modules directly, in conjuction with the "features" statements in the YANG Library.

Ideally this guide would include the twenty-three YANG modules used by YANGcore, but they are a combined total of over twelve thousand lines, which is too much to include here.

In lieu of including the YANG modules in this guide, the reader may find these YANG modules in the "yang" directory for the 'yangcore' packages. Where these files are located varies by Python installation. The following command find the directory the YANG files are located for the "yangcore" package:

```
python -c 'import importlib.resources as resources; print(resources.files("yangcore") / "yang")'
```

For example: "/Users/kent/.pyenv/versions/3.12.8/lib/python3.12/site-packages/yangcore/yang".

5.3 Complete Tree Diagram

This section presents the complete YANG tree diagram for YANGcore's Northbound API. This tree diagram is generated using the YANG library presented in YANG Library. This diagram is useful as it presents the entire API one diagram.

```
================================ NOTE: '\' line wrapping per RFC 8792 ================================

+--ro yangcore:audit-log
|  +--ro audit-log-record*
|     +--ro comment? <string>
|     +--ro host <string>
|     +--ro input?
|     +--ro method <enumeration>
|     +--ro outcome <enumeration>
|     +--ro path <string>
|     +--ro source-ip <ip-address(union)>
|     +--ro source-proxies* <string>
|     +--ro timestamp <date-and-time(string)>
|     +--ro username? <string>
+--rw yangcore:dynamic-callouts
|  +--rw dynamic-callout* [name]
|     +--rw (callout-type)
|     |  +--:(call-function)
|     |     +--rw call-function
|     |        |  +--rw function <leafref>
|     |        |  +--rw plugin <leafref>
|     |     +--rw opaque?
|     +--rw name <string>
|     +--rw rpc-supported <identityref>
+--rw ietf-keystore:keystore
|  +--rw asymmetric-keys
|     +--rw asymmetric-key* [name]
|        +--rw certificates
|        |  +--rw certificate* [name]
```



```
|       |     +--rw cert-data <end-entity-cert-cms(binary)>
|       |     +--rw name <string>
|       +--rw name <string>
|       +--rw private-key-format? <identityref>
|       +--rw (private-key-type)
|       |  +--:(cleartext-private-key)
|       |     +--rw cleartext-private-key? <binary>
|       +--rw public-key? <binary>
|       +--rw public-key-format? <identityref>
+--ro yangcore:notification-log
|  +--ro notification-log-record*
|     +--ro notification
|     +--ro timestamp <date-and-time(string)>
+--rw yangcore:plugins
|  +--rw plugin* [name]
|     +--rw functions
|     |  +--rw function* [name]
|     |     +--rw name <string>
|     +--rw name <string>
+--rw yangcore:preferences
|  +--rw accounting
|  +--rw authentication
|  |  +--rw internal-authenticator!
|  |     +--rw passwords-allowed!
|  |        +--rw aging-timeout!
|  |        |  +--rw amount? <uint16>
|  |        |  +--rw units? <enumeration>
|  |        +--rw is-required? <boolean>
|  +--rw authorization
|  +--rw general
|  |  +--rw email-address? <string>
|  |  +--rw hostname? <host(union)>
|  +--rw outbound-interactions
|     +--rw relay-audit-log-record-callout? <leafref>
|     +--rw relay-notification-log-record-callout? <leafref>
+--rw ietf-restconf-server:restconf-server
|  +--rw listen!
|     +--rw endpoints
|        +--rw endpoint* [name]
|           +--rw name <string>
|           +--rw (transport)
|           |  +--:(http-over-tcp)
|           |  |  +--rw http-over-tcp
|           |  |     +--rw external-endpoint!
|           |  |     |  +--rw address <host(union)>
|           |  |     |  +--rw client-cert-var? <string>
|           |  |     |  +--rw port? <port-number(uint16)>
|           |  |     |  +--rw trusted-proxy-count? <uint8>
|           |  |     +--rw http-server-parameters
|           |  |     |  +--rw client-authentication!
|           |  |     |  +--rw server-name? <string>
|           |  |     +--rw restconf-server-parameters
|           |  |     |  +--rw client-identity-mappings
|           |  |     |     +--rw cert-to-name* [id]
|           |  |     |        +--rw fingerprint? <tls-fingerprint(string)>
|           |  |     |        +--rw id <uint32>
|           |  |     |        +--rw map-type <identityref>
|           |  |     |        +--rw name <string>
|           |  |     +--rw tcp-server-parameters
|           |  |        +--rw local-bind* [local-address]
|           |  |           +--rw local-address <ip-address(union)>
|           |  |           +--rw local-port? <port-number(uint16)>
|           |  +--:(http-over-tls)
|           |     +--rw http-over-tls
|           |        +--rw http-server-parameters
|           |        |  +--rw client-authentication!
|           |        |  +--rw server-name? <string>
|           |        +--rw restconf-server-parameters
|           |        |  +--rw client-identity-mappings
|           |        |     +--rw cert-to-name* [id]
|           |        |        +--rw fingerprint? <tls-fingerprint(string)>
|           |        |        +--rw id <uint32>
|           |        |        +--rw map-type <identityref>
|           |        |        +--rw name <string>
|           |        +--rw tcp-server-parameters
|           |        |  +--rw local-bind* [local-address]
|           |        |     +--rw local-address <ip-address(union)>
|           |        |     +--rw local-port? <port-number(uint16)>
|           |        +--rw tls-server-parameters
|           |           +--rw client-authentication!
|           |           |  +--rw ca-certs!
|           |           |  |  +--rw (inline-or-truststore)
|           |           |  |     +--:(central-truststore)
|           |           |  |        +--rw central-truststore-reference? <central-certificate-bag-ref(leafre\
f)>
|           |           |  +--rw ee-certs!
|           |           |     +--rw (inline-or-truststore)
```



```
|           |           |        +--:(central-truststore)
|           |           |           +--rw central-truststore-reference? <central-certificate-bag-ref(leafre\
f)>
|           |          +--rw server-identity
|           |             +--rw (auth-type)
|           |                +--:(certificate)
|           |                   +--rw certificate
|           |                      +--rw (inline-or-keystore)
|           |                         +--:(central-keystore)
|           |                            +--rw central-keystore-reference
|           |                               +--rw asymmetric-key? <central-asymmetric-key-ref(leafref)>
|           |                               +--rw certificate? <leafref>
|          +--rw yangcore:use-for? <identityref>
+--rw ietf-truststore:truststore
|  +--rw certificate-bags
|     +--rw certificate-bag* [name]
|        +--rw certificate* [name]
|        |  +--rw cert-data <trust-anchor-cert-cms(binary)>
|        |  +--rw name <string>
|        +--rw description? <string>
|        +--rw name <string>
+--rw yangcore:users
|  +--rw user* [login]
|     +--rw authentication
|     |  +--rw password-based
|     |     +--rw password? <crypt-hash(string)>
|     |     +--ro password-last-modified? <date-and-time(string)>
|     +--rw authorization
|     |  +--rw (auth-type)
|     |     +--:(unrestricted)
|     |        +--rw unrestricted? <empty>
|     +--rw email-address <string>
|     +--rw fullname? <string>
|     +--rw login <string>
|     +--rw preferences
|     +---n user-password-aging
|     |  +--ro expiration-date <date-and-time(string)>
|     |  +--ro user <string>
|     +---n user-password-expired
|        +--ro expiration-date <date-and-time(string)>
|        +--ro user <string>
+--ro ietf-yang-library:yang-library
   +--ro content-id <string>
   +--ro datastore* [name]
   |  +--ro name <datastore-ref(identityref)>
   |  +--ro schema <leafref>
   +--ro module-set* [name]
   |  +--ro import-only-module* [name revision]
   |  |  +--ro location* <uri(string)>
   |  |  +--ro name <yang-identifier(string)>
   |  |  +--ro namespace <uri(string)>
   |  |  +--ro revision <union>
   |  |  +--ro submodule* [name]
   |  |     +--ro location* <uri(string)>
   |  |     +--ro name <yang-identifier(string)>
   |  |     +--ro revision? <revision-identifier(string)>
   |  +--ro module* [name]
   |  |  +--ro deviation* <leafref>
   |  |  +--ro feature* <yang-identifier(string)>
   |  |  +--ro location* <uri(string)>
   |  |  +--ro name <yang-identifier(string)>
   |  |  +--ro namespace <uri(string)>
   |  |  +--ro revision? <revision-identifier(string)>
   |  |  +--ro submodule* [name]
   |  |     +--ro location* <uri(string)>
   |  |     +--ro name <yang-identifier(string)>
   |  |     +--ro revision? <revision-identifier(string)>
   |  +--ro name <string>
   +--ro schema* [name]
      +--ro module-set* <leafref>
      +--ro name <string>
```

5.4 NBI Top-level Nodes

This section presents each of the top-level nodes in the NBI, providing both detailed information on each in turn. Each node's name uses the naming convention ">module-name<gt;top-level-node-name<".

This section provides a high-level overview of the API using YANG tree diagrams.

The following subsections are presented in sorted order.

5.4.1 /ietf-keystore:keystore

The top-level 'ietf-keystore:keystore' node, which is read-write (see "rw" in the tree diagram) contains a container called "asymmetric-keys" that contains a list of "asymmetric-key" nodes, each of which contains the key (only cleartext supported currently) and an optional number of signed certificates.

The "ietf-keystore" module is formally defined by in the IETF in RFC 9642.

The asymmetric key objects and their certificates are referenced by the "endpoint" structure described in /ietf-restconf-server:restconf-server. Specifically, when configuring YANGcore to listen on an 'https' endpoint, the TLS-level 'server-identity' node references the asymmetric key objects and their certificates. See also the Installation Guide here.

```
+--rw keystore {central-keystore-supported}?
   +--rw asymmetric-keys {asymmetric-keys}?
      +--rw asymmetric-key* [name]
         +--rw name                  string
         +--rw public-key-format?    identityref
         +--rw public-key?           binary
         +--rw private-key-format?   identityref
         +--rw (private-key-type)
         +--rw certificates
            +--rw certificate* [name]
               +--rw name         string
               +--rw cert-data    end-entity-cert-cms
```

5.4.2 /ietf-restconf-server:restconf-server

The top-level 'ietf-restconf-server:restconf-server' node, which is read-write (see "rw" in the tree diagram) contains a "listen/endpoints" container having a list of "endpoint" nodes, each defining a listening ports YANGcore opens.

The "ietf-restconf-server:restconf-server" module will be formally defined by the IETF in draft-ietf-netconf-restconf-client-server, an Internet Draft that is about to be published. Please note the data model presented here is subject to change, pending the final publication of those drafts as RFCs.

Each 'endpoint' must specify a 'yangcore:use-for' value, which (for YANGcore alone) must be the value 'yangcore:native-interface', but other interface identities may be defined by higher-level applications. YANGcore API ensures that there is always at least one endpoint that presents the 'yangcore:native-interface'.

```
+--rw restconf-server {central-restconf-server-supported}?
   +--rw listen! {listen}?
      +--rw endpoints
         +--rw endpoint* [name]
            +--rw name                  string
            +--rw (transport)
            |  +--:(http-over-tcp) {tcp-listen}?
            |  |  +--rw http-over-tcp
            |  |     +--rw external-endpoint!
            |  |     |  +--rw address                inet:host
            |  |     |  +--rw port?                  inet:port-number
            |  |     |  +--rw trusted-proxy-count?   uint8
            |  |     |  +--rw client-cert-var?       string
            |  |     +--rw tcp-server-parameters
            |  |     |  +--rw local-bind* [local-address]
            |  |     |     +--rw local-address    inet:ip-address
            |  |     |     +--rw local-port?      inet:port-number
            |  |     +--rw http-server-parameters
            |  |     |  +--rw server-name?   string
            |  |     +--rw restconf-server-parameters
            |  |        +--rw client-identity-mappings
            |  |           +--rw cert-to-name* [id]
            |  |              +--rw id             uint32
            |  |              +--rw fingerprint?   x509c2n:tls-fingerprint
            |  |              +--rw map-type       identityref
            |  |              +--rw name           string
            |  +--:(http-over-tls) {tls-listen}?
            |     +--rw http-over-tls
            |        +--rw tcp-server-parameters
            |        |  +--rw local-bind* [local-address]
            |        |     +--rw local-address    inet:ip-address
            |        |     +--rw local-port?      inet:port-number
            |        +--rw tls-server-parameters
            |        |  +--rw server-identity
            |        |  |  +--rw (auth-type)
            |        |  |     +--:(certificate) {server-ident-x509-cert}?
```



```
|        |  |        +--rw certificate
|        |  |           +--rw (inline-or-keystore)
|        |  +--rw client-authentication! {client-auth-supported}?
|        |     +--rw ca-certs! {client-auth-x509-cert}?
|        |     |  +--rw (inline-or-truststore)
|        |     +--rw ee-certs! {client-auth-x509-cert}?
|        |        +--rw (inline-or-truststore)
|        +--rw http-server-parameters
|        |  +--rw server-name?   string
|        +--rw restconf-server-parameters
|           +--rw client-identity-mappings
|              +--rw cert-to-name* [id]
|                 +--rw id              uint32
|                 +--rw fingerprint?    x509c2n:tls-fingerprint
|                 +--rw map-type        identityref
|                 +--rw name            string
+--rw yangcore:use-for?   identityref
```

5.4.3 /ietf-truststore:truststore

The top-level 'ietf-truststore:truststore' node, which is read-write (see "rw" in the tree diagram) has a container called "certificate-bags" that contains a list of "certificate-bag" nodes.

The "ietf-truststore" module is formally defined by in the IETF in RFC 9641.

The certificate-bag objects are referenced by the "endpoint" structure described in /ietf-restconf-server:restconf-server. Specifically, when configuring YANGcore to listen on an 'https', the TLS-level 'client-authentication' node references the certificate bag objects. See also the Installation Guide here.

```
+--rw truststore {central-truststore-supported}?
   +--rw certificate-bags {certificates}?
      +--rw certificate-bag* [name]
         +--rw name           string
         +--rw description?   string
         +--rw certificate* [name]
            +--rw name         string
            +--rw cert-data    trust-anchor-cert-cms
```

5.4.4 /yangcore:audit-log

The top-level 'yangcore:audit-log' node, which is read-only (see "ro" in the tree diagram) contains a list of "audit-log-record" nodes, each of which captures the following information for each inbound HTTP request:

- the timestamp for when the HTTP request was received
- the IP address of the HTTP client that sent the request
- the HTTP proxies, if any, used by the HTTP client
- the host that the HTTP request was sent to
- the HTTP method (e.g., GET, PUT, POST, etc.)
- the path (relative URL) of the HTTP request
- the outcome of the request's authorization (success or failure)
- any additional information for unauthorized requests[5]

YANGcore creates an audit log record for each incoming HTTP request.

```
+--ro audit-log
   +--ro audit-log-record* []
      +--ro timestamp          yang:date-and-time
      +--ro source-ip          inet:ip-address
      +--ro source-proxies*    string
      +--ro host               string
      +--ro username?          string
      +--ro method             enumeration
      +--ro path               string
      +--ro input?             <anydata>
      +--ro outcome            enumeration
      +--ro comment?           string
```

[5]Note that clients only receive a '401 Unauthorized' response, with no reason provided for why. This field enables YANGcore-users to see the why and authorization failed.

5.4.5 /yangcore:dynamic-callouts

The top-level "yangcore:dynamic-callouts" node, which is read-write (see "rw" in the tree diagram, contains a list of "dynamic-callout" nodes, each of which specifies the YANG 'rpc' the dynamic callout supports (see "rpc-supported") as well as a pointer (called a "leafref" and YANG venacular) to the specific plugin function that is executed (see /yangcore:plugins). YANGcore uses a dynamic callout for a number of use cases, including relaying audit log records and relaying notification log records.

```
+--rw dynamic-callouts
   +--rw dynamic-callout* [name]
      +--rw name                    string
      +--rw rpc-supported           identityref
      +--rw (callout-type)
         +--:(call-function)
            +--rw call-function
            |  +--rw plugin      -> /plugins/plugin/name
            |  +--rw function    -> /plugins/plugin[name = current()/../plugin]/functions/function/name
            +--rw opaque?          <anydata>
```

5.4.6 /yangcore:plugins

The top-level "yangcore:plugins" node, which is read-write (see "rw" in the tree diagram, contains a list of "plugin" nodes, each defining the name of a plugin (a Python package, i.e., a ".py" file) and a list of accessible functions inside the package.

Note that the plugin (the ".py" file) must be in the "plugins" directory before configuring a "plugin" node referencing it. The "plugins" directory is located where the package was installed, which is deployment specific, but can be found using the following command:

```
python -c 'import importlib.resources as resources; print(resources.files("yangcore") / "plugins")'
```

There is no limit as to how many plugins can be installed, how many functions each plugin (Python package) contains, or what RPCs the functions implement. Deployments can have a single plugin containing all functions, a plugin per function, of any combination in between.

```
+--rw plugins
   +--rw plugin* [name]
      +--rw name          string
      +--rw functions
         +--rw function* [name]
            +--rw name    string
```

5.4.7 /yangcore:preferences

The top-level "yangcore:preferences" node, which is read-write (see "rw" in the tree diagram used to configure account-level preferences[6]. User-specific preferences are not currently supported by YANGcore.

The "accounting" container is a placeholder for future preferences for the Audit Log (e.g., how much details to store per audit log record).

The "authentication" container first enables the configuration of authenticators. Currently only an internal authenticator is supported (see the "internal-authenticator" node), but support for external authenicators is on the roadmap. For the internal authenticator, first is a specification for what kinds of credentials are allowed. Currently only passwords are supported (see the "passwords-allowed" node), but X.509 client-certificates are on the roadmap. Each kind of crendential has (will have) a "is-required" flag, that indicates that clients must present this kind of credential, which only make sense to set when more then one kind is allowed. The "passwords-allowed" node can optionally (see the "!" in the tree diagram) specify a password-aging policy.

The "authorization" container is a placeholder for preferences for a future RBAC (role-based access control) mechanism.

The "general" container enables the server's "hostname" and "email-address" to be configured, which is optional (see the "?" in the tree diagram), but will be required when configuring, e.g., the server to send emails, which is on the roadmap.

[6] Currently there is only the 'operator' account. When multitenancy is enabled, there will be tenant-level accounts as well.

The 'outbound—interactions' container configures which dynamic callout should be called when relaying audit- and notification- log records.

```
+—rw preferences
   +—rw general
   |  +—rw hostname?         inet:host
   |  +—rw email—address?    string
   +—rw authentication
   |  +—rw internal—authenticator!
   |     +—rw passwords—allowed!
   |        +—rw is—required?      boolean
   |        +—rw aging—timeout!
   |           +—rw amount?    uint16
   |           +—rw units?     enumeration
   +—rw authorization
   +—rw accounting
   +—rw outbound—interactions
      +—rw relay—audit—log—record—callout?          -> /dynamic—callouts/dynamic—callout/name
      +—rw relay—notification—log—record—callout?   -> /dynamic—callouts/dynamic—callout/name
```

5.4.8 /yangcore:users

The top-level "yangcore:users" node, which is read-write (see "rw" in the tree diagram contains a list of "user" nodes. The primary key for each user is the "login" node.

When /yangcore:preferences indicate that passwords are allowed for the internal authenticator, then users may configure passwords. Note that passwords may be sent to the server hashed or in the clear, in which case they will be hashed prior to being persisted in the database.

```
+—rw users
   +—rw user* [login]
      +—rw login                      string
      +—rw email—address              string
      +—rw fullname?                  string
      +—rw preferences
      +—rw authentication
      |  +—rw password—based
      |     +—rw password?                  ianach:crypt—hash
      |     +—ro password—last—modified?    yang:date—and—time
      +—rw authorization
      |  +—rw (auth—type)
      |     +—:(unrestricted)
      |        +—rw unrestricted?   empty
      +—n user—password—aging
      |  +— user                  string
      |  +— expiration—date       yang:date—and—time
      +—n user—password—expired
         +— user                  string
         +— expiration—date       yang:date—and—time
```


6 Outbound Interface Details

YANGcore uses an Outbound interface to interact with peering systems (e.g., authenticators, loggers, monitors, etc.).

Unlike the Northbound interface, the Outbound interface does not present an API, rather it defines an API it expects plugins implementing functions for Dynamic Callouts.

6.1 Notifications

These are the notifications that may be sent via the "relay-notification-log-record" dynamic callout.

Subsections describe the notifications sent by YANGcore.

6.1.1 The "user-password-aging" Notification

The tree-diagram for this notification is as follows:

```
+---n user-password-aging
   +---- user                string
   +---- expiration-date     yang:date-and-time
```

6.2 Dynamic Callouts

Dynamic callouts may be configured to reach out to external systems to either push information to them or fetch information from them.

Dynamic callouts are currently only implemented via plugins (see Plugins) loaded into YANGcore, enabling the plugin to implement a protocol of its choosing, but it is planned to also enable YANGcore to also execute via webhooks (i.e., an HTTP "POST" request).

In either case, dynamic callouts are modeled using YANG 'rpc' statements (e.g., see the yangcore-rpcs.yang file).

Subsections describe the dy The following "rpc" statements are defined in the "yangcore-rpcs" YANG module:

```
rpcs
  +-- rpc relay-notification-log-record
  +-- rpc relay-audit-log-record
```

Following lists all of the dynamic callouts implemented by YANGcore (see Dynamic Callouts).

6.2.1 The "relay-notification-log-record" Dynamic Callout

A dynamic callout implementing the "relay-notification-log-record" RPC is used to relay notifications supported by YANGcore and higher-level applications. The notifications supported by YANGcore are described in Notifications.

6.2.1.1 Tree Diagram

Following is the tree diagram for the "relay-notification-log-record" dynamic callout:

```
rpcs:
  +---x relay-notification-log-record
  |  +---w input
  |     +---w notification-log-record
  |     |  +---w timestamp        yang:date-and-time
  |     |  +---w notification     anydata
  |     +---w opaque?                     anydata
  +---x relay-audit-log-record
     +---w input
        +---w audit-log-record
        |  +---w timestamp          yang:date-and-time
        |  +---w source-ip          inet:ip-address
        |  +---w source-proxies*    string
        |  +---w host               string
        |  +---w username?          string
        |  +---w method             enumeration
        |  +---w path               string
        |  +---w input?             anydata
        |  +---w outcome            enumeration
        |  +---w comment?           string
        +---w opaque?              anydata
```

6.2.1.2 Example Usage

Here is a Python plugin that implements a function that can be used for the "relay-notification-log-record" RPC.

```
import json

async def my_relay_notification_log_record_function(input: dict, opaque: dict):

    # show input
    print(json.dumps({"input": input, "opaque": opaque}, indent=3))

    # rpc has no 'output'
```

Note that the first line in the function might output the followings:

```
{
   "input": {
      "notification-log-record": {
         "timestamp": "2025-01-22T00:09:45Z",
         "payload": {
            "yangcore:user-password-aging": {
               "user": "my-admin@example.com"
            }
         }
      }
   },
   "opaque": null
}
```


6.2.2 The "relay-audit-log-record" Dynamic Callout

A dynamic callout implementing the "relay-audit-log-record" RPC is used to relay audit log entries, as defined by the "yangcore" YANG module.

6.2.2.1 Tree Diagram

Following is the tree diagram for the "relay-notification-log-record" dynamic callout:

```
rpcs:
  +---x relay-notification-log-record
  |  +---w input
  |     +---w notification-log-record
  |     |  +---w timestamp        yang:date-and-time
  |     |  +---w notification     anydata
  |     +---w opaque?                     anydata
  +---x relay-audit-log-record
     +---w input
        +---w audit-log-record
        |  +---w timestamp          yang:date-and-time
        |  +---w source-ip          inet:ip-address
        |  +---w source-proxies*    string
        |  +---w host               string
        |  +---w username?          string
        |  +---w method             enumeration
        |  +---w path               string
        |  +---w input?             anydata
        |  +---w outcome            enumeration
        |  +---w comment?           string
        +---w opaque?              anydata
```

6.2.2.2 Example Usage

Here is a Python plugin that implements a function that can be used for the "relay-audit-log-record" RPC.

```
import json

async def my_relay_audit_log_record_function(input: dict, opaque: dict):

    # show input
    print("json.dumps({"input": input, "opaque": opaque}, indent=3))

    # rpc has no 'output'
```

Note that the first line in the function might output the followings:

```
{
   "input": {
      "audit-log-record": {
         "timestamp": "2025-01-22T00:09:46Z",
         "source-ip": "127.0.0.1",
         "host": "localhost:8080",
         "method": "DELETE",
         "path": "/restconf/ds/ietf-datastores:running/yangcore:users/user=my-admin",
         "outcome": "success"
      }
   },
   "opaque": null
}
```

