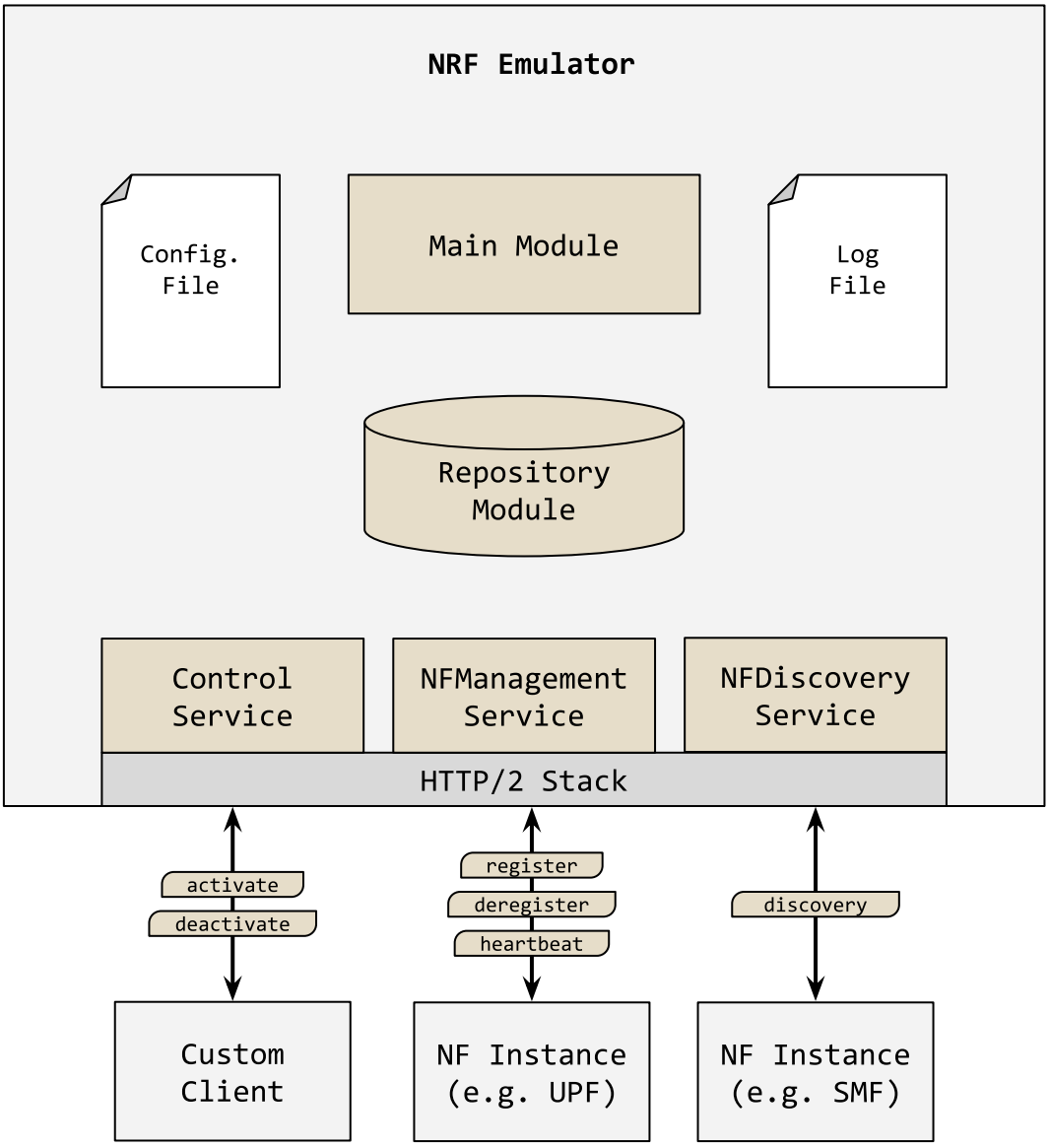
# NRF Emulator - User Guide

## Table of Contents

# Introduction

This document describes the Emblasoft NRF Emulator designed for performance and functional tests, providing a subset of the NRF interface services as specified in TS 29.510[1].

The following picture shows an overview of the implementation and provided interfaces:




The NRF Emulator is implemented in Java and provides 5G Network Function interfaces on top of an HTTP/2[2] stack.

# Installation

## Prerequisites

The NRF Emulator requires Java 11. The latest Java 11 version is recommended.

## Distribution

The NRF Emulator is delivered as a tar file to be extracted in an appropriate directory.
The tar file contains:

- README.txt general information
- bin/ script to start emulator
- config/ configuration file(s)
- doc/ documentation
- lib/ library files
- log/ log output directory
- resources/ 5GC specification file(s)

## Start Script Modification

The script to start the NRF Emulator, *bin/nrf-emulator.sh*, must be edited to specify the JAVA_HOME variable.

Change the following line:
`JAVA_HOME=/opt/java/jdk-11`
to where Java 11 is installed.

# Configuration

The server is configured in the *config/nrf-emulator.properties* file:

```
#
# HTTP server settings
#

# Log level [ DEBUG | INFO | ERROR ]
http-server.log.level=INFO

# IP address for http listener (set to blank to disable http).
http-server.http.ip=127.0.0.1
# Listening port for http.
http-server.http.port=38080
# IP address for https listener (set to blank to disable https).
http-server.https.ip=127.0.0.1
# Listening port for https.
http-server.https.port=38443

# Comma separated list of SSL protocols to enable (must enable at least one).
# Supported SSL protocols: tlsv1, tlsv1.1, tlsv1.2, tlsv1.3
http-server.ssl.protocols=tlsv1.3
# Comma separated list of cipher suites to enable (leave blank for default).
http-server.ssl.ciphers=
# Server certificate and private key file in PKCS#12 format (leave blank for self-signed).
http-server.cert.file=
# Server certificate passphrase.
http-server.cert.passphrase=

#
# NRF settings
#

# NRF nfInstanceId (leave blank for automatically generated random UUID).
nrf.id=

# Heart-beat timer value in seconds (used if registering consumer does not propose a value).
nrf.heart-beat.timer=90

# Search result validity period in seconds.
nrf.search.validity=100

# Flag to control if incoming requests should be validated.
nrf.validate.requests=true
# Flag to control if outgoing responses should be validated.
nrf.validate.responses=true
# Set to true to "pretty-print" JSON responses.
nrf.pretty.json=true
```

# Operation

## Starting the Emulator

The NRF Emulator is started using the nrf-emulator.sh start script in the bin directory.

```
bin$ ./nrf-emulator.sh
09:01:16.721 INFO  NRF Emulator started
09:01:16.805 INFO  http listener: 192.168.1.2:38080
09:01:17.289 INFO  https listener: 192.168.1.2:38443
```

The NRF Emulator starts in an active state with an empty repository.

## HTTP Listeners

The NRF Emulator uses two listeners for incoming requests:

**http listener** accepts unencrypted HTTP/1.1 requests
**https listener** accepts TLS encrypted HTTP/2 and HTTP/1.1 requests

## Emulator State

The NRF Emulator executes as a separate process and has two main states: active and inactive. When in active state, the emulator responds to NRF service operations and maintains an in-memory repository. When in inactive state, all incoming service requests are denied with "503 Service Unavailable" responses.

## Validation

Incoming requests and outgoing responses from the NRF Emulator can be configured to be validated against the corresponding OpenAPI service specification. For performance tests, validation can be disabled by setting the **nrf.validate.requests** and **nrf.validate.respones** configuration parameters to false.

## JSON Format

JSON response messages can be configured to use "pretty printing" for human readability. This is configured by setting the **nrf.pretty.json** configuration parameter.

# NRF Services

The NRF Emulator implements partial support of the Nnrf_NFManagement and Nnrf_NFDiscovery services. There is also a custom Control service to set and query the state of the emulator.

## Nnrf_NFManagement Service

The Nnrf_NFManagement service allows a Network Function Instance in the serving PLMN to register, update or deregister its profile in the NRF. See section 5.2 of TS 29.510[1].

### NFRegister

The NFRegister operation registers a new NF Instance in the NRF. See section 5.2.2.2 of TS 29.510[1].

| | | |
|---|---|---|
| **PUT** | {apiRoot}/nnrf-nfm/v1/nf-instances/{nfInstanceID} | Register a new NF Instance |

The NRF Emulator will accept any heartBeatTimer value proposed by the consumer. If none is provided by the consumer, the nrf.heart-beat.timer value in the configuration will be used.

### NFUpdate

The NFUpdate operation updates the profile of a previously registered NF Instance. See section 5.2.2.3 of TS 29.510[1].

| | | |
|---|---|---|
| **PATCH** | {apiRoot}/nnrf-nfm/v1/nf-instances/{nfInstanceID} | Update NF Instance profile |

The NFUpdate operation must be invoked periodically by registered NF Instances to show that they are still operative. See section 5.2.2.3.2 of TS 29.510[1].

### NFDeregister

The NFDeregister operation deregisters a given NF Instance from the NRF. See section 5.2.2.4 of TS 29.510[1].

| | | |
|---|---|---|
| **DELETE** | {apiRoot}/nnrf-nfm/v1/nf-instances/{nfInstanceID} | Deregister a given NF Instance |

### NFListRetrieval

This service operation allows the retrieval of a list of NF Instances that are currently registered in NRF. See section 5.2.2.8 of TS 29.510[1].

| | | |
|---|---|---|
| **GET** | {apiRoot}/nnrf-nfm/v1/nf-instances | Retrieves a collection of NF Instances |

### NFProfileRetrieval

This service operation allows the retrieval of the NF profile of a given NF instance currently registered in NRF. See section 5.2.2.9 of TS 29.510[1].

| | | |
|---|---|---|
| **GET** | {apiRoot}/nnrf-nfm/v1/nf-instances/{nfInstanceID} | Read the profile of a given NF Instance |

## Nnrf_NFDiscovery Service

The Nnrf_NFDiscovery service allows a Network Function Instance to discover services offered by other Network Function Instances, by querying the local NRF. See section 5.3 of TS 29.510[1].

### NFDiscover

This service operation discovers the set of NF Instances (and their associated NF Service Instances), represented by their NF Profile, that are currently registered in NRF. See section 5.3.2.2 of TS 29.510[1].

| | | |
|---|---|---|
| **GET** | {apiRoot}/nnrf-disc/v1/nf-instances?<query parameters> | Search a collection of NF Instances |

The NRF Emulator supports discovery within the same PLMN.

The query parameter **target-nf-type** can be used to filter the type of target NF. All other filtering query parameters are ignored by the NRF Emulator.

## Control Service

The custom Control Service is used to set and query the state of the NRF Emulator.

### Set State

This operation sets the state of the NRF Emulator and returns "204 No Content" on success.

| | | |
|---|---|---|
| **PUT** | {apiRoot}/nrf-emu/control | Set state of NRF Emulator |

The payload body of the PUT request shall contain a representation of the NRF Emulator state:

```
{
  "state": "active"
}
```

The "state" can be one of "active" or "inactive". When set to inactive state, the NRF Emulator will clear the NF Instance repository and respond to all incoming service requests with "503 Service Unavailable" until set to active state again.

### Get State

This operation gets the current state of the NRF Emulator.

| | | |
|---|---|---|
| **GET** | {apiRoot}/nrf-emu/control | Get state of NRF Emulator |

A representation of the current NRF Emulator state is returned in a "200 OK" response payload body.

# References

[1] 3GPP TS 29.510: "Network Function Repository Services; Stage 3".
[2] IETF RFC 7540: "Hypertext Transfer Protocol Version 2 (HTTP/2)".