THENOWFACTORY℠
INSIGHTS MADE INSTANT

CNA 9.1

Mediation Operations Guide

Contents

1 OVERVIEW 6

1.1 Intended Audience 6

1.2 Document History 6

1.3 Document Approval 7

1.4 Related Documents 7

1.5 Glossary 7

2 EXECUTIVE SUMMARY 9

3 INTRODUCTION 10

3.1 Data flow 11

3.2 Collector 12

3.3 ITE 12

3.4 RawTDR 12

3.5 Centralised DB Loader 12

4 INSTALLATION GUIDE 14

4.1 Prerequisites 14

4.1.1 Key sharing with master server 14

4.1.2 Redis 15

4.1.3 FTP server 16

4.2 RPM installation 16

4.2.1 Building Prerequisites 17

4.2.2 Building Collector 17

4.2.3 Building the centralised DB Loader 18

4.2.4 Building the local DB Loader 18

4.2.5 Building Lookup Manager 19

4.2.6 Building ITE 20

4.2.7 Building RawTDR 21

4.2.8 Listing and removing packages 21

4.3 Streams domain and instance configuration 22

4.3.1 Zookeeper setup 22

4.3.1.1 Embedded Zookeeper 22

4.3.1.2 External Zookeeper 23

4.3.2 Streams main domain 23

4.3.3 Streams instances 24

4.3.3.1 Streams Collector instance 24
4.3.3.2 Streams Centralized DB Loader instance 25
4.3.3.3 Streams ITE/RawTDR instance 25
4.3.3.4 Operating Streams instances 25
4.3.4 Sizing considerations 26

4.4 Customization 26
4.4.1 Compile-time parameters 27
4.4.1.1 DB Loader 27
4.4.1.2 Lookup Manager 27
4.4.1.3 ITE 28
4.4.1.4 RawTDR 30
4.4.2 Collector 31
4.4.2.1 Streams instance 31
4.4.2.2 Run local Collector automatically from cna.sh 32
4.4.2.3 Collectors section 32
4.4.2.4 Global section 33
4.4.2.5 Post download commands 37
4.4.2.6 Symlinks explanation 37
4.4.2.7 Voice/SMS Interface Adaptor installation 38
4.4.3 Local and centralised DB Loader 39
4.4.3.1 How to switch 39
4.4.3.2 Configuring sudoers 39
4.4.3.3 Streams instance 39
4.4.3.4 Local DB Loader: loader_*.ini 40
4.4.3.5 Centralized DB Loader: cLoader_*.ini 41
4.4.4 ITE 42
4.4.4.1 probes.csv 42
4.4.4.2 consts_gn.ini, consts_lte.ini 43
4.4.4.3 consts_voice.ini 44
4.4.4.4 consts_ran.ini 45
4.4.4.5 consts_fixedline.ini 45
4.4.4.6 consts_gb.ini 46
4.4.4.7 consts_iups.ini 47
4.4.4.8 consts_s1mme_s6a.ini 47
4.4.4.9 consts_gx_gy.ini 48
4.4.4.10 consts_s13_sgs.ini 49
4.4.5 RawTDR 49
4.4.5.1 probes-tdr.csv 49
4.4.5.2 rawtdr_*.ini 50
4.4.6 Provisioning Files 50
4.4.6.1 cells.csv 50
4.4.6.2 mcc_mnc.csv 50
4.4.6.3 collector.csv 51
4.4.6.4 bnumber.csv 51
4.4.6.5 status.csv 51
4.4.6.6 gn_qoscategory.csv 51
4.4.6.7 lte_qoscategory.csv 52
4.4.6.8 fixedline_qos.csv 52
4.4.6.9 sgsn.csv 52
4.4.6.10 ggsn.csv 52
4.4.6.11 sgw.csv 53
4.4.6.12 bsc.csv 53

4.4.6.13 rnc.csv 53
4.4.6.14 mme.csv 53
4.4.6.15 bts.csv 53
4.4.6.16 nodeb.csv 54
4.4.6.17 enodeb.csv 54
4.4.6.18 applications.csv 54
4.4.6.19 sip_status_codes.csv 54
4.4.6.20 csid.csv 55
4.4.6.21 tier0.csv 55
4.4.6.22 tier1.csv 55
4.4.6.23 fixedline_transaction_type.csv 55
4.4.6.24 fixedline_transaction_status.csv 55
4.4.6.25 gb_status.csv 56
4.4.6.26 iups_status.csv 56
4.4.6.27 sx_status.csv 56
4.4.6.28 sx_transactiontype.csv 56
4.4.6.29 dra.csv 57
4.4.6.30 hss.csv 57
4.4.6.31 s6a_dra_position.csv 57
4.4.6.32 pcrf.csv 57
4.4.6.33 ocs.csv 57
4.4.6.34 gxgy_transactiontype.csv 58
4.4.6.35 gxgy_transaction_status.csv 58
4.4.6.36 vlr.csv 58
4.4.6.37 s13_sgs_transactiontype.csv 58
4.4.6.38 s13_sgs_transaction_status.csv 59
4.4.6.39 dns.csv 59
4.4.7 Provisioning Update Configuration 59
4.4.7.1 Provisioning configuration in cna.sh 59
4.4.7.2 Provisioning.cfg 60

4.5 Installing the Analytics Platform Components 60

5 ADMINISTRATION GUIDE 62

5.1 Starting the applications 62

5.2 Stopping the applications 62

5.3 Checking the application status 62

5.4 Cleaning temporary files 63

5.5 Provisioning updates 63
5.5.1 Provisioning auto-updates 63
5.5.2 Manual provisioning auto update 63

5.6 Consulting log files 63
5.6.1 Collector 63
5.6.2 Collection 64
5.6.3 Lookup 64
5.6.4 Aggregation 65
5.6.5 Database 65

5.6.6 Performance 66
5.6.7 Streams logs 66

5.7 Recovery mechanisms 66

5.8 Housekeeping 67

1 Overview

1.1 Intended Audience

Operations team working on customer site.

1.2 Document History

Date	Author	Version	Notes
16/02/2016	Vit Letal	0.1.0	Branched from CNA 9.0
18/02/2016	Vit Letal	0.2.0	Provisioned keys must be unique otherwise Lookup Manager fails in Streams 4.1.1
19/02/2016	Oliver Brandt	0.2.1	Added description for KeepSources
19/02/2016	Vit Letal	0.2.2	Streams domain setup clarification: Embedded vs External Zookeeper
23/02/2016	Vit Letal	0.2.3	Increased CellCacheSizeMB to 50 MB
25/02/2016	Ivan Vecanski	0.3	Added Gb Interface
26/02/2016	Vit Letal	0.3.1	Added Vedia Supercollector support for S1MME/S6a, Gb, IuPS and S8
29/02/2016	Sergey Scobaro	0.4	Added S1MME/S6a Interface
01/03/2016	Ivan Vecanski	0.4.1	Added Gb hadoop loading options
04/03/2016	Ivan Vecanski	0.4.2	Added S1MME/S6a hadoop loading options
10/03/2016	Ferdinando Formica	0.5	Added IuPS Interface
14/03/2016	Ivan Vecanski	0.6	Added collector.csv provisioning file to support roaming calculations for S8
18/03/2016	Ivan Vecanski	0.6.1	Added IuPS hadoop loading options
29/03/2016	Sergey Scobaro	0.7	Added Gx/Gy Interface
06/04/2016	Ivan Vecanski	0.7.1	Added Gx/Gy hadoop loading options
12/04/2016	Ferdinando Formica	0.8	Added Centralised DB Loader
12/04/2016	Vit Letal	0.9	Added Streams Supercollector
14/04/2016	Ferdinando Formica	0.9.1	Collected Centralised and Local DB Loader into their own section
19/04/2016	Ahmad Nouri	0.9.2	Rename Supercollector -> Collector Adapted the new script names and their locations
19/04/2016	Ferdinando Formica	0.10	Added Redis and dependencies
21/04/2016	Vit Letal	0.11	Organized the Customization section differently, revised the Collector and Centralized DB Loader
25/04/2016	Ferdinando Formica	0.11.1	Added some remarks on the supported FTP configuration, changed loader_script path
27/04/2016	Ferdinando Formica	0.11.2	Added new Centralised Loader build dependency (ant), additional remarks on launching Redis from a script
28/04/2016	Sergey Scobaro	0.12	Added S13/SGs Interface
02/05/2016	Vit Letal	0.13	Detailed Streams instances better (per Collector, per CLoader, dataflow, admin). Added Collector clean command
03/05/2016	Vit Letal	0.14	Fixed NumParallelParsers
04/05/2016	Vit Letal	0.15	Added disabling the local Collector from cna.sh

Date	Author	Version	Notes
11/05/2016	Ferdinando Formica	0.15.1	Minor changes to configuration (FTP vs SFTP, number of subscriber segments)
18/05/2016	Vit Letal	0.16	Added Collector global parameter remove_if_command_fails Also fixed the name of post-download scripts: adaptor->adapter
23/05/2016	Ferdinando Formica	0.17	Added Collector build prerequisites
24/05/2016	Vit Letal	0.18	Using symlinks in Collector and Centralized DB loader for 3rd party access Added the Housekeeping section
25/05/2016	Vit Letal	0.19	Added recommendation to increase numberOfParallelRunningRemoteDirectoryScanners in the centralized DB loader
31/05/2016	Sergey Scobaro	0.20	Added DNS KPI's for Gn and LTE

1.3 Document Approval

Role	Name	Approval Date
Document Owner		

1.4 Related Documents

Document	Description
Streams 4.1.1 Documentation	Available on http://www-01.ibm.com/support/knowledgecenter/SSCRJU_4.1.1/com.ibm.streams.welcome.doc/doc/kc-homepage.html?lang=en

1.5 Glossary

Acronyms and abbreviations used in this document are described in the table below.

Abbreviation	Description
CI	Cell Identifier
DB	Database
ECI	E-UTRAN Cell Identifier
GGSN	Gateway GPRS Support Node
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
LAC	Location Area Code
LTE	Long Term Evolution
MCC	Mobile Country Code
MNC	Mobile Network Code
MSISDN	Mobile Station International Subscriber Directory Number
RPM	RedHat Package Manager
SGSN	Serving GPRS Support Node
Streams	IBM InfoSphere Streams 4.1.1

TAC	Track Area Code
TDR	Telecommunications Data Record

2 Executive Summary

This document provides information regarding installation, deployment, configuration and maintenance of the mediation layer of the Customer Network Analytics (CNA) 9.1 system.

The business logic of CNA 9.1 is similar to Vantage 8.3; the main difference is that mediation is Streams-based instead of Vedia-based.

The mediation solutions include the following Streams-based components:

- Collector
- Lookup Manager
- ITE Application to produce 5 minute aggregations
- Raw TDR processing Application
- DB Loader Application (centralized or local)

Note that this document will not cover the installation of the Streams platform, which is a prerequisite for the application deployment.

Note also that the Hadoop installation and Hadoop-based aggregation is out of scope of this document.

3 Introduction

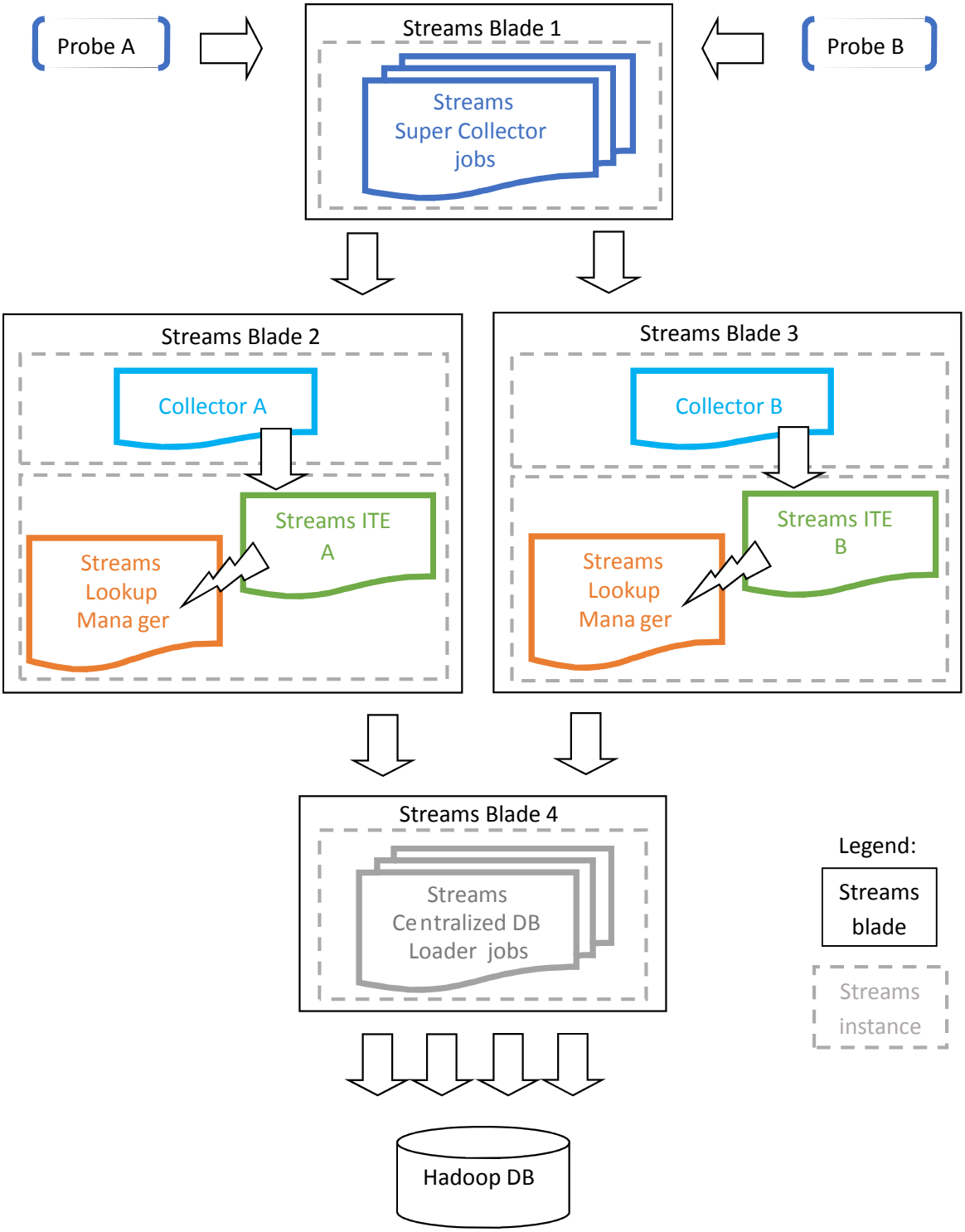
This document describes the steps needed to install and deploy mediation components for the following CNA 9.1 interfaces:

- Gn
- LTE S1U / S11
- Voice & SMS
- Voice over LTE (VoLTE)
- Fixed Line
- Radio Access Network (RAN)
- Gb
- S1MMe / S6a
- Gx / Gy
- S13 / SGs

It also describes how to install and configure all mediation components described in the previous Executive section.

3.1 Data flow

The following diagram outlines the typical Streams mediation data flow using the Streams components listed above.



3.2 Collector

The Streams ITE and RawTDR components cannot collect files from a remote host (ftp/sftp). The remote collection is done using the Streams Collector component.

The Collector is typically deployed and configured to operate in two different modes:

1. The “super collector” mode (optional):
   In this mode the Collector is deployed as a separate and stand-alone Streams host. It connects to a list of probes (Sourceworks or 3rd party) and downloads TDR files to the local host’s storage (the storage requirements need to be consulted with the sizing document as the storage needs to be capable to contain few hours of compressed feed from multiple probes). In this mode the Collector can also call the external interface adaptor and create as many symlinks to downloaded (and adapted) TDR files as necessary (typically for Streams ITE and RawTDR, but possibly for data access points, too).

2. The “local collector” mode (mandatory):
   In this mode the Collector is deployed in every Streams host where either Streams ITE or RawTDR is deployed. It collects TDR files (adapted or non-adapted) from either the probe directly (bypassing the “collector” mode above) or from the above described “super collector” host.
   If downloading from the “super collector” host then no feed adaption is needed, because it was already done in that host.
   If downloading from the probe then it is also possible to call the interface adapter.
   Symlinks to one downloaded feed file are also created; typically just one. In case that there is both Streams ITE and RawTDR deployed in the same host then only one Collector job is to be used that generates two symlinks.

The “local collector” needs to be deployed always in order to download the feed files remotely. The “super collector” is not necessary if only ITE or only RawTDR component is deployed and there is no need to have a centralized TDR storage for all probes’ feed. However, if both ITE and RawTDR components are deployed then the “collector” is mandatory in order to minimize the network overhaul (that would be caused by downloading the same feed file twice).

Both Collector modes also support exposing the downloaded files for 3rd party access points. This is done by creating a configurable list of symlinks (see section 4.4.2.6).

3.3 ITE

The ITE application is *tnf.rapidanalytics.ite* which has an output of aggregate files (agg).

3.4 RawTDR

The RawTDR application is *tnf.rapidanalytics.rawtdr* which has an output of raw TDR files (raw).

3.5 Centralised DB Loader

The Streams ITE and RawTDR components output files into the local filesystem, and an additional component is needed to load these files into the Hadoop database.

Currently two different DB loader components are supported: a local and a centralised DB loader. The local DB loader is the default, but the centralised one is recommended for production environments.

The centralised DB loader is a Streams application running on a per-interface basis. Data flow is:

- Scans all configured remote hosts looking for Streams output files (created by ITE or RawTDR jobs)
- Downloads files using FTP or SFTP
- Bundles downloaded files into bigger files according to pre-configured rules
- Invokes the configured loading script for every merged file

The bundling reduces the loading overhead (by invoking less DB connections) and improves performance when collecting from several blades running the same ITE interface: by default, it is not performed on RawTDR files that are already big enough (there would be little performance improvement, if any).

The state of the centralised DB loader is stored in a separate key-value server in order to provide high availability. Currently the only supported server is Redis. It must be downloaded, installed and run independently before the centralised DB loader is started.

The centralized DB loader also supports exposing the bundled files for 3rd party access points. This is done by creating a configurable list of symlinks (see the parameter symbolicLinkDirectories in the section 4.4.3.5). This feature is not supported by the local DB loader.

4 Installation Guide

This section describes how to install the mediation layer of CNA 9.1.

The system is composed of the following Streams applications:

- ITE – Aggregates TDRs over 5 minutes. Several builds of the same application are provided, one for every supported interface.
- RawTDR – Prepares raw TDRs for loading into the DB. Several builds of the same application are provided, one for every supported interface.
- Lookup Manager – The controlling process for all ITE jobs, which provides access to shared provisioning data (Home and Visited MCC/MNC, Cells)
- DB Loader – Loads aggregated records and raw TDRs to the Hadoop DB. The DB Loader can be either centralized (recommended in order to reduce the number of parallel Hadoop loaders) or local (running in every Streams host where ITE or RawTDR is deployed).
- Collector – Collects TDR files from the probes and makes them available in the local filesystem, creating symlinks where necessary.

It is possible to install one or more interfaces in the same blade if needed.

4.1 Prerequisites

This installation assumes that:

- You have done a sizing assessment (per interface) in terms of traffic throughput (Gbit/s), number of cells and locations and number of unique daily subscribers for each of the Sourceworks probes
- IBM InfoSphere Streams 4.1.1 is installed
- Redis key-value server is installed (only if the Streams Centralized DB Loader is to be deployed), see section 4.1.2
- There is a boss user belonging to a boss group, and this user can run Streams commands
- The ulimit settings for the user are compatible with Streams (see http://www-01.ibm.com/support/knowledgecenter/SSCRJU_4.1.1/com.ibm.streams.install.doc/doc/ibminfospherestreams-install-operating-system-settings.html?lang=en)
- BigInsights and its dependencies (Hadoop and the CNA Data Loader) are correctly installed as per the Installation-And-Configuration-Guide.pdf located in the CNA framework release.

4.1.1 Key sharing with master server

In order to allow the provisioning autoupdate feature within the cna.sh script to check the provisioning files on the remote server via ssh/scp, it is required to share the Mediation box public key with the central server. This has to be done for each Mediation blade. The following

demonstrates the commands that need to be invoked on Mediation blades to generate and share their public keys. The master server in this example is 9.162.178.179, replace it with the actual server IP address or hostname.

```
# Generate a pair of authentication keys. Do not enter a passphrase
[streamsadmin@streamsqse ~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/streamsadmin/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/streamsadmin/.ssh/id_rsa.
Your public key has been saved in /home/streamsadmin/.ssh/id_rsa.pub.
# Create a directory ~/.ssh for tnf user if it doesn't exist already
[streamsadmin@streamsqse ~]$ ssh tnf@9.162.178.179 mkdir -p .ssh
tnf@9.162.178.179's password:
# Append the new public key to tnf's authorized_keys
[streamsadmin@streamsqse ~]$ cat .ssh/id_rsa.pub | ssh tnf@9.162.178.179 'cat >>
.ssh/authorized_keys'
tnf@9.162.178.179's password:
# You should now be able to login without the password:
[streamsadmin@streamsqse ~]$ ssh tnf@9.162.178.179
Last login: Tue Nov 10 09:21:38 2015 from icon-9-164-161-9.megacenter.de.ibm.com
[tnf@tnfvmbig3 ~]$
```

4.1.2 Redis

Redis is an open source key-value server available under the BSD licence and it is required only for the Centralised DB Loader. Redis supports clustering from version 3, and this mode is especially well suited to support high availability: however, at the moment it's not possible to customise the Centralised DB Loader to use it. Instead a local standalone server should be used.

In order to install Redis:

- Download the source package from http://download.redis.io/releases/
- Extract the tarball
- Enter the extracted directory
- Execute "make" (see the detailed installation instructions in the Redis download webpage above)

After this Redis executables are generated in the src directory. The server can be started by executing `redis-server`: at its simplest, it can be run without any option, and in that case it will use the default configuration. In the production environment it might be necessary to specify a configuration file with some additional options, possibly regarding security and encryption: please check the full documentation at http://redis.io/documentation.

When the Centralised DB Loader starts it expects to connect to the Redis-server on the local machine at the default port 6379, so Redis should be started before. In order to ensure state persistence, Redis shouldn't be restarted, and in production it should be considered adding it to the default runlevel.

When starting Redis from a script, in order to decouple the script termination from Redis, the following syntax should be used:

- To start: nohup ./redis-server >/dev/null 2>&1 &
- To stop: ./redis-cli shutdown

4.1.3 FTP server

The Centralised DBLoader app works preferably over FTP; then all mediation blades then need to run an FTP server. The tested solution is vfstpd (https://security.appspot.com/vsftpd.html). The following considerations apply in this case:

- In the file /etc/vsftpd/vsftpd.conf, make sure that the option chroot_local_user is not set or set to “NO”;
- Make sure that selinux is not enforcing or that the option ftp_home_dir is on; the following command can be run to enable it:
  `setsebool -P ftp_home_dir on`

If a different FTP server is selected, check for the equivalent settings to allow the user *boss* to change directory to the selected loading path.

4.2 RPM installation

The installation of mediation components is performed through RedHat's RPM system.

All components of CNA 9.1 are contained within a TAR archive **CN7Y0EN.tar**, at the path CNA/dist/CNA/rpm/streams: the contents of the directory are the following.

```
cloader-9.1.0.0.0.compile.rpm
collector-9.1.0.0.0.compile.rpm
interface-adapter-8.2.0.1.2.rpm
ite_fixedline-9.1.0.0.0.compile.ini
ite_fixedline-9.1.0.0.0.compile.rpm
ite_gn-9.1.0.0.0.compile.ini
ite_gn-9.1.0.0.0.compile.rpm
ite_lte-9.1.0.0.0.compile.ini
ite_lte-9.1.0.0.0.compile.rpm
ite_ran-9.1.0.0.0.compile.ini
ite_ran-9.1.0.0.0.compile.rpm
ite_voice-9.1.0.0.0.compile.ini
ite_voice-9.1.0.0.0.compile.rpm
ite_volte-9.1.0.0.0.compile.ini
ite_volte-9.1.0.0.0.compile.rpm
ite_gb-9.1.0.0.0.compile.ini
ite_gb-9.1.0.0.0.compile.rpm
ite_s1mme_s6a-9.1.0.0.0.compile.ini
ite_s1mme_s6a-9.1.0.0.0.compile.rpm
ite_iups-9.1.0.0.0.compile.ini
ite_iups_s6a-9.1.0.0.0.compile.rpm
ite_gx_gy-9.1.0.0.0.compile.ini
ite_gx_gy-9.1.0.0.0.compile.rpm
loader-9.1.0.0.0.compile.ini
loader-9.1.0.0.0.compile.rpm
lookup_manager-9.1.0.0.0.compile.ini
lookup_manager-9.1.0.0.0.compile.rpm
rawtdr_gn-9.1.0.0.0.compile.ini
rawtdr_gn-9.1.0.0.0.compile.rpm
```

```
rawtdr_lte-9.1.0.0.0.compile.ini
rawtdr_lte-9.1.0.0.0.compile.rpm
rawtdr_voice-9.1.0.0.0.compile.ini
rawtdr_voice-9.1.0.0.0.compile.rpm
rawtdr_fixedline-9.1.0.0.0.compile.ini
rawtdr_fixedline-9.1.0.0.0.compile.rpm
```

4.2.1 Building Prerequisites

The Streams-based RPMs are self-compiling: installing the provided RPM generates a binary RPM which can then be deployed on all the blades. Because of this the blade where the compilation is performed must meet the following additional prerequisites:

- The packages *rpm-build* and *zlib-devel* are installed
- In the case of the Collector, *libcurl-devel* and *openssl-devel* must also be installed
- In the case of the Centralised DB Loader, *libcurl-devel*, *openssl-devel*, *openldap-devel* and ant must also be installed
- InfoSphere Streams 4.1.1.0 is sourced in the shell from which the commands are issued

4.2.2 Building Collector

The Collector RPM is:

```
CNA/dist/CNA/rpm/streams/collector-9.1.0.0.0.compile.rpm
```

The package can then be built by issuing the following command as root in the same directory as the RPM:

```
rpm -ivh collector-9.1.0.0.0.compile.rpm --prefix $PWD
```

The binary RPM will be created as collector-9.1.0.0.0.rpm; if the operation fails, the file will still be present, but unusable, and the package will have to be removed (see later).

The compilation log, named collector_*datetime*.log (where *datetime* is 14 digits representing the date and time when the compilation was started) is produced in the same directory as the output RPM, and it contains the selected customisation options. Note that it is not automatically removed, so it should be cleaned manually when no longer relevant.

After a successful completion, the binary RPM will be ready to be deployed on all blades. To install the binary RPM execute:

```
rpm -ivh collector-9.1.0.0.0.rpm
```

After the binary RPM is successfully deployed, the compiling package must be removed from the blade with:

```
rpm -e cna_collector_compile
```

This operation will remove the generated binary RPM, so it should be backed up somewhere else if it's necessary to preserve it.

4.2.3 Building the centralised DB Loader

The centralised DB Loader RPM is:

```
CNA/dist/CNA/rpm/streams/cloader-9.1.0.0.0.compile.rpm
```

The package can then be built by issuing the following command as root in the same directory as the RPM:

```
rpm -ivh cloader-9.1.0.0.0.compile.rpm --prefix $PWD
```

The binary RPM will be created as cloader-9.1.0.0.0.rpm; if the operation fails, the file will still be present, but unusable, and the package will have to be removed (see later).

The compilation log, named cloader_*datetime*.log (where *datetime* is 14 digits representing the date and time when the compilation was started) is produced in the same directory as the output RPM, and it contains the selected customisation options. Note that it is not automatically removed, so it should be cleaned manually when no longer relevant.

After a successful completion, the binary RPM will be ready to be deployed on all blades. To install the binary RPM execute:

```
rpm -ivh cloader-9.1.0.0.0.rpm
```

After the binary RPM is successfully deployed, the compiling package must be removed from the blade with:

```
rpm -e cna_cloader_compile
```

This operation will remove the generated binary RPM, so it should be backed up somewhere else if it's necessary to preserve it.

4.2.4 Building the local DB Loader

The local DB Loader RPM is:

```
CNA/dist/CNA/rpm/streams/loader-9.1.0.0.0.compile.rpm
```

A configuration file is provided to specify additional options; it's named loader-9.1.0.0.0.compile.ini in the same directory as the RPM, as specified in section 0. The file contains all options commented out, and customised values can be specified by uncommenting and modifying the desired value: one important exception is the *Interface* option, as leaving it commented will produce a DB Loader that collects from all ITE (aggregation) directories, but not RawTDR. The package can then be built by issuing the following command as root in the same directory as the RPM:

```
rpm -ivh loader-9.1.0.0.0.compile.rpm --prefix $PWD
```

The binary RPM will be created as loader-9.1.0.0.0.rpm; if the operation fails, the file will still be present, but unusable, and the package will have to be removed (see later).

The compilation log, named loader_*datetime*.log (where *datetime* is 14 digits representing the date and time when the compilation was started) is produced in the same directory as the output RPM,

and it contains the selected customisation options. Note that it is not automatically removed, so it should be cleaned manually when no longer relevant.

After a successful completion, the binary RPM will be ready to be deployed on all blades. To install the binary RPM execute:

```
rpm -ivh loader-9.1.0.0.0.rpm
```

After the binary RPM is successfully deployed, the compiling package must be removed from the blade with:

```
rpm -e cna_loader_compile
```

This operation will remove the generated binary RPM, so it should be backed up somewhere else if it's necessary to preserve it.

In case the DB Loader needs to be compiled for more than one interface (e.g. gn and raw_gn), the steps are as follow:

1. Configure the desired interface as the value of the *Interface* parameter in the configuration file loader-9.1.0.0.0.compile.ini
2. Build the package with
   `rpm -ivh loader-9.1.0.0.0.compile.rpm --prefix $PWD`
3. Move the output file `loader-9.1.0.0.0.rpm` to a different name or install it with
   `rpm -ivh loader-9.1.0.0.0.rpm`
4. Remove the build package with `rpm -e cna_loader_compile`
5. Restart at step 1 for the next interface

If at step #3 it's been chosen to rename the files, the final step would be to deploy the files to the blades where they need to be installed.

4.2.5 Building Lookup Manager

The LookupManager RPM is:

`CNA/dist/CNA/rpm/streams/lookup_manager-9.1.0.0.0.compile.rpm`

A configuration file is provided to specify additional options; it's named lookup_manager-9.1.0.0.0.compile.ini in the same directory as the RPM, as specified in section 4.4.1. The file contains all options commented out, and customised values can be specified by uncommenting and modifying the desired value. The package can then be built by issuing the following command as root in the same directory as the RPM:

```
rpm -ivh lookup_manager-9.1.0.0.0.compile.rpm --prefix $PWD
```

The binary RPM will be created as lookup_manager-9.1.0.0.0.rpm; if the operation fails, the file will still be present, but unusable, and the package will have to be removed.

The compilation log, named lookup_manager_{datetime}.log (where {datetime} is 14 digits representing the date and time when the compilation was started) is produced in the same directory as the output RPM, and it contains the selected customisation options. Note that it is not automatically removed, so it should be cleaned manually when no longer relevant.

After a successful completion, the binary RPM will be ready to be deployed on all blades. To install the binary RPM execute:

```
rpm -ivh lookup_manager-9.1.0.0.0.rpm
```

After the binary RPM is successfully deployed, the compiling package must be removed from the blade with:

```
rpm -e cna_lookup_manager_compile
```

This operation will remove the generated binary RPM, so it should be backed up somewhere else if it's necessary to preserve it.

4.2.6 Building ITE

The ITE RPMs are (one per interface):

```
CNA/dist/CNA/rpm/streams/ite_gn-9.1.0.0.0.compile.rpm
CNA/dist/CNA/rpm/streams/ite_lte-9.1.0.0.0.compile.rpm
CNA/dist/CNA/rpm/streams/ite_voice-9.1.0.0.0.compile.rpm
CNA/dist/CNA/rpm/streams/ite_volte-9.1.0.0.0.compile.rpm
CNA/dist/CNA/rpm/streams/ite_fixedline-9.1.0.0.0.compile.rpm
CNA/dist/CNA/rpm/streams/ite_ran-9.1.0.0.0.compile.rpm
CNA/dist/CNA/rpm/streams/ite_gb-9.1.0.0.0.compile.rpm
CNA/dist/CNA/rpm/streams/ite_s1mme_s6a-9.1.0.0.0.compile.rpm
CNA/dist/CNA/rpm/streams/ite_iups-9.1.0.0.0.compile.rpm
CNA/dist/CNA/rpm/streams/ite_gx_gy-9.1.0.0.0.compile.rpm
```

A configuration file is provided to specify additional options; it's named ite_*interface*-9.1.0.0.0.compile.ini (where *interface* is the ITE interface to build) in the same directory as the RPM, as specified in section 4.4.1. The file contains all options commented out, and customised values can be specified by uncommenting and modifying the desired value. The package can then be built by issuing the following command as root in the same directory as the RPM:

```
rpm -ivh ite_interface-9.1.0.0.0.compile.rpm --prefix $PWD
```

The binary RPM will be created as ite_*interface*-9.1.0.0.0.rpm; if the operation fails, the file will still be present, but unusable, and the package will have to be removed (see later).

The compilation log, named ite_*interface_datetime*.log (where *interface* is the compiled interface and *datetime* is 14 digits representing the date and time when the compilation was started) is produced in the same directory as the output RPM, and it contains the selected customisation options. Note that it is not automatically removed, so it should be cleaned manually when no longer relevant.

After a successful completion, the binary RPM will be ready to be deployed on all blades. To install the binary RPM execute:

```
rpm -ivh ite_interface-9.1.0.0.0.rpm
```

After the binary RPM is successfully deployed, the compiling package must be removed from the blade with:

```
rpm -e cna_ite_interface_compile
```

This operation will remove the generated binary RPM, so it should be backed up somewhere else if it's necessary to preserve it.

4.2.7 Building RawTDR

The RawTDR RPMs are (one per interface):

```
CNA/dist/CNA/rpm/streams/rawtdr_gn-9.1.0.0.0.compile.rpm
CNA/dist/CNA/rpm/streams/rawtdr_lte-9.1.0.0.0.compile.rpm
CNA/dist/CNA/rpm/streams/rawtdr_voice-9.1.0.0.0.compile.rpm
CNA/dist/CNA/rpm/streams/rawtdr_fixedline-9.1.0.0.0.compile.rpm
```

A configuration file is provided to specify additional options; it's named rawtdr_*interface*-9.1.0.0.0.compile.ini (where *interface* is the ITE interface to build) in the same directory as the RPM, as specified in section 0. The file contains all options commented out, and customised values can be specified by uncommenting and modifying the desired value. The package can then be built by issuing the following command as root in the same directory as the RPM:

```
rpm -ivh rawtdr_interface-9.1.0.0.0.compile.rpm --prefix $PWD
```

The binary RPM will be created as rawtdr_*interface*-9.1.0.0.0.rpm; if the operation fails, the file will still be present, but unusable, and the package will have to be removed (see later).

The compilation log, named rawtdr_*interface*_*datetime*.log (where *interface* is the compiled interface and *datetime* is 14 digits representing the date and time when the compilation was started) is produced in the same directory as the output RPM, and it contains the selected customisation options. Note that it is not automatically removed, so it should be cleaned manually when no longer relevant.

After a successful completion, the binary RPM will be ready to be deployed on all blades. To install the binary RPM execute:

```
rpm -ivh rawtdr_interface-9.1.0.0.0.rpm
```

After the binary RPM is successfully deployed, the compiling package must be removed from the blade with:

```
rpm -e cna_rawtdr_interface_compile
```

This operation will remove the generated binary RPM, so it should be backed up somewhere else if it's necessary to preserve it.

4.2.8 Listing and removing packages

The installed Streams RPMs can be listed with:

```
rpm -qa | grep cna_ | sort
```

An example output is:

```
[streamsadmin@streamsqse ~]$ rpm -qa | grep cna
cna_ite_voice-9.1.0.0.0-r604.x86_64
cna_lm-9.1.0.0.0-r604.x86_64
```

Note that the release number (604 in the example) is the SVN revision number of the source code.

In order to remove all packages, you can execute the following command as *root*:

```
rpm -qa | grep cna_ | xargs rpm -e
```

For Vedia Supercollector installation please refer to the next section.

4.3 Streams domain and instance configuration

For the setup of the streams domains there are 2 choices:

1. Separated domains on each blade with embedded zookeeper
2. One enterprise domain for all blades with external zookeeper

Also, ITE application and RawTDR application use different Streams instances.

4.3.1 Zookeeper setup

There are two options for a Zookeeper:

- Embedded (simpler)
- External (better performance)

4.3.1.1 Embedded Zookeeper

No installation is required to use the embedded Zookeeper. The embedded Zookeeper is already part of the Streams 4.1.1 installation. You need to remove the environment variable '`STREAMS_ZKCONNECT`' and to specify the option `--embeddedzk` when creating a Streams domain using the `mkdomain` command (see section 4.3.2), i.e.

```
unset STREAMS_ZKCONNECT
streamtool mkdomain -d cna_domain --embeddedzk
```

It is needed to increase the heap size using the following command before any Streams domain is created:

```
streamtool setbootproperty streams.zookeeper.jvmFlags=-Xmx4096m

To setup the streams domain and instance execute:
   1. Create the checkpoint directoy:
      mkdir -p /home/boss/CheckPtRepository
   2. Set boot property for zookeeper
      streamtool setbootproperty streams.zookeeper.jvmFlags=-Xmx4096m
   3. Make the domain:
      streamtool mkdomain --embeddedzk --property SWS.Port=8443 --property
      JMX.Port=9443 --property domain.highAvailabilityCount=1 --property
      domain.checkpointRepository=fileSystem --property
```

```
   domain.checkpointRepositoryConfiguration=" { \"Dir\" :
   \"/home/boss/CheckPtRepository\" } "
4. Generate the key:
   streamtool genkey –embeddedzk
5. Start the domain:
   streamtool startdomain –embeddedzk
6. Make the instance:
   streamtool mkinst --embeddedzk (-i instancename)
7. Start instance:
   streamtool startinst –embeddedzk (-i instancename)
```

4.3.1.2 External Zookeeper

An external Zookeeper can deliver potentially better performance.

4.3.1.2.1 Installing the external Apache Zookeeper

Ideally, the installation should be performed in a host where there is no other Streams domain. The recommended host is the host where Vedia Supercollector is installed (see section 3.2).

After untarring the Zookeeper package (zookeeper-3.4.6.tar) into `/opt/tnf/cna/zookeeper-3.4.6/` set the following parameters in the config file `/opt/tnf/cna/zookeeper-3.4.6/conf/zoo.cfg`:

```
dataDir=/opt/tnf/cna/zookeeper-3.4.6/data
clientPort=2181
maxClientCnxns=0
```

Then the heap size of Zookeeper needs to be increased by putting a file `java.env` in the configuration directory (`/opt/tnf/cna/zookeeper-3.4.6/conf/java.env`) with this command:

```
export JVMFLAGS="-Xmx4096m"
```

To **start** the Apache Zookeeper:

```
cd /opt/tnf/cna/zookeeper-3.4.6/bin
./zkServer.sh start /opt/tnf/cna/zookeeper-3.4.6/conf/zoo.cfg
```

To **stop** the Apache Zookeeper:

```
cd /opt/tnf/cna/zookeeper-3.4.6/bin
./zkServer.sh stop /opt/tnf/cna/zookeeper-3.4.6/conf/zoo.cfg
```

Next step is to define an environment variable $STREAMS_ZKCONNECT in all hosts where streams is going to be deployed. It must be set to the IP and port of the running Zookeeper instance, e.g.

```
export STREAMS_ZKCONNECT=blade01:2181
```

On each of the Streams blades, set the environment variable $STREAMS_ZKCONNECT to the same value (in a .profile or a .bash_profile file)

4.3.2 Streams main domain

If using the **embedded Zookeeper** then each Streams host will be independent and Streams domains can have the same name; the default name `cna_domain` is recommended. This is the **recommended and tested setup**.

If using the external Zookeeper then Streams hosts have to be linked to the external Zookeeper and they all need to be part of one Streams domain (the name `cna_domain` is recommended) or grouped into multiple Streams domains (that have different names).For more information, see Streams IBM Knowledge Center on how to setup the application on multiple resources.

Provided that the environment variable $STREAMS_ZKCONNECT is correctly configured (i.e. it is not set for the embedded Zookeeper), the domain can be set up with the following commands (where *cna_domain* is the domain to be created):

```
streamtool mkdomain -d cna_domain
streamtool genkey -d cna_domain
streamtool startdomain -d cna_domain
```

The environment variable $STREAMS_DOMAIN_ID should be set to this Streams domain (in a .profile or a .bash_profile file) in every blade so that it is not necessary to specify it when running a Streams command every time.

Note `-d cna_domain` in the commands above can be avoided if $STREAMS_DOMAIN_ID is set.

4.3.3 Streams instances

There has to be a different Streams instance per blade/per interface/per application type (ITE or Raw TDR processor). This Streams instance will be also used to host the local DB loader (if it is enabled). Also, there needs to be additional Streams instance for the Streams Collector (super collector or local collector); one for each host. Additionally, the Centralized DB Loader needs its own Streams instance in the blade where it's deployed only.

In a deployment where only one Streams interface (i.e. only one blade, one interface) is configured that one Streams instance can be set to an environment variable $STREAMS_INSTANCE_ID so that it is not necessary to type it with every Streams command. This instance will be also used for the Streams Collector instance and can be used for the Centralized DB Loader instance, too.

Note that the `-i instance` in the commands bellow can be avoided if $STREAMS_INSTANCE_ID is set.

4.3.3.1 Streams Collector instance

As for the Collector instance, there must be **one instance** in each Streams blade to support **all interfaces** in that blade.

The recommended name is e.g. `cna_instance_blade02_collector`, i.e. it should reflect the blade and that it is the collector instance. The same name can be used for both the "super collector" and "local collector" modes (you can choose `cna_instance_blade02_supercollector` for the "super collector" mode if you prefer).

Names of this Streams instance is to be set during the Collector customization; see section 4.4.2. Also, when administering (start/stop/status) the "super collector" mode of the Collector you need to use this instance, see section 5.1.

4.3.3.2 Streams Centralized DB Loader instance

If there is one host that hosts the Centralized DB Loader module then a Streams instance also needs to be created for this. There can be one instance per interface or one instance for all interfaces; the choice is up to the operator. The preferred choice would be to have just one instance, because the operator can control different interfaces on the job level anyway (see the administration section). Please note that it is possible to collocate the Centralized DB Loader with other modules, too.

The recommended instance name is e.g. `cna_instance_blade02_cloader`, i.e. it should reflect the blade and that it is the Centralized DB Loader instance.

There is no need to set the name of this Streams instance during the Centralized DB Loader customization. When administering (start/stop/status) the Centralized DB you need to use this instance, see section 5.1.

4.3.3.3 Streams ITE/RawTDR instance

There has to be one Streams instance per blade/per interface/per application type (ITE or Raw TDR processor) and might be sensible to choose a name that reflects this, e.g. a combination of blade name, application type and interface, e.g. `cna_instance_blade02_ite_gn`.

The list of all Streams ITE/RawTDR instances is to be set when customizing the ITE or RawTDR applications, see sections 4.4.4.1 and 4.4.5.1.

4.3.3.4 Operating Streams instances

This section details all operations with a Streams instance and it applies to all the Streams instance types described above (Collector, Centralized DB Loader and ITE/RawTDR).

Run the following commands to **create** and **start** the instance (where *cna_instance_blade02_ite_gn* is the name of the interface to be created and *blade02* is one of the blade's external IP addresses or a host name pointing to it):

```
streamtool mkinstance -i cna_instance_blade02_ite_gn --hosts blade02
streamtool startinstance -i cna_instance_blade02_ite_gn
```

Before switching off or rebooting the blade, it's recommended to manually **stop** the instance with:

```
streamtool stopinstance -i cna_instance_blade02_ite_gn
```

In case the blade was rebooted without stopping the instance, it can be **stopped forcefully** with:

```
streamtool stopinstance -i cna_instance_blade02_ite_gn --force
```

Afterwards it can be restarted with:

```
streamtool startinstance -i cna_instance_blade02_ite_gn
```

You can also **list** all created instances with:

```
streamtool lsinstance
```

Or list all instances for one domain by adding the `-d cna_domain` option:

```
streamtool lsinstance -d cna_domain
```

4.3.4 Sizing considerations

By default the ITE applications use 4 parallel collection chains and a maximum of 5 parallel aggregation contexts (configurable at compile-time as specified in section 0) divided in 3 groups for Gn and LTE (UP, UPS, CP+CPS) and 2 for Voice&SMS (VSMS, VSMSStatus); how the contexts are split between the groups is configurable as per section 4.4.4, but by default 3 contexts are used for UP, 4 for VSMS and 1 for the others.

With the above setup the tested throughput per probe is:

For Gn: 4 million TDR/min on an HP blade and 6.5 million TDR/min on a Flex blade; performances degrade gradually when increasing the number of probes, and the maximum recommended amount of fully loaded probes is 3 on an HP blade (totaling 7.5 million TDR/min) and 4 on a Flex blade (totaling 16 million TDR/min).

For LTE: with the default setup (see 4.4.1) 6.8 million TDR/min on a Flex blade with one probe job. Performance decreases when increasing the number of probes. The maximum recommended amount of probes is 2 on a Flex blade, with 5 million TDR/min each. Totalling 10 million TDR/min.

For Voice/SMS: Using 6 parallel chains and 3 contexts, 3.4 million TDR/min for a single probe (job) on a Flex blade; 6.1 million TDR/min for two probes (e.g. A and IuCS). Performances degrade gradually when increasing the number of probes, and the maximum recommended amount of fully loaded probes is 4 (totaling 10.1 million TDR/min)

If the probes' throughput is lower than 1 million TDR/min, the ITE applications can be configured with appSplitterEnabled=false in the constants file (see section 4.4.4), so that all aggregation is performed in the same context and more probes than listed above can be processed by the same blade.

The application's footprint can be further decreased by reducing the overall maximum number of contexts (NumContexts=1) and chain processors (NumChains=1), but this configuration is a compile-time parameter (see section 0).

4.4 Customization

The following is a list of customization files and parameters for all interfaces and all Streams components. Please review all of them and change the configuration as needed.

There are two types of customization parameters:

- **Compilation** time: Parameters that need to be set before a Streams component is compiled. This is covered in the section 4.4.1.
- **Submission** time: Parameters that need to be set before a Streams component is started. Unless otherwise stated these parameters are defined in files located in `/opt/tnf/cna/cfg/`.

4.4.1 Compile-time parameters

The Streams RPMs (see section 4.2) accept a configuration file at the path indicated by the --prefix option; it must be named as the RPM file but with suffix .ini, so e.g. `loader-9.1.0.0.0.compile.rpm` will use a configuration file named `loader-9.1.0.0.0.compile.ini`. The content must be a list of options in the form A=B, one per line, with blank lines and comments (starting with '#') allowed; the following table lists the available options.

4.4.1.1 DB Loader

Parameter	Description
HadoopSegments **Interface**	It specifies how many segments are configured in Hadoop (default **10**) It specifies the interface the app will load: possible values are gn, lte, voice, volte, ran, fixedline, gb, iups, s1mme_s6a, raw_gn, raw_lte, raw_voice, raw_fixedline. To load more than one interface the package must be built and installed once per interface.

4.4.1.2 Lookup Manager

Parameter	Description
NumAggProbes	It specifies how many probes can be configured at most per each interface in probes.csv, i.e. how many ITE jobs can run in the same instance (see next section, default **4**)
CellCacheSizeMB	It specifies how much space to allocate for the Cell lookup, in megabytes (default **50**)
MCCMNCCacheSizeMB	It specifies how much space to allocate for the MCC-MNC lookup, in megabytes (default **1**)
BNumberCacheSizeMB	It specifies how much space to allocate for the BNumber lookup, in megabytes (default **30**)
StatusCacheSizeMB	It specifies how much space to allocate for the Status code lookup, in megabytes (default **2**)
GnQoSCacheSizeMB	It specifies how much space to allocate for the Gn QoS category lookup, in megabytes (default **3**)
LTEQoSCacheSizeMB	It specifies how much space to allocate for the LTE QoS category lookup, in megabytes (default **3**)
SGSNCacheSizeMB	It specifies how much space to allocate for the SGSN lookup, in megabytes (default **1**)
GGSNCacheSizeMB	It specifies how much space to allocate for the GGSN lookup, in megabytes (default **1**)
SGWCacheSizeMB	It specifies how much space to allocate for the SGW lookup, in megabytes (default **1**)
BSCCacheSizeMB	It specifies how much space to allocate for the BSC lookup, in megabytes (default **1**)
RNCCacheSizeMB	It specifies how much space to allocate for the RNC lookup, in megabytes (default **1**)
MMECacheSizeMB	It specifies how much space to allocate for the MME lookup, in megabytes (default **1**)
BTSCacheSizeMB	It specifies how much space to allocate for the BTS lookup, in megabytes (default **1**)

NodeBCacheSizeMB	It specifies how much space to allocate for the NodeB lookup, in megabytes (default **1**)
EnodeBCacheSizeMB	It specifies how much space to allocate for the eNodeB lookup, in megabytes (default **1**)
AppCacheSizeMB	It specifies how much space to allocate for the Applications lookup, in megabytes (default **10**)
SipStatusCacheSizeMB	It specifies how much space to allocate for the SIP status lookup, in megabytes (default **1**)
AppStatusCacheSizeMB	It specifies how much space to allocate for the ApplicationStatus lookup, in megabytes (default 1)
DisableAllThreads	Disable (1) or enable (**0**) the full thread pool. It's recommended to use all threads, but on poorly performing systems (e.g. VM) it might be necessary to reduce the amount of threads.
KeepSources	Set this parameter to "1" in order to keep the source code after the package installation. The source can be used to implement additional adapters. See the Adapter Development Guide.

4.4.1.3 ITE

Parameter	Description
HadoopSegments	It specifies how many segments are configured in Hadoop (default **10**)
NumChains	It specifies how many parallel collection chains to use (default **6**)
NumAggProbes	It specifies how many probes can be configured at most per each interface in probes.csv (see next section, default **4**)
AdditionalAdapters	Set the value of this parameter to an archive file in TGZ format, that contains source code for additional adapters. The new adapters will be compiled into the application. See the Adapter development guide for more details.
KeepSources	Set this parameter to "1" in order to keep the source code after the package installation. The source can be used to implement additional adapters. See the Adapter Development Guide.

The following options apply to **Gn/LTE** only:

Parameter	Description
NumParallelParsers	Number of parallel chain processors (default **3**)
NumUPSubAggregators	Number of aggregators for UP subscriber level (default **6**)
NumUPAppAggregators	Number of aggregators for UP application level (default **2**)
NumUPCellAggregators	Number of aggregators for UP cell level (default **2**)
NumUPNetAggregators	Number of aggregators for UP network level (default **2**)
NumUPSSubAggregators	Number of aggregators for UPS subscriber level (default **1**)
NumUPSNetAggregators	Number of aggregators for UPS network level (default **1**)
NumCPSubAggregators	Number of aggregators for CP subscriber level (default **1**)
NumCPCellAggregators	Number of aggregators for CP cell level (default **1**)
NumCPNetAggregators	Number of aggregators for CP network level (default **1**)
NumCPSSubAggregators	Number of aggregators for CPS subscriber level (default **1**)
NumCPSCellAggregators	Number of aggregators for CPS cell level (default **1**)
NumCPSNetAggregators	Number of aggregators for CPS network level (default **1**)

minFields6013,maxFields6013, minFields6021,maxFields6021, minFileds6022,maxFields6022	Used only for Gn interface ITE app. Specify the minimum and maximum number of CSV fields, expected in input files of the given type. TDRs with less fields or more fields than expected are rejected.
minFields7013,maxFields7013, minFields7021,maxFields7021, minFields7022,maxFields7022	Used only for LTE interface ITE app. Specify the minimum and maximum number of CSV fields, expected in input files of the given type. TDRs with less fields or more fields than expected are rejected.
CesSupported	Enable/Disable customer experience scores (yes/no)

The following options apply to **GN/LTE** when **DnsKPIs=yes** only:

Parameter	Description
DnsKPIs	Enable/disable DNS KPI's aggregations (default **no**)
NumDNSSubAggregators	Number of aggregators for DNS subscriber level (default **1**)
NumDNSNetAggregators	Number of aggregators for DNS network level (default **1**)
NumDNSSSubAggregators	Number of aggregators for DNS status subscriber level (default **1**)
NumDNSSNetAggregators	Number of aggregators for DNS status network level (default **1**)

The following options apply to **Voice&SMS** only:

Parameter	Description
NumContexts	It specifies how many aggregation contexts to use (default **5**)
minFieldsVoice,maxFieldsVoice	Used only for Voice/SMS interface ITE app. Specify the minimum and maximum number of CSV fields, expected in input file. TDRs with less fields or more fields than expected are rejected.

The following options apply to **RAN** only:

Parameter	Description
NumParallelParsers	Number of parallel chain processors (default **3**)
NumTopAggregators	Number of aggregators for top level (default **6**)
NumSubAggregators	Number of aggregators for subscriber level (default **6**)
NumDevAggregators	Number of aggregators for device level (default **4**)
NumCellAggregators	Number of aggregators for cell level (default **2**)
NumNetAggregators	Number of aggregators for network level (default **4**)
minFieldsRan,maxFieldsRan	Used only for RAN interface ITE app. Specify the minimum and maximum number of CSV fields, expected in input file. TDRs with less fields or more fields than expected are rejected.

The following options apply to **FixedLine** only:

Parameter	Description
NumParallelParsers	Number of parallel chain processors (default **3**)
NumUPSubAggregators	Number of aggregators for UP subscriber level (default **6**)
NumUPAppAggregators	Number of aggregators for UP application level (default **2**)
NumUPNetAggregators	Number of aggregators for UP network level (default **2**)
NumUPSSubAggregators	Number of aggregators for UPS subscriber level (default **1**)
NumUPSNetAggregators	Number of aggregators for UPS network level (default **1**)
NumCPSubAggregators	Number of aggregators for CP subscriber level (default **1**)

NumCPNetAggregators	Number of aggregators for CP network level (default **1**)
NumCPSSubAggregators	Number of aggregators for CPS subscriber level (default **1**)
NumCPSNetAggregators	Number of aggregators for CPS network level (default **1**)
minFields4013,maxFields4013, minFields4021,maxFields4021, minFields4022,maxFields4022	Specify the minimum and maximum number of CSV fields, expected in input files of the given type. TDRs with less fields or more fields than expected are rejected.
CesSupported	Enable/Disable customer experience scores (yes/no)

The following options apply to **Gb** only:

Parameter	Description
minFieldsGb, maxFieldsGb	Specify the minimum and maximum number of CSV fields, expected in input files of the given type. TDRs with less fields or more fields than expected are rejected.

The following options apply to **IuPS** only:

Parameter	Description
minFieldsIuPS, maxFieldsIuPS	Specify the minimum and maximum number of CSV fields, expected in input files of the given type. TDRs with less fields or more fields than expected are rejected.

The following options apply to **S13/SGs** only:

Parameter	Description
NumParallelParsers	Number of parallel chain processors (default **3**)
NumS13SubAggregators	Number of aggregators for S13 subscriber level (default **1**)
NumS13DevAggregators	Number of aggregators for S13 device level (default **1**)
NumS13NetAggregators	Number of aggregators for S13 network level (default **1**)
NumS13SSubAggregators	Number of aggregators for S13 status subscriber level (default **1**)
NumS13SDevAggregators	Number of aggregators for S13 status device level (default **1**)
NumS13SNetAggregators	Number of aggregators for S13 status network level (default **1**)
NumSGsSubAggregators	Number of aggregators for SGs subscriber level (default **1**)
NumSGsDevAggregators	Number of aggregators for SGs device level (default **1**)
NumSGsNetAggregators	Number of aggregators for SGs network level (default **1**)
NumSGsSSubAggregators	Number of aggregators for SGs status subscriber level (default **1**)
NumSGsSDevAggregators	Number of aggregators for SGs status device level (default **1**)
NumSGsSNetAggregators	Number of aggregators for SGs status network level (default **1**)

4.4.1.4 RawTDR

Parameter	Description
NumChains	It specifies how many parallel collection chains to use (default for **Gn/LTE 4,** default for **Voice 6**)
minFieldsVoice,maxFieldsVoice	Used only for Voice/SMS interface RawTDR app. Specify the minimum and maximum number of CSV fields, expected in input file. TDRs with less fields or more fields than expected are rejected.

KeepSources	Set this parameter to "1" in order to keep the source code after the package installation. The source can be used to implement additional adapters. See the Adapter Development Guide.

4.4.2 Collector

There are two default configuration XML files in `/opt/tnf/cna/cfg/`:

1. `collector.xml` – This is to be used for the "super collector" mode. I.e. the Collector deployment that is in a standalone host and collects feed from multiple Sourceworks probes. This configuration file is used when the Collector is started using the `collector.sh` script.

2. `collector_local.xml` – This is to be used for the "local collector" mode. I.e. the Collector deployment that is in every Streams host and is feeding individual Streams ITE or RawTDR jobs. In this mode the Collector connects to the "super collector" dedicated host (as per the mode above) or to the probe directly.
   This configuration file is used when the Collector is automatically started using the `cna.sh` script.

The structure of both files is identical, but the default parameters are better suited for each of these two modes. There are global and per-collector parameters.

There is also a possibility to call post download scripts to:

- Create symlinks for one or more Streams targets (ITE, RawTDR, 3rd party)
- Adapt feed
- Combination of both the above

If Voice/SMS feed adaption is needed then also the interface adaptors need to be deployed.

4.4.2.1 Streams instance

By default, the Streams instance for the Collector is the one defined by the environment variable $STREAMS_INSTANCE_ID or $COLLECTOR_STREAMS_INSTANCE_ID where the later takes precedence over the first one.

For the "local collector" mode, there are two ways how to set one or both:

- In the shell login profile

- In the file `/opt/tnf/cna/cna.env`

Alternatively, it can be also specified when running the cna.sh admin command (see section 5.1) using the argument -ci {InstanceName}, .e.g.

```
/opt/tnf/cna/cna.sh -ci cna_instance_blade02_collector start
```

As for the "super collector" mode, the only way how to set the instance is by defining it in the shell login profile. Alternatively, it can be also specified when running the collector.sh admin command (see section 5.1) using the argument -i {InstanceName}, .e.g.

```
/opt/tnf/cna/collector.sh -i cna_instance_blade02_supercollector start
```

4.4.2.2 Run local Collector automatically from cna.sh

By default, the "local mode" of the Collector is operated automatically using the cna.sh script (see the admin section 5). This includes starting, stopping, checking status and cleaning.

This functionality can be disabled if needed. To disable it please add `RUN_COLLECTOR=0` to the file `/opt/tnf/cna/cna.env`. To enable the functionality again please either remove the above or set it to `RUN_COLLECTOR=1`.

If the local Collector is disabled from cna.sh then it can be operated by adding the `-l` option to the `collector.sh` script (please see the admin section 5 for more details).

4.4.2.3 Collectors section

There can be multiple <collector> sections within the <collectors> section. One <collector> section defines one Collector job that downloads feed from one probe and one remote directory.

Parameters within the <collector> section are these by default:

Parameter	Description
collector_name	Unique name of the job. The naming convention is Collector_{ITE/RawTDR}_{Interface}{FeedType}_{ProbeName}. E.g. **Collector_ITE_GnNetro_P1** Or **Collector_GnNetro_P1** (for the "collector" mode)
Hostname	Remote host name or IP address
Username	Username to login to the remote host
Password	Password to login to the remote host
source_dir	Remote directory to download from. If set to the same as the target_dir (see the parameter below) it indicates that this job will download feed from the "super collector" host.
target_dir	Local directory for downloaded feed. For both the "super collector" and "local collector" mode it should be /opt/tnf/cna/tdrs/{ProbeName}/{FeedType}/ E.g. **/opt/tnf/cna/tdrs/P1/netro/** This is where the downloaded physical files are to be stored. If symlinks are to be created (using the external_command parameter below) then they will be stored in /opt/tnf/cna/tdrs/{SymlinkDir}/{ProbeName}/in/data/{FeedType}/ E.g. **/opt/tnf/cna/tdrs/Streams-ITE/P1/in/data/netro/** The {ProbeName} above is the unique identifier of the Sourceworks probe. Different feeds of the same probe can have the same {ProbeName} (typically Gn/LTE netro & pagebased, or Voice/SMS Aif/IuCS). The base of the target directory (i.e. without {ProbeName}/{FeedType}/) needs to match to the ITE and RawTDR input directories as defined in Streams in configuration files probes.csv and probes-tdr.csv, see sections 4.4.4.1 and 4.4.5.1.

search_pattern	Regular expression matching the file names; must include a timestamp between parentheses in the format below. E.g. **netro_([0-9]{14})\\.log\\.gz**
filetime_extract_pattern	Needs to match to the search_pattern above
filetime_extract_format	Timestamp format as found in the collected file names E.g. **YYYYMMDDhhmmss**
external_command	Script to execute for each downloaded file. The purpose of this script is to: - Create symlinks for multiple Streams targets (ITE, RawTDR) - Adapt feed using the external Java-based interface adapter - Combination of both of the above The following scripts can be used out of the box: - **/opt/tnf/cna/etc/script/create-symlinks.sh** *{SymlinkList}* - **/opt/tnf/cna/etc/script/call-interface-adapter-voice.sh** *{TargetBaseDir} {AdapterDataFeedId}* - **/opt/tnf/can/etc/script/call-interface-adapter-voice-symlinks.sh** *{TargetBaseDir} {AdapterDataFeedId} {SymlinkList}* Where {SymlinkList}, {TargetBaseDir} and {AdapterDataFeedId} are parameters that need to be defined. Typically: - {SymlinkList} is Streams-ITE,Streams-RawTDR - {TargetBaseDir} is /opt/tnf/cna/tdrs/adapted/ - {AdapterDataFeedId} is either polystar_a_v1 or polystar_iucs_v1

4.4.2.4 Global section

All parameters defined inside the <global> section apply to all the Collector jobs defined using the <collector> sections as per the paragraph above (unless overridden).

Parameter	Description
protocol	Remote host connection protocol Supported protocols: ftp, ftpSSLAll, ftpSSLControl, ftpSSLTry, ftps, sftp Default local collector: **ftp** Default collector: **sftp** In case of sftp the path of source directory is always an absolute path, while in case of ftp it's relative unless it starts with a double slash. Note that ftp has much better performance than sftp.
control_dir	Directory to store control and persistence files (no need to modify) Default: **/opt/tnf/cna/control/collector/**
statistics_dir	Directory to store statistics and error log files (no need to modify) Default: **/opt/tnf/cna/stat/collector/**
search_pattern_match_count	The number of brackets in the source file pattern (no need to modify) Default: **0**
dynamic_config_scan_period	Scan interval for changes in configuration files [seconds]. Currently this only supports password changes, nothing else (no need to modify) Default: **60.0**
max_fill_level	The maximum fill level of the target file system in percentages. When the fill level exceeds this value file transfers are paused

	Default: **98**
get_fill_level_command	Defines a command to evaluate the disk fill level if the above parameter is set (no need to modify) Default: **runTimeScripts/checkDiskSpace.sh**
statistics_archive_dir	Directory to archive statistics and error log files after one day (no need to modify) Default: **/opt/tnf/cna/stat/collector/archive/**
no_connections	The number of parallel connections to be used for the file transfer Default: **4**
min_line_speed	Minimal line speed that is required for a successful file transfer [kbit/sec] Default: **64**
connection_timeout	Maximum time in seconds that you allow the connection to the server to take. This only limits the connection phase Default: **30**
dir_scan_timeout	Maximum time in seconds that you allow the remote directory scan to complete Default: **120**
ftp_command_timeout	Maximum time in seconds that you allow an ftp command to complete Default: **60**
time_shift	Time difference between the local host and the server [seconds] Default: **0**
verbose	The verbosity of the ftp operation execution. If set to true it may expose passwords! This parameter is enabled only when the compile time parameter ENABLE_CURL_VERBOS is true Default: **false**
useEPSV	If true, it tells curl to use the EPSV command when doing passive FTP downloads. Using EPSV means that it will first attempt to use EPSV before using PASV, but if you pass false to this option, it will not try using EPSV, only plain PASV. If the server is an IPv6 host, this option will have no effect (no need to modify) Default: **true**
useEPRT	If true, it tells curl to use the EPRT (and LPRT) command when doing active FTP downloads (which is enabled by ftpPORT) default true. Using EPRT means that it will first attempt to use EPRT and then LPRT before using PORT, but if you pass false to this option, it will not try using EPRT or LPRT, only plain PORT. If the server is an IPv6 host, this option will have no effect (no need to modify) Default: **true**
usePORT	Used to get the IP address to use for the FTP PORT instruction (no need to modify) Default:
skipPASVIp	If true, it instructs libcurl to not use the IP address the server suggests in its 227-response to libcurl's PASV command when libcurl connects the data connection. Instead libcurl will re-use the same IP address it already uses for the control connection. But it will use the port number from the 227-response (no need to modify) Default: **false**
force_close_connection	Defines the connection handling after file transfer. If false then no connection is forced to be closed. The connections may be closed after a certain idle time.

	If true then connections are forced to be closed after a single scan cycle (no need to modify) Default: **false**
month	Scheduler to start scanning the remote directory using the crontab syntax. Month of the year from 1..12 (no need to modify) Default: "*"
mday	Day of month 1..31 (no need to modify) Default: "*"
wday	Day of the week from 0=Sunday .. 6=Saturday (no need to modify) Default: "*"
hour	Hour of the day from 0..23 (no need to modify) Default: "*"
minute	Minute from 0..59 (no need to modify) Default: "*"
tensec	Scheduler to start scanning the remote directory using the crontab syntax. Ten second interval from 0..5 Default: **1** (i.e. scan every ten seconds)
initial_trigger	Specifies whether a scan cycle is triggered after initial delay (no need to modify) Default: **false**
initial_delay	The number of seconds to wait before starting the scheduler scan logic. If this parameter is omitted, the application starts the scheduler logic immediately (no need to modify) Default: **0.0**
local_name_definition	Rename file pattern to generate the local file name (rename pattern substitutions) - (no need to modify) Components can be: %0 - the source file pattern match which means the whole match to the search pattern defines the local name %1 - the match for bracket #1 %2 - the match for bracket #2 %sY - the year of the scan time %sM - the month of the scan time %sD - the day of the scan time %sh - the hour of the scan time %sm - the minute of the scan time %ss - the second of the scan time %mY - the year of the file modification date %mM - the month of the file modification date (available with option GET_FILE_MTIME = true) %mD -the day of the file modification date (available with option GET_FILE_MTIME = true) %mh - the hour of the file modification date (available with option GET_FILE_MTIME = true) %mm -the minute of the file modification date (available with option GET_FILE_MTIME = true) %ms - the second of the file modification date (available with option GET_FILE_MTIME = true) %HN – the remote hostname

	%SD - the source directory %CS – the custom string Default: **"%0"**
custom_string	The custom string for the local name replacement (no need to modify) Default:
duplicate_file_treatment	Defines the treatment of files which are already in the transfer directory (i.e. duplicates) - (no need to modify) Valid values: - skip: The file transfer is skipped (an error message is logged if source_file_treatment is 'move' or 'remove') - overwrite: The existing file is overwritten (no error is raised) - size: The existing file is overwritten if the file size of remote and the local file are different (no error is raised) Default: **skip**
source_file_treatment	Defines the treatment of source files after they are successfully transferred (no need to modify) Valid values: none, move, remove Default: **none**
move_to_dir	Directory for the source file treatment 'move'. The directory is relative to the source directory (no need to modify) Default: **archive/**
max_file_age	Maximum age in seconds for a source file. When a file is older than this value the file transfer is skipped Default: **0**
start_pattern	When a file is scanned which matches this pattern the file transfer is enabled. If this file matches also the search pattern it is transferred like a common file. The empty string starts the transfer immediately (no need to modify). The purpose of this is to start/stop the Collector using a trigger file without the need to stop the job. Default:
end_pattern	When a file is scanned which matches this pattern the file transfer is disabled until a new start pattern match is detected. If this file matches also the search pattern it is transferred like a common file. The empty string means that no end pattern is used (no need to modify). The purpose of this is to start/stop the Collector using a trigger file without the need to stop the job. Default:
file_download_delay	This defines the blocking period in seconds after each file was delivered to transfer logic. A value of zero disables blocking. Type: float64 Default: **0**
remove_if_command_fails	This parameter defines the treatment of transferred file after an unsuccessful execution of the external command (defined by the parameter external_command). If this parameter is true then the downloaded file is removed from the target directory and the transmission is treated as failure. This triggers a re-transmission of the file and a re-execution of the external command. If the parameter is false, the file is kept in the target directory and the transmission is treated as success. Default: **false**

4.4.2.5 Post download commands

As described in the Collectors section there is a possibility to set an external script in the <external_command> parameter per Collector job. The command is executed for each downloaded file and can be parametrized by providing a list of arguments (depending on the script).

The purpose of these scripts is to do one of the following and they come out of the box:

- Create symlinks for multiple Streams targets (ITE, RawTDR)
  Script: **/opt/tnf/cna/etc/script/create-symlinks.sh {**SymlinkList**} {**File**}**
  - This creates symlinks with the same filename in:
    /opt/tnf/cna/tdrs/{SymlinkDir}/{ProbeName}/in/data/{FeedType}/
    for a physical file in:
    /opt/tnf/cna/tdrs/{ProbeName}/{FeedType}/

- Call interface adaptor for Voice/SMS (Aif or IuCS)
  Script: **/opt/tnf/cna/etc/script/call-interface-adapter-voice.sh** {TargetBaseDir} **{**AdaptorDataFeedId**} {**File**}**
  - This calls an executable that acts as an interface adaptor as a command:
    {AdaptorExecutable} {File} {TargetBaseDir}/{ProbeName}/{FeedType}/
    {AdaptorDataFeedId}
  - The adapted files are created in { TargetBaseDir}/{ProbeName}/{FeedType}/
    where {ProbeName} and {FeedType} are extracted from the path of {File}

  The {AdaptorExecutable} is configurable inside the script.

- Call interface adaptor and create symlinks for Voice/SMS (Aif or IuCS)
  Script: **/opt/tnf/cna/etc/script/call-interface-adapter-voice-symlinks.sh** {TargetBaseDir} {AdaptorDataFeedId} {SymlinkList} {File}
  This is the combination of both of the scripts above. First interface adaptor is called and for each adapted file a list of symlinks is created in
  /opt/tnf/cna/tdrs/{SymlinkDir}/{ProbeName}/in/data/{FeedType}/

4.4.2.6 Symlinks explanation

This section clarifies the concept of creating multiple symlinks to one downloaded physical file.

Consider the following example. When the following mediation components are set up:

- ITE Streams application for Gn
- Raw TDR Processor Streams application for Gn
- Vantage 8.3 Gn mediation (Vedia based)
- Smartfeeds 2.0 Gn mediation (Vedia based)

Additionally, the downloaded (and eventually adapted) file can be exposed for the 3rd party access (i.e. an access point).

Then it means that there are 4 jobs (plus the 3rd party access point) that collect one Gn netro file: 2 Streams-based and 2 Vedia-based. However, Streams jobs move any processed file to a different

directory. This makes the file unavailable to other mediation components. In this case there have to be three symlinks created for Streams.

I.e. **{SymlinkList}** argument in <external_command> in the Collectors section has to be set to S**treams-ITE,Streams-RawTDR,Access-Point**.

Symlinks are created for each downloaded file in a location where Streams application (and the 3rd party access point) expects them. E.g. for netro feed:

```
/opt/tnf/can/tdrs/Streams-ITE/P1/in/data/netro/netro_20120425000019.log.gz
/opt/tnf/can/tdrs/Streams-RawTDR/P1/in/data/netro/netro_20120425000019.log.gz
/opt/tnf/can/tdrs/Access-Point/P1/in/data/netro/netro_20120425000019.log.gz
```

Both reference the same file:
`/opt/tnf/can/tdrs/P1/netro/netro_20120425000019.log.gz`

Similarly for the pagebased feed:

```
/opt/tnf/can/tdrs/Streams-ITE/P1/in/data/page/pbqoe_20120425000019.log.gz
/opt/tnf/can/tdrs/Streams-RawTDR/P1/in/data/page/pbqoe_20120425000019.log.gz
/opt/tnf/can/tdrs/Access-Point/P1/in/data/page/pbqoe_20120425000019.log.gz
```

Both reference the same file:
`/opt/tnf/can/tdrs/P1/page/pbqoe_20120425000019.log.gz`

It is expected that these directories (`/opt/tnf/can/tdrs/Streams-ITE/` and `/opt/tnf/can/tdrs/Streams-RawTDR/`) are set for individual Streams jobs in configuration files probes.csv (section 4.4.4.1) and probes-tdr.csv (section 4.4.5.1).

Please note that Collector external scripts (section 4.4.2.5) create the necessary directory hierarchy automatically. I.e. the following is created automatically within the output directory:

```
./in/urgent/
./in/data/
./in/data/{feed_type}/
./in/inputDirectories.txt
```

Important note: It is expected that different file types (such as netro feed and pagebased feed) have unique file name. I.e. prefix of different file types is different.

4.4.2.7 Voice/SMS Interface Adaptor installation

If some of the external commands call the script **call-interface-adapter-voice.sh** (or call-interface-adapter-voice-symlinks.sh) then the installation of the Voice/SMS Interface Adaptor also needs to be performed. The adaptor converts Aif and IuCS feeds from different voice probe formats to a CSV format that can be parsed by Streams.

Extract the following file from the CNA 9.1 ZIP archive: `cna/build/interface-adapter/interface-adapter-8.2.0.1.2.rpm`. Then, as user *root*, perform the RPM installation in each host where Vedia Collector was deployed already:

```
rpm -ivh interface-adapter-8.2.0.1.2.rpm
```

This will install the Interface Adapter into `/opt/tnf/apps/interface-adapter` as the user boss.

The adapter is invoked automatically from the Collector.

4.4.3 Local and centralised DB Loader

The DB Loader apps collect aggregated and TDR files, convert them through an external script (optional) and then load the records to the Hadoop DB through yet another external script. The external script configuration is performed through the `loader_{interface}.ini` files.

The DB Loader app accepts several submission time parameters which are automatically provided to the application by the `cna.sh` script depending on the loader type so there are currently no configurable settings available in `loader.ini`.

The Centralised DB Loader app has a greater amount of options and they are configured in the `cLoader_{interface}.ini` files.

4.4.3.1 How to switch

The local DB Loader app is currently enabled by default. In order to use the Centralised DB Loader app it must be disabled by setting CNA_LOADER=0 in the file `/opt/tnf/cna/cna.env` (not provided) or exported in the environment before calling the `cna.sh` script.

4.4.3.2 Configuring sudoers

Starting with CNA Framework v3 it is no longer possible to perform loading to hadoop with any user (usually boss). Instead, the `tnf` user should be used for this purpose. This is an access control issue and depends on Hadoop rather than on the client (see ENGRA-738)

In case the data cannot get loaded using the default configuration, the following may need to be configured:

1. Set `dataloaderSudoPrefix = sudo -u tnf` in `/opt/tnf/cna/etc/script/loader.ini`
2. Add the following line to the very end of the sudoers file (sudo visudo):

   ```
   boss    ALL=(ALL) NOPASSWD: ALL
   ```

3. Set the following in the sudoers file (sudo visudo)

   ```
   Defaults    !requiretty
   ```

   The above will allow sudo commands to be invoked in non-tty mode and prevent errors such as "sudo: no tty present and no askpass program specified" from appearing in the DB Loader script invocation logs (`/opt/tnf/cna/stat/loader`)

4.4.3.3 Streams instance

For the local DB Loader there is no need to specify the Streams instance; it will use the same one as the ITE or RawTDR application as per sections 4.4.4.1 and 4.4.5.1.

For the centralized DB loader, by default the Streams instance is the one defined by the environment variable $STREAMS_INSTANCE_ID. This can be set in the shell login profile.

Alternatively, it can be also specified when running the loader.sh admin command (see section 5.1) using the argument -ci {InstanceName}, .e.g.

```
/opt/tnf/cna/loader.sh -i cna_instance_blade02_cloader start
```

4.4.3.4 Local DB Loader: loader_*.ini

Settings that may need modification:

Parameter	Description
logFile	Define where the script will write its log. Default: /opt/tnf/cna/stat/$interface/loader_script/loader.log
dataLoaderLog	Log file for the CNA data loader. Default: /opt/tnf/cna/stat/$interface/loader_script/dataloader.log
dataloaderSudoPrefix	If dataloader needs to be invoked as a different user, specify the sudo prefix here. For example, while the BIS framework v2 allowed any user to invoke the loader, in v3 it is required that the loader is invoked by the *tnf* user (configurable by hadoop). In this case, the dataloaderSudoPrefix should be set to *sudo -u tnf*. In case the sudo prefix is used, please see the "Configuring sudoers" section below.
enableHadoopVoicesms	Enable/disable the loading Voice/SMS aggregations into hadoop
enableHadoopVoicesmsSc	Enable/disable the loading Voice/SMS status code aggregations into hadoop
enableHadoopUP	Enable/disable the loading of Gn/LTE/Fixed Line user plane aggregations into Hadoop
enableHadoopUPS	Enable/disable the loading of Gn/LTE/Fixed Line user plane status code aggregations into Hadoop
enableHadoopCP	Enable/disable the loading of Gn/LTE/Fixed Line control plane aggregations into Hadoop
enableHadoopCPS	Enable/disable the loading of Gn/LTE/Fixed Line control plane status code aggregations into Hadoop
enableHadoopRAN	Enable/disable the loading of RAN aggregations into Hadoop
enableHadoopSIP	Enable/disable the loading of VoLTE SIP aggregations into Hadoop
enableHadoopSIPS	Enable/disable the loading of VoLTE SIP status code aggregations into Hadoop
enableHadoopRTP	Enable/disable the loading of VoLTE RTP aggregations into Hadoop
enableHadoopGb	Enable/disable the loading of Gb aggregations into Hadoop
enableHadoopGbS	Enable/disable the loading of Gb Status aggregations into Hadoop
enableHadoopIuPS	Enable/disable the loading of IuPS aggregations into Hadoop
enableHadoopIuPSS	Enable/disable the loading of IuPS Status aggregations into Hadoop
enableHadoopS1MME	Enable/disable the loading of S1MME aggregations into Hadoop
enableHadoopS1MMES	Enable/disable the loading of S1MME Status aggregations into Hadoop
enableHadoopS6a	Enable/disable the loading of S6a aggregations into Hadoop
enableHadoopS6aS	Enable/disable the loading of S6a Status aggregations into Hadoop
enableHadoopGx	Enable/disable the loading of Gx aggregations into Hadoop
enableHadoopGxS	Enable/disable the loading of Gx Status aggregations into Hadoop
enableHadoopGy	Enable/disable the loading of Gy aggregations into Hadoop

enableHadoopGyS	Enable/disable the loading of Gy Status aggregations into Hadoop

The mapping section between the aggregation levels and table names is used in case of ITE applications (not RawTDR); the default mappings can be left as is.

Other settings in this file would usually not require changes.

4.4.3.5 Centralized DB Loader: cLoader_*.ini

These files are the main configuration files for the Centralised DB Loader: there is one file per interface, and by default the options *scriptPath* and *scriptArgs* are set to use the same DB loading script and configuration files as the local DB Loader (covered in the previous sections).

The most important section in the ini files is the `[remote]` section: there must be one for each host to collect from with the following parameters:

Parameter	Description
host	Mandatory. The remote host name or the IP address.
user	The username used to connect to the remote host (default **boss**).
password	The password for the username on the remote host (default **boss**).
scanDirectory	The root output directory of the application (default **/opt/tnf/cna/loaderDir**); the actual collection path is built as $scanDirectory/$interface/$subdirectory(/$segment)/load, where the segment is missing in non-segmented aggregation levels and the $subdirectory is extracted from the list in the [directories] section. **Note:** When using the FTP protocol this is a relative path from the user's home unless it starts with double slash, while when using SFTP it should be absolute.

The `[directories]` section should not be modified.
The `[parameters]` section contains some options that may need customising on site, in particular:

Parameter	Description
protocol	Specifies the protocol to use for the connection (**ftp**/sftp). Note that ftp has much better performance than sftp.
databaseSegments	Specifies the number of subscriber segments configured in the database. It is assumed that all hosts are aligned to the same number (default **10**).
minFreeDiskSpace	Specifies the amount of free disk space that is required to scan remote directories and process the remote files (default **1GB**).
numberOfParallelRunningRemoteDirectoryScanners	Specifies the number of parallel running remote directory scanners (default **1**). Should be overwritten to max(#CPU_cores,11). This will be improved in the future!
numberOfParallelRunningBundleProcessors	Specifies the number of parallel running bundle processors (default **1**). Should be overwritten to max(#CPU_cores,11). This will be improved in the future!
temporaryDirectoryForMergedFiles	Specifies the directory in which the files that are merged are created.

	This directory should contain only the in-progress files during the merge process. An external housekeeping should observe this directory and remove files that are too old and do not change for a long time. A relative directory is relative to the data directory. Default **/opt/tnf/cna/stat/$interface/merging**.
targetDirectoryForMergedFiles	Specifies the directory where the bundle files are moved when finalised. Default **/opt/tnf/cna/stat/$interface/merged**.
symbolicLinkDirectories	Specifies a comma separated list of directories where a symbolic link will be created when each bundle file is finalised. If non empty, the loading script is called on the symbolic link in the first directory. Default **empty**.
scanPeriod	Specifies the time in seconds between two scan cycles. The scan period starts after processing, so the application idles for this time. Default **60** seconds.
bundleMaxSize	Specifies the maximum summed size of files per bundle. If a single file is larger than the bundleMaxSize, it is processed as bundle with this one file (default **128MB**).
moveToDirectory	Processed source files are either deleted if moveToDirectory is omitted or moved to the specified directory. The path is relative to the scanDirectory path specified in the remote section(s). Default **empty**.

Other settings in this file would usually not require changes.

4.4.4 ITE

The ITE app prepares 5 minute aggregated datasets to be loaded into the database.

4.4.4.1 probes.csv

This file lists all probes collected by the blade for processing by the ITE apps. The default content of the file is a guide on how to configure it. For each probe you must specify on one line, pipe separated, the following fields:

Parameter	Description
instance	The name of the Streams instance that the probe will be processed by
type	One of *gn, lte*, *voice-a*, *voice-iucs*, *volte*, *ran*, *fixedline, gb, iups, s1mme_s6a, gx_gy* (note that it must be the same for all lines referring to the same instance)
name	The probe name as it should appear in the output records
directory	A directory path (see also the section with the local Collector target directory settings) to collect input files. Inside which a directory must exist with the same name as the probe, and inside this directory the file in/inputDirectories.txt, containing the input directory names, one per line. Default is **/opt/tnf/cna/tdrs/Streams-ITE/**

Please note that you cannot configure more probes per interface than specified in the NumAggProbes option (see section 0).

As indicated in the description, an input directory for a probe has to have a specific directory hierarchy. The default hierarchy starts at the (default) path `/opt/tnf/cna/tdrs/Streams-ITE/`.

In the above example path:

- the input TDRs are expected to appear in `/opt/tnf/cna/tdrs/Streams-ITE/{probe_name}/in/data/{feed_type}/`, where `{probe_name}` is the name of the probe as per the table above and `{feed_type}` corresponds to different file types (e.g. netro, pagebased, etc).
- Any data that appear in `/opt/tnf/cna/tdrs/Streams-ITE/{probe_name}/in/urgent/` will be processed with a higher priority (required during recovery).
- A successfully processed file is moved to the `/opt/tnf/cna/tdrs/Streams-ITE/{probe_name}/in/archive/` directory.

Please also note that when the Collector is deployed then input files are actually symlinks, because one input file might be collected by multiple mediation components. When a symlink is moved to the `archive/` directory then the original file is still available for the other mediation component.

4.4.4.2 consts_gn.ini, consts_lte.ini

These two files have similar contents, but each is used by the specific interface; among the many options, the following are those of greater interest (defaults in **bold**):

Parameter	Description
netroFileMask	The file mask for netro input files*
pagebasedFileMask	The file mask for page based input files*
enableUPSubscriberAggr	Enable/disable subscriber (level 3) aggregation for User Plane. When set to false, the subscriber aggregated files should not be created for User Plane
enableUPCellsAggr	Enable/disable cell (level 2a) aggregation. When set to false, the cell aggregated files should not be created for User Plane
enableCPSubscriberAggr	Enable/disable subscriber (level 3) aggregation for Control Plane. When set to false, the subscriber aggregated files should not be created for Control Plane
enableCPCellsAggr	Enable/disable cell (level 2a) aggregation for Control Plane. When set to false, the cell aggregated files should not be created for Control Plane
enableImeiAttribute	Enable/Disable the setting of the IMEI attribute. Default value is **true**. If set to false, the IMEI attribute should always be blank
enableMsisdn	Enable/Disable MSISDN field (**true** - populate MSISDN from feed, false - set MSISDN to empty)
loadUnknownSubscribers	Enable/Disable the filter for unknown IMSIs in level 3 tables (true – load data for unknown IMSIs to DB, **false** – filter out unknown IMSIs)
filterCPsuccessCodes	Enable/Disable control plane status code filters (**true** – filter out success codes, false – load all codes to DB)
CPsuccessCodes	The comma-separated list of success codes for the interface
ignorePageBased	Enable/Disable page based feed (true – ignore PB TDRs, **false** – process PB TDRs)
operatorPrefix	Specify a list of operator prefixes; if set to a non-null value, only IMSIs starting with a value in the comma-separated list will be processed
subscriberOperatorPrefix	Specify a list of operator prefixes; if set to a non-null value, only IMSIs starting with a value in the comma-separated list will be processed at subscriber level
loaderDirectory	Specify where the output should be written (**/opt/tnf/cna/loaderDir**)
bytesUpMax	Specify the maximum number of bytes to allow in upload in each TDR; higher values will cause the TDR to be discarded (**200000000**)
bytesDownMax	Specify the maximum number of bytes to allow in download in each TDR; higher values will cause the TDR to be discarded (**400000000**)

parserType	In case your installation supports multiple adapters, you can select the adapter for this application. Available adapters depend on the interface. Adapters are enumerated (1, 2, 3…). The default value is 1, and selects the TNF probe adapters.
lte.status.code.min.value	The minimum allowed Statuscode value for LTE Controlplane. The default is 16.
enableDNSSubscriberAggr	Enable/Disable DNS subscriber (level 3) aggregations. When set to false the aggregated files are not created
DnsApplication	Name of DNS application. Default value: **dns**
dns.host	Enable/Disable lookup of DNS host. When set to lookup the Host dimension will be lookup from dns.csv file. Default value: **lookup**

* The file masks must contain a timestamp in the format YYYYMMDD[.]hhmmss (where the square brackets indicate that the dot is optional) in order for the file ordering to correctly sort the input feed by file timestamp.

4.4.4.3 consts_voice.ini

This file is used by the Voice&SMS interface; among the many options, the following are those of greater interest (defaults in **bold**):

Parameter	Description
aFileMask	The file mask for Aif (voice-a) input files*
iucsFileMask	The file mask for IuCS (voice-iucs) input files*
appSplitterEnabled	Enable/Disable aggregation segmentation (**true** – split among multiple contexts, false – do not split)
numContextsVSMS	Specify how many contexts to split Voice&SMS data in when appSplitterEnabled=true (**4**)
numContextsVSMSStatus	Specify how many contexts to split Voice&SMS Status data in when appSplitterEnabled=true (**1**)
enableSubscriberAggr	Enable/disable subscriber (level 3) aggregation. When set to false, the subscriber aggregated files should not be created
enableCellAggr	Enable/disable cell (level 2a) aggregation. When set to false, the cell aggregated files should not be created
enableImeiAttribute	Enable/Disable the setting of the IMEI attribute. Default value is **true**. If set to false, the IMEI attribute should always be blank
loadUnknownSubscribers	Enable/Disable the filter for unknown IMSIs in level 3 tables (true – load data for unknown IMSIs to DB, **false** – filter out unknown IMSIs)
operatorPrefix	Specify a list of operator prefixes; if set to a non-null value, only IMSIs starting with a value in the comma-separated list will be processed
loaderDirectory	Specify where the output should be written (**/opt/tnf/cna/loaderDir**)
voicesms.proprietary.cause	Set to 0 if the standard GSM/UMTS termination causes are used in the feed, set to 1 if the cause is proprietary. **Default value is 0**. Set to 1 for Singtel NDC feed. May be set to other values in the future to support non-standard feeds
voicesms.adapter.calculated.rat	Set to 0 if the RAT is to be determined based on the collected file type (2G for Aif, 3G for IucS), Set to 1 if the RAT is being calculated by the interface adapter. **Default is 0**
voicesms.call.duration.get.from.feed **voicesms.setup.duration.get.from.feed**	Set these to 1 if the durations are provided in the incoming feed (either in the raw feed or calculated by the adapter). Set to 0 if the durations should be calculated by Vedia based on timestamps. **Default: 0**
voicesms.sms.duration.get.from.feed	

parserType	In case your installation supports multiple adapters, you can select the adapter for this application. Available adapters depend on the interface. Adapters are enumerated (1, 2, 3…). The default value is 1, and selects the TNF probe adapters.

* The file masks must contain a timestamp in the format YYYYMMDD[.]hhmmss (where the square brackets indicate that the dot is optional) in order for the file ordering to correctly sort the input feed by file timestamp.

4.4.4.4 consts_ran.ini

This file is used by the RAN interface; among the many options, the following are those of greater interest (defaults in **bold**):

Parameter	Description
fileMask	The file mask for input files*
enableSubscriberAggr	Enable/disable subscriber (level 3) aggregation. When set to false, the subscriber aggregated files should not be created
enableDeviceAggr	Enable/disable device (level 2) aggregation. When set to false, the device aggregated files should not be created
enableCellAggr	Enable/disable cell (level 2a) aggregation. When set to false, the cell aggregated files should not be created
enableImeiAttribute	Enable/Disable the setting of the IMEI attribute. Default value is **true**. If set to false, the IMEI attribute should always be blank
internalCellId	If set to true, the cell lookup will be performed based on the internal Cell ID field, otherwise it will be performed based on the full cell key (MCC:MNC:LAC:CellID)
internalNodeId	If set to true, the node lookups (BSC, RNC, MME, BTS, NodeB, eNodeB) will be performed based on the internal node Ids, otherwise they will be performed on the nodes IP address.
loadUnknownSubscribers	Enable/Disable the filter for unknown IMSIs in level 3 tables (true – load data for unknown IMSIs to DB, **false** – filter out unknown IMSIs)
loadUnknownDevices	Enable/Disable the filter for unknown devices in level 2 tables (true – load data for unknown devices to DB, **false** – filter out unknown devices)
operatorPrefix	Specify a list of operator prefixes; if set to a non-null value, only IMSIs starting with a value in the comma-separated list will be processed
subscriberOperatorPrefix	Specify a list of operator prefixes; if set to a non-null value, only IMSIs starting with a value in the comma-separated list will be processed at subscriber level
loaderDirectory	Specify where the output should be written (**/opt/tnf/cna/loaderDir**)
parserType	In case your installation supports multiple adapters, you can select the adapter for this application. Available adapters depend on the interface. Adapters are enumerated (1, 2, 3…). The default value is 1, and selects the TNF probe adapters.

* The file masks must contain a timestamp in the format YYYYMMDD[.]hhmmss (where the square brackets indicate that the dot is optional) in order for the file ordering to correctly sort the input feed by file timestamp.

4.4.4.5 consts_fixedline.ini

This file is used by the FixedLine interface; among the many options, the following are those of greater interest (defaults in **bold**):

Parameter	Description

netroFileMask	The file mask for netro input files*
pagebasedFileMask	The file mask for page based input files*
enableUPSubscriberAggr	Enable/disable subscriber (level 3) aggregation for User Plane. When set to false, the subscriber aggregated files should not be created for User Plane
enableCPSubscriberAggr	Enable/disable subscriber (level 3) aggregation for Control Plane. When set to false, the subscriber aggregated files should not be created for Control Plane
loadUnknownSubscribers	Enable/Disable the filter for unknown CSIDs in level 3 tables (true – load data for unknown CSIDs to DB, **false** – filter out unknown CSIDs)
filterCPsuccessCodes	Enable/Disable control plane status code filters (**true** – filter out success codes, false – load all codes to DB)
CPsuccessCodes	The comma-separated list of success codes for the interface
ignorePageBased	Enable/Disable page based feed (true – ignore PB TDRs, **false** – process PB TDRs)
subscriberOperatorPrefix	Specify a list of operator prefixes; if set to a non-null value, only IMSIs starting with a value in the comma-separated list will be processed at subscriber level
loaderDirectory	Specify where the output should be written (**/opt/tnf/cna/loaderDir**)
bytesUpMax	Specify the maximum number of bytes to allow in upload in each TDR; higher values will cause the TDR to be discarded (**200000000**)
bytesDownMax	Specify the maximum number of bytes to allow in download in each TDR; higher values will cause the TDR to be discarded (**400000000**)
parserType	In case your installation supports multiple adapters, you can select the adapter for this application. Available adapters depend on the interface. Adapters are enumerated (1, 2, 3…). The default value is 1, and selects the TNF probe adapters.

* The file masks must contain a timestamp in the format YYYYMMDD[.]hhmmss (where the square brackets indicate that the dot is optional) in order for the file ordering to correctly sort the input feed by file timestamp.

4.4.4.6 consts_gb.ini

This file is used by the Gb interface; among the many options, the following are those of greater interest (defaults in **bold**):

Parameter	Description
gbFileMask	The file mask for input files. The default value is **'gb_([0-9]{14})\\.log\\.gz'** which matches Gb files in TMA. For Interface Adapter feeds, the mask should be modified accordingly (note that the adapter generates files with .csv.gz extensions)
enableSubscriberAggr	Enable/disable subscriber (level 3) aggregation. When set to false, the subscriber aggregated files should not be created
enableDeviceAggr	Enable/disable device (level 2) aggregation. When set to false, the device aggregated files should not be created
enableCellAggr	Enable/disable cell (level 2a) aggregation. When set to false, the cell aggregated files should not be created
enableImeiAttribute	Enable/Disable the setting of the IMEI attribute. Default value is **true**. If set to false, the IMEI attribute should always be blank
loadUnknownSubscribers	Enable/Disable the filter for unknown IMSIs in level 3 tables (true – load data for unknown IMSIs to DB, **false** – filter out unknown IMSIs)
operatorPrefix	Specify a list of operator prefixes; if set to a non-null value, only IMSIs starting with a value in the comma-separated list will be processed
subscriberOperatorPrefix	Specify a list of operator prefixes; if set to a non-null value, only IMSIs starting with a value in the comma-separated list will be processed at subscriber level
loaderDirectory	Specify where the output should be written (**/opt/tnf/cna/loaderDir**)
parserType	There are two types of supported Gb files, with the following parserType:

	1 – Netscout feed (used in TMA) 2 – Feeds generated by the Interface Adapter The parserType setting should be configured accordingly, based on the feed that is being processed. Default value is **1**. There are several differences between these feeds, the most important ones that feeds of parserType 1 have "|" as the delimiter and the cause fields missing, while feeds of parserType 2 have "," as the delimiter and contain the cause fields.
causesAvailable	Define whether GMM and SM causes are available in the feed, default value is **true**. Only has an impact on feeds generated by the Interface Adapter (parserType=2).

4.4.4.7 consts_iups.ini

This file is used by the IuPS interface; among the many options, the following are those of greater interest (defaults in **bold**):

Parameter	Description
iupsFileMask	The file mask for input files. The default value is **'iups_([0-9]{14})\\.log\\.gz'** which matches IuPS files in TMA. For Interface Adapter feeds, the mask should be modified accordingly (note that the adapter generates files with .csv.gz extensions)
enableSubscriberAggr	Enable/disable subscriber (level 3) aggregation. When set to false, the subscriber aggregated files should not be created
enableDeviceAggr	Enable/disable device (level 2) aggregation. When set to false, the device aggregated files should not be created
enableCellAggr	Enable/disable cell (level 2a) aggregation. When set to false, the cell aggregated files should not be created
enableImeiAttribute	Enable/Disable the setting of the IMEI attribute. Default value is **true**. If set to false, the IMEI attribute should always be blank
enableMsisdn	Enable/Disable the setting of the MSISDN attribute. Default value is **true**. If set to false, the MSISDN attribute should always be blank
loadUnknownSubscribers	Enable/Disable the filter for unknown IMSIs in level 3 tables (true – load data for unknown IMSIs to DB, **false** – filter out unknown IMSIs)
operatorPrefix	Specify a list of operator prefixes; if set to a non-null value, only IMSIs starting with a value in the comma-separated list will be processed
subscriberOperatorPrefix	Specify a list of operator prefixes; if set to a non-null value, only IMSIs starting with a value in the comma-separated list will be processed at subscriber level
filterSuccessCodes	Specify a list of status codes to filter out from the status aggregation: by default all success codes are filtered out, set to empty to allow all status codes.
loaderDirectory	Specify where the output should be written (**/opt/tnf/cna/loaderDir**)
parserType	There are two types of supported IuPS files, with the following parserType: 1 – Netscout feed (used in TMA) 2 – Feeds generated by the Interface Adapter The parserType setting should be configured accordingly, based on the feed that is being processed. Default value is **1**. There are several differences between these feeds, the most important ones that feeds of parserType 1 have "|" as the delimiter and the cause fields missing, while feeds of parserType 2 have "," as the delimiter and contain the cause fields.

4.4.4.8 consts_s1mme_s6a.ini

This file is used by the S1MMe and S6a interface; among the many options, the following are those of greater interest (defaults in **bold**):

Parameter	Description
netroFileMask	The file mask for netro input files*
enableS1MMeSubscriberAggr	Enable/disable subscriber (level 3) aggregation for S1MMe. When set to false, the subscriber aggregated files should not be created for S1MMe.
enableS1MMeDeviceAggr	Enable/disable device (level 2) aggregation for S1MMe. When set to false, the device aggregated files should not be created for S1MMe.
enableS1MMeCellAggr	Enable/disable cell (level 2a) aggregation for S1MMe. When set to false, the cell aggregated files should not be created for S1MMe.
enableS6aSubscriberAggr	Enable/disable subscriber (level 3) aggregation for S6A. When set to false, the subscriber aggregated files should not be created for S6A.
loadUnknownSubscribers	Enable/Disable the filter for unknown CSIDs in level 3 tables (true – load data for unknown CSIDs to DB, **false** – filter out unknown CSIDs)
subscriberOperatorPrefix	Specify a list of operator prefixes; if set to a non-null value, only IMSIs starting with a value in the comma-separated list will be processed at subscriber level
enableImeiAttribute	Enable/Disable the setting of the IMEI attribute. Default value is **true**. If set to false, the IMEI attribute should always be blank
enableMsisdn	Enable/Disable MSISDN field (**true** - populate MSISDN from feed, false - set MSISDN to empty)
loaderDirectory	Specify where the output should be written (**/opt/tnf/cna/loaderDir**)
parserType	In case your installation supports multiple adapters, you can select the adapter for this application. Available adapters depend on the interface. Adapters are enumerated (1, 2, 3…). The default value is 1, and selects the TNF probe adapters.

* The file masks must contain a timestamp in the format YYYYMMDD[.]hhmmss (where the square brackets indicate that the dot is optional) in order for the file ordering to correctly sort the input feed by file timestamp.

4.4.4.9 consts_gx_gy.ini

This file is used by the Gx and Gy interface; among the many options, the following are those of greater interest (defaults in **bold**):

Parameter	Description
netroFileMask	The file mask for netro input files*
enableGxSubscriberAggr	Enable/disable subscriber (level 3) aggregation for Gx. When set to false, the subscriber aggregated files should not be created for Gx.
enableGySubscriberAggr	Enable/disable subscriber (level 3) aggregation for Gy. When set to false, the subscriber aggregated files should not be created for Gy.
loadUnknownSubscribers	Enable/Disable the filter for unknown IMSIs in level 3 tables (true – load data for unknown IMSIs to DB, **false** – filter out unknown IMSIs)
subscriberOperatorPrefix	Specify a list of operator prefixes; if set to a non-null value, only IMSIs starting with a value in the comma-separated list will be processed at subscriber level
enableMsisdn	Enable/Disable MSISDN field (**true** - populate MSISDN from feed, false - set MSISDN to empty)
loaderDirectory	Specify where the output should be written (**/opt/tnf/cna/loaderDir**)
parserType	In case your installation supports multiple adapters, you can select the adapter for this application. Available adapters depend on the interface. Adapters are enumerated (1, 2, 3…). The default value is 1, and selects the TNF probe adapters.

* The file masks must contain a timestamp in the format YYYYMMDD[.]hhmmss (where the square brackets indicate that the dot is optional) in order for the file ordering to correctly sort the input feed by file timestamp.

4.4.4.10 consts_s13_sgs.ini

This file is used by the S13 and SGs interface; among the many options, the following are those of greater interest (defaults in **bold**):

Parameter	Description
netroFileMask	The file mask for netro input files*
enableS13SubscriberAggr	Enable/disable subscriber (level 3) aggregation for S13. When set to false, the subscriber aggregated files should not be created for S13.
enableSGsSubscriberAggr	Enable/disable subscriber (level 3) aggregation for SGs. When set to false, the subscriber aggregated files should not be created for SGs.
loadUnknownSubscribers	Enable/Disable the filter for unknown CSIDs in level 3 tables (true – load data for unknown CSIDs to DB, **false** – filter out unknown CSIDs)
subscriberOperatorPrefix	Specify a list of operator prefixes; if set to a non-null value, only IMSIs starting with a value in the comma-separated list will be processed at subscriber level
enableMsisdn	Enable/Disable MSISDN field (**true** - populate MSISDN from feed, false - set MSISDN to empty)
loaderDirectory	Specify where the output should be written (**/opt/tnf/cna/loaderDir**)
parserType	In case your installation supports multiple adapters, you can select the adapter for this application. Available adapters depend on the interface. Adapters are enumerated (1, 2, 3…). The default value is 1, and selects the TNF probe adapters.

* The file masks must contain a timestamp in the format YYYYMMDD[.]hhmmss (where the square brackets indicate that the dot is optional) in order for the file ordering to correctly sort the input feed by file timestamp.

4.4.5 RawTDR

The RawTDR app prepares raw TDRs for loading into the database.

4.4.5.1 probes-tdr.csv

This file lists all probes collected by the blade for processing by the RawTDR app. The default content of the file is a guide on how to configure it. For each probe you must specify on one line, pipe separated, the following fields:

Parameter	Description
instance	The name of the Streams instance that the probe will be collected by
type	One of: *gn*, *lte, voice* or *fixedline* (note that it must be the same for all lines referring to the same instance)
name	The probe name as it should appear in the output records
directory	A directory path (see also the section with the local Collector target directory settings) to collect input files. Inside which a directory must exist with the same name as the probe, and inside this directory the file in/inputDirectories.txt, containing the input directory names, one per line. Default is **/opt/tnf/cna/tdrs/Streams-RawTDR/**

The input directory contains sub-directories as per the probes.csv section related to the ITE app (see 4.4.4.1). The only exception is that instead of `Streams-ITE` subdirectory the RawTDR application uses `Streams-RawTDR` as per the default directory in the parameter list above.

4.4.5.2 *rawtdr_*.ini*

The submission time parameters are:

Parameter	Description
fileMask	The file mask for collected files. Default is '**^.+\\.gz$**'. Files that don't match the mask will not be processed and will be placed into the 'invalid' directory (in/invalid)
waitMin	Define how long (in minutes) should the 1-hour partitions wait untouched until being sent for loading to hadoop. Default **60** minutes
maxRecordsGn	Max number of records that can be stored in 1h partitioned files (for Gn) before they get closed for loading. If set to 0 the file will grow indefinitely and will close after the *waitMin* time
maxRecordsLTE	Max number of records that can be stored in 1h partitioned files (for LTE) before they get closed for loading. If set to 0 the file will grow indefinitely and will close after the *waitMin* time
maxRecordsVoice	Max number of records that can be stored in 1h partitioned files (for Voice) before they get closed for loading. If set to 0 the file will grow indefinitely and will close after the *waitMin* time
ignorePageBased	Define whether page based records (x013) should be ignored (applicable for gn and lte) 1: ignore page based, **0**: don't ignore
loaderDirectory	The absolute path to the directory where rawtdr will store output files for processing by the loader (**/opt/tnf/cna/loaderDir/raw/**)

4.4.6 Provisioning Files

Provisioning files are regularly updated from the master application server whenever changes are detected. These updates are performed automatically in the background by the cna.sh health monitor on all Mediation blades so there is no need to modify the provisioning files directly on the Mediation blades. However, the formats of the provisioning files are listed here for reference.

The user also needs to be aware that **keys** in the provisioned files need to be **unique**. Otherwise the Lookup Manager fails with an error message as per the section 5.6.3.

4.4.6.1 *cells.csv*

The cells provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
mcc_mnc_lac_ci	MCC:MNC:LAC:CI of the cell
name	The unique cell name
internal_id	The internal cell id – being a secondary key, it must be unique in each row

4.4.6.2 *mcc_mnc.csv*

The country/network provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
mcc	The mobile country code
mnc	The mobile network code
operator type	One of 0 (international), 1 (home) or 2 (probe)
country	The name of the country
prefix	The country's international prefix
network	The name of the network provider

4.4.6.3 collector.csv

The collector provisioning file defines the MCC/MNC and the type of the probe (home, roamer). This information is required for roaming calculations.

Parameter	Description
probe_name	Name of the probe, as used in Streams probes.csv files (e.g P1, etc)
probe_mccmnc	The local MCCMNC the probe is collecting information for (e.g. 27201, etc)
probe_traffic_type	The attribute specifies type of traffic collected by probe. Possible values are: 0: HOME (e.g. for S5) 1: ROAMER (e.g. for S8)

4.4.6.4 bnumber.csv

The B-number provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
prefix	The B-number prefix (prefixed by “P” if international)
label	The country name

4.4.6.5 status.csv

The Voice&SMS status code provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
cause_code	The cause code, in the form record_type:operation_type:code, e.g. “mo_call:bssmap:0”
label	The cause label
result	Success or Failure
operation_type	Same as in cause_code, but in printable form, e.g. “BSSMAP”
record_type	Same as in cause_code, but in printable form, e.g. “MO Call”

4.4.6.6 gn_qoscategory.csv

The Gn QoS Category provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
arpthp	The requested allocation and retention priority (ARP) and traffic handling priority (THP), colon separated, e.g. “1:1”
qos_category	The integer value of QoS category
label	The label corresponding to the selected QoS category

4.4.6.7 lte_qoscategory.csv

The LTE QoS Category provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
qos_category	The integer value of QoS category
label	The label corresponding to the selected QoS category
qci	The integer value of QoS QCI
qos_priority	The integer value of QoS priority

4.4.6.8 fixedline_qos.csv

The FixedLine QoS Category provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
qos_priority	The integer value of QoS priority
label	The label corresponding to the selected QoS priority

4.4.6.9 sgsn.csv

The SGSN provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
ip_address	The SGSN IP address
group	The SGSN group

4.4.6.10 ggsn.csv

The GGSN provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
ip_address	The GGSN IP address
group	The GGSN group

4.4.6.11 sgw.csv

The SGW provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
ip_address	The SGW IP address
group	The SGW group

4.4.6.12 bsc.csv

The BSC provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
ip_address	The BSC IP address
group	The BSC group
internal_id	The internal BSC id – being a secondary key, it must be unique in each row

4.4.6.13 rnc.csv

The RNC provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
ip_address	The RNC IP address
group	The RNC group
internal_id	The internal RNC id – being a secondary key, it must be unique in each row

4.4.6.14 mme.csv

The MME provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
ip_address	The MME IP address
group	The MME group
internal_id	The internal MME id – being a secondary key, it must be unique in each row

4.4.6.15 bts.csv

The BTS provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
ip_address	The BTS IP address

group	The BTS group
internal_id	The internal BTS id – being a secondary key, it must be unique in each row

4.4.6.16 nodeb.csv

The NodeB provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
ip_address	The NodeB IP address
group	The NodeB group
internal_id	The internal NodeB id – being a secondary key, it must be unique in each row

4.4.6.17 enodeb.csv

The eNodeB provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
ip_address	The eNodeB IP address
group	The eNodeB group
internal_id	The internal eNodeB id – being a secondary key, it must be unique in each row

4.4.6.18 applications.csv

The Applications provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
application	The application name as found in the TDRs
label	The application name as it should appear in the aggregated output

4.4.6.19 sip_status_codes.csv

The VoLTE SIP status codes provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
status_code	The status code in the format VOLTE:Transaction:Response, e.g. VOLTE:NOTIFY:Timeout or VOLTE:CANCEL:604
label	The code label
result	Success or Failure

4.4.6.20 csid.csv

The FixedLine CSID provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
csid	The Call Station ID
dslam	The Digital Subscriber Line Access Multiplexer
cpe_dev	Customer Premise Equipment Device

4.4.6.21 tier0.csv

The FixedLine tier zero provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
ip_addr	The tier 0 IP address
group	The tier 0 group

4.4.6.22 tier1.csv

The FixedLine tier one provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
ip_addr	The tier 1 IP address
group	The tier 1 group

4.4.6.23 fixedline_transaction_type.csv

The FixedLine transaction type provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
type	The transaction type
label	The transaction type label (Session Create, etc)

4.4.6.24 fixedline_transaction_status.csv

The FixedLine transaction status provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
status	The transaction status as 3 colon separated values, ex: FIXEDLINE:2:3
label	The cause code label
result	Success or Failure

4.4.6.25 gb_status.csv

The Gb status provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
status	The type of cause and cause value separated by an underscore, e.g. GMM2G_2
label	The cause code label
result	Success or Failure

4.4.6.26 iups_status.csv

The IuPS status provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
status	The type of cause and cause value separated by an underscore, e.g. GMM3G_2
label	The cause code label
result	Success or Failure

4.4.6.27 sx_status.csv

The S1MME and S6a transaction statuses provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
lte_cause_code	The cause code key, which contains a colon separated list of the message type, procedure type and cause code value (example: S1MME:0:1:1000)
label	The cause code label (example: “unspecified (000)”)
result	Success or Failure
type	Cause Type (4G LTE Session Access and Mobility Metrics (S1-MME))
message_type	Message Type (S1AP, etc.)
transaction_type	Transaction type (Handover Resource Allocation, etc.)

4.4.6.28 sx_transactiontype.csv

The S1MME and S6a transaction types provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
lte_transaction_type	The transaction type key, which contains a colon separated list of the message type and procedure type (example: S1MME:0:1)
label	The transaction type label (example: Handover Preparation)
transaction_group	The transaction group (example: 4G LTE Session Access and Mobility Metrics (S1-MME))

message_type	Message Type (S1AP, etc.)

4.4.6.29 dra.csv

The DRA IP addresses and DRA names provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
dra_ip	IP Address (v4 or v6)
dra_name	DRA name

4.4.6.30 hss.csv

The HSS IP addresses and HSS names provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
hss_ip	IP Address (v4 or v6)
hss_name	HSS name

4.4.6.31 s6a_dra_position.csv

The DRA position values and DRA position names provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
dra_position	DRA position
dra_position_name	DRA position name

4.4.6.32 pcrf.csv

The PCRF IP addresses and PCRF names provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
pcrf_ip	IP Address (v4 or v6)
pcrf_name	PCRF name

4.4.6.33 ocs.csv

The OCS IP addresses and OCS names provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
ocs_ip	IP Address (v4 or v6)

ocs_name	OCS name

4.4.6.34 gxgy_transactiontype.csv

The Gx and Gy transaction types provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
gx_gy_transaction_type	The transaction type key, which contains a colon separated list of the message type and command code (example: Gx:272)
label	The transaction type label (example: Credit Control (Gx))
transaction_group	The transaction group (example: Mobile Policy Charging and Rules (Gx))
message_type	Message Type (example: Diameter)

4.4.6.35 gxgy_transaction_status.csv

The Gx and Gy transaction statuses provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
gx_gy_cause_code	The cause code key, which contains a colon separated list of the message type, command code and transaction type value (example: Gx:272:1001)
label	The cause code label (example: DIAMETER_MULTI_ROUND_AUTH (Gx:272:1001))
result	Success or Failure
type	Cause Type (Mobile Policy Charging and Rules (Gx))
message_type	Message Type (Diameter, etc.)
transaction_type	Transaction type (Credit Control, etc.)

4.4.6.36 vlr.csv

The VLR IP addresses or name and VLR group names provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
vlr_ip	IP Address (v4 or v6) or name
vlr_name	VLR group name

4.4.6.37 s13_sgs_transactiontype.csv

The S13 and SGs transaction types provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
s13_sgs_transaction_type	The transaction type key, which contains a colon separated list of the message type and command code (example: S13:324)
label	The transaction type label (example ME Identity Check)
transaction_group	The transaction group (example: 4G LTE User Equipment Identity (S13))

message_type	Message Type (example: Diameter)

4.4.6.38 s13_sgs_transaction_status.csv

The S13 and SGs transaction statuses provisioning file: it must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
s13_sgs_cause_code	The cause code key, which contains a colon separated list of the message type, command code and transaction type value (example: S13:324:1001)
label	The cause code label (example: DIAMETER_MULTI_ROUND_AUTH (1001))
result	Success, Failure or Informational
type	Cause Type (4G LTE User Equipment Identity (S13))
message_type	Message Type (Diameter, etc.)
transaction_type	Transaction type (ME Identity Check, etc.)

4.4.6.39 dns.csv

This file contains mappings between DNS IP addresses and DNS Host names. This lookup must be enabled by default, see **dns.host** configuration setting. Multiple IP addresses can be mapped to a single name. Assign not found dns hosts to "Uncategorised". It must contain the following fields, comma-separated (comment lines start with #).

Parameter	Description
dns_ip	IP Address (v4 or v6) or name
dns_name	DNS group name

4.4.7 Provisioning Update Configuration

Configuring the provisioning update feature consists of configuring the remote and local locations for provisioning files, which is configured in the cna.sh script, and of configuring per-dimension details which is configured in the provisioning.cfg file, located in the same dir with cna.sh.

4.4.7.1 Provisioning configuration in cna.sh

The following provisioning update settings can be configured in cna.sh

Parameter	Description
PROVISIONING_CENTRAL_SERVER	IP Address or hostname of the central server where the auto exported provisioning files are stored
PROVISIONING_USERNAME	User on the remote server (**tnf**)
PROVISIONING_REMOTE_DIR	Directory on the remote server where the exported provisioning files are stored (**/opt/tnf/apps/bis-main-var/provisioning/**)
PROVISIONING_LOCAL_DIR	Directory where the files will be downloaded to. This should be the configuration dir where the reprovision command expects the files (**/opt/tnf/cna/cfg/**)

PROVISIONING_CFG_FILE	Local file where the dimensions are mapped to remote filenames and the timestamps of downloaded files are stored (**/opt/tnf/cna/cfg/provisioning.cfg**)

4.4.7.2 Provisioning.cfg

The provisioning configuration file is a CSV based file which stores the mappings between dimensions and corresponding local and remote files as well as the timestamps of the last downloaded file for the dimension. Each entry consists of the following fields:

- Dimension (e.g. cell, apn, etc), used as a dimension key for the script onlt, can be set to any meaningful value (doesn't have to exactly match dimension names in CNA)
- Local filename: The name of the file as expected by the Lookup Manager (e.g. cells.csv, etc)
- Remote filename: The name of the file as provided by the central server auto export feature
- Timestamp: The timestamp in seconds since epoch of the most recent downloaded file for the dimension. This field is populated by the script. For new dimensions the value should be set to 0.

The default provisioning.cfg is built into the Lookup Manager RPM and installed into the CNA root dir (/opt/tnf/cna/).

Example content:

```
# Dimension,LocalFilename,RemoteFilename,Timestamp
applications,applications.csv,APPLICATIONS_auto_export.csv,0
bnumber,bnumber.csv,BNUMBER_auto_export.csv,0
cell,cells.csv,CELL_auto_export.csv,0
mccmnc,mcc_mnc.csv,MCC_MNC_auto_export.csv,0
status,status.csv,STATUS_auto_export.csv,0
```

The file should be edited in cases where the RemoteFilename does not match the already configured value, and to add additional dimensions if required.

4.5 Installing the Analytics Platform Components

In order to load data to Hadoop for the Analytics Platform, each streams blade that loads data must have an analytics client installed on it, which will install all necessary Hadoop components and data loader tools. The following steps outline the full process.

Step 1:

Build bis-client RPM file on the Analytics Platform (IOP master) master node:

```
$ cd /opt/tnf/apps/bis-main/bis-tools/bis-client-rpm

$ ./build.sh
```

Step 2:

Copy to and install bis-client-<server-name>-3.0.0-XXXXX.x86_64.rpm to **all** the Streams Mediation blades (remove first if it exists).

```
# rpm -e bis-client-<server-name>

# rpm -i bis-client-<server-name>-3.0.0-XXXXX.x86_64.rpm
```

Note

Before installing client rpm ensure that previous version of Big Insights profile is removed. As an indication of the previous profile settings still in place next step may give errors related to wrong hadoop, hive or java locations.

Step 3:

Validating your installation

```
# su - boss

$ hive

hive> use tnf;

hive> select count(*) from dual;

hive> exit;
```

You should receive 1 record from dual table.

5 Administration Guide

In the following *instance* represents the instance name as configured in probes.csv or in probes-tdr.csv.

5.1 Starting the applications

Before starting the applications check that they are not already running as detailed in section 5.3.

In order to start the applications, run:

```
/opt/tnf/cna/cna.sh -i instance start
```

To start the Centralised DB Loader for *interface*, run:

```
/opt/tnf/cna/loader.sh -i instance interface start
```

To start the Collector in the stand alone host (“super collector” mode), run:

```
/opt/tnf/cna/collector.sh -i instance start
```

Note: It is also possible start the Collector in the “local collector” mode manually (it is not normally necessary, because it is done from cna.sh automatically). To do this run (the option –l is added):

```
/opt/tnf/cna/collector.sh –l -i instance start
```

5.2 Stopping the applications

To stop the applications, run:

```
/opt/tnf/cna/cna.sh -i instance stop
```

To stop the Centralised DB Loader for *interface*, run:

```
/opt/tnf/cna/loader.sh -i instance interface stop
```

To stop the Collector in the stand alone host (“super collector” mode), run:

```
/opt/tnf/cna/collector.sh [-i instance] stop
```

5.3 Checking the application status

To check what applications are running, use:

```
streamtool lsjobs -i instance
```

To check the internal state of running applications, use:

```
/opt/tnf/cna/cna.sh -i instance status
```

And for the Centralised DB Loader:

```
/opt/tnf/cna/loader.sh -i instance interface status
```

And for the Collector in the stand alone host (“super collector” mode):

```
/opt/tnf/cna/collector.sh [-i instance] status
```

5.4 Cleaning temporary files

To clean up temporary files run (excluding local Collector files):

```
/opt/tnf/cna/cna.sh -i instance clean
```

To clean up local Collector temporary files run:

```
/opt/tnf/cna/collector.sh –l -i instance clean
```

To clean up super Collector temporary files run:

```
/opt/tnf/cna/collector.sh -i instance clean
```

To clean up logs and checkpoints in the probe directories and also the Collector temporary files run:

```
/opt/tnf/cna/cna.sh -i instance cleanprobes
```

To clean all the contents from the loader directory run:

```
/opt/tnf/cna/cna.sh -i instance cleanload
```

To clean all the contents from the Centralised DB Loader directory:

```
/opt/tnf/cna/loader.sh -i instance interface clean
```

5.5 Provisioning updates

5.5.1 Provisioning auto-updates

Provisioning is performed on the master application server which exports CSV files whenever a provisioning change is made. Mediation blades monitor the location of those CSV files and performs an auto update whenever an update is available. This task is performed by the cna.sh health monitor which is invoked approximately every minute, which means that as long as the start command of the cna.sh script has been invoked with the –R option, the provisioning files will be updated automatically. The process is logged in `/opt/tnf/cna/stat/{interface}/health.log`

5.5.2 Manual provisioning auto update

Provisioning files can be updated manually by invoking the following command

```
/opt/tnf/cna/cna.sh -i instance updateprov
```

This command can be used to update the provisioning files at times when the Streams applications are not running.

5.6 Consulting log files

5.6.1 Collector

Collector logs are by default in `/opt/tnf/cna/stat/collector/`.

Each job defined in cfg/collector.xml or cfg/local_collector.xml has its own log file `{date}_CJ_{CollectorName}_Statistics.txt` e.g. `20160421_CJ_Collector_GnNetro_P1_Statistics.txt`. After a day the log file is moved to the archive directory, i.e. `/opt/tnf/cna/stat/collector/archive/`.

To be completed.

5.6.2 Collection

The files `{probe_dir}/{probe_name}/out/statistics/{date}_ite.{i}_Statistics.txt`, where `{probe_dir}` and `{probe_name}` are the values configured in probes.csv (see section 0), `ite.{i}` is the job name as shown in section 5.3, while `{date}` is in the format YYYYMMDD, contain the collection statistics for all chains (4 by default). The content of the files are a list of labelled file attributes, as in the following LTE example (one line per one file):


```
{filename="/opt/tnf/file_loader/vedia/storage/Streams-
ITE/P1/in/data/netro_20120227080601.log.gz",fileType="netro",filesize=17346,urgent=false,repro
cess=false,filetime=1330358761,processingStartedAt="2015-08-26
07:57:06",processingStoppedAt="2015-08-26
07:57:06",duration=0.025828,startTimestamp=(1440601026,719628000,0),sentRecords=238,nRecordsDe
codedASN1=0,nBytesDecodedASN1=0,nBytesReceivedASN1=0,nBytesDroppedASN1=0,recordStatsASN1={},nB
ytesReceivedBIN=0,nBytesDroppedBIN=0,recordCountsBIN={},byteCountsBIN={},recordStatsBIN={},nLi
nesDroppedCSV=0,recordStatsCSV={"m7013":{"errors":0,"records":0},"m7021":{"errors":0,"records"
:232},"m7022":{"errors":0,"records":6}},warnings=[],errors=[],duplicate=false,invalidFile=vali
dFile,rejectedInvalids=0,tableStats={},tableFiles=[],chainSequenceNumber=311,groupId="00:311,0
1:0,02:0,03:0,04:0",chainId="02",logFileDate="20150826"}
```


The fields of interest are:

- filename: identifies the file the attributes refer to;
- sentRecords: contains the total number of TDRs in the file sent to the aggregation contexts;
- recordStatsCSV: this attribute is further divided in specific counters per TDR type;
- invalidFile: can be used to trace rejected files;
- rejectedInvalids: can be used to trace rejected TDRs;
- groupId: this string lists the aggregation contexts (comma separated) and specifies how many of the processed TDRs were sent to each (colon separated) – note that this number can be higher than sentRecords in case TDRs are duplicated for the aggregation;
- chainId: identifies the chain that processed the file.

The rejected TDRs will have a more detailed explanation listed in a corresponding file in `/opt/tnf/cna/stat/{interface}/probes/{probe_name}/rejected/`, together with the line number in the file where the TDR was located and a numeric error code.

5.6.3 Lookup

The statistics for lookup tables are printed to `/opt/tnf/cna/stat/{interface}/{date}_LookupManagerStatistics.txt`, where `{interface}` is the interface type and `{date}` is in the format YYYYMMDD. Example content from the file follows:


```
{filename="",command="SUCCESS request:submit",commandStartedAt="Wed Aug 26 07:56:10
2015",commandProcessedAt="",hostStatistics=[],logFileDate="20150826"}

{filename="/opt/tnf/cna/control/lte/cmd/init_all.cmd",command="init",commandStartedAt="Wed Aug
26 07:56:15 2015",commandProcessedAt="Wed Aug 26 07:56:51 2015",hostStatistics=["Streams
segments:{[sgmnt:mcc_mnc- table:mcc_mnc- reserved:1048576- free:525184-free %:50-
```

```
processed:1637][sgmnt:status- table:status- reserved:2097152- free:1139760-free %:54-
processed:1020][sgmnt:bnumber- table:bnumber- reserved:31457280- free:21855792-free %:69-
processed:89088][sgmnt:cells- table:cells- reserved:10485760- free:4247616-free %:40-
processed:81077][sgmnt:imsi_msisdn- table:imsi_msisdn- reserved:629145600- free:629145600-free
%:100- processed:0]}"],logFileDate="20150826"}

{filename="appl.ctl.cmd",command="SUCCESS request:restart,ite.0",commandStartedAt="Wed Aug 26
07:58:16 2015",commandProcessedAt="",hostStatistics=[],logFileDate="20150826"}
```

In particular, the middle log refers to the tables being loaded: the attribute hostStatistics provides the number of records loaded (*processed*) and the remaining memory (*free %*) for each table. Please ensure that there is remaining memory for all tables; the cache size can be configured at compile-time as specified in section 0.

The user needs to be aware that keys in the provisioned files need to be unique. Otherwise the Lookup Manager fails with this error message:

```
ERROR #splapptrc,J[0],P[0],Main.ResultCollector.ResultCollector,LookupManagerCore
M[LookupManagerCore.splmm:appTrc:873]  -  (LM010) processing command 'init' on
segment 'cells' reported as:[Error detected writing data on command 'init' in
repository 'cells' on host 'dslvm1371': The value not inserted for the key in
segment 'cells'. Reason: Duplicate of key found while inserting in
Map(InternalCellId)=18, in store IntCell. The value is different to the existing:
{lookup_cmd="init",FullCellId="272:03:3134:18",CellName="272:03:3134:18",InternalCe
llId="18"}.] cannot ensure the data integrity.
```

5.6.4 Aggregation

The files `/opt/tnf/cna/stat/{interface}/probes/{probe_name}/context_statistics_{i}.txt`, where `{interface}` is the interface type, `{probe_name}` is the value configured in probes.csv while `{i}` represents the context number, contain the aggregation statistics. An example line from the file follows:

```
{message="file=netro_20120227080601.log.gz, chain=03,
composite=Context00.ContextRecordRegionStream.CustomContext.CustomContext.UP_SubscriberAggrega
teIntermediate, toolate=208,
totalCount={\"20120227075000\":3,\"20120227075500\":2,\"20120227080000\":200,\"20120227072000\
":1,\"20120227080500\":24,\"20120227073000\":2}"}
```

In particular:

- totalCount: can be used to trace the number of TDRs (colon separated) processed in each period (comma separated);
- toolate: can be used to trace the amount of TDRs ignored in the period.

5.6.5 Database

The Centralised DB Loader logs are by default in `/opt/tnf/cna/stat/{interface}/log/`.

The files in `/opt/tnf/cna/stat/{interface}/loader_script/`, contain eventual errors relating to the DB loader.

The variables logFile and dataLoaderLog in the file `/opt/tnf/cna/etc/script/loader.ini` contain the path of the log files for the loader scripts.

The `/opt/tnf/cna/etc/script/checkLoaderLog.sh` script can be invoked manually at any time to printout the DB loader performance statistics. Sample output:

```
Start Time                      = 20151201 11:14:27.458
End Time                        = 20151201 13:05:48.642
Successfully loaded files       = 3636
Number of failed files          = 289
Total processed records         = 3598002
Size of loaded files (Bytes)    = 1266488951
Average records in one file     = 989
Average file size (Bytes)       = 348319
Duration (sec.)                 = 6681
Duration (min.)                 = 111:21
Processed records/sec.          = 538
Processed records/sec/core      = 134
Processed records/min.          = 32280
Processed input data KB/sec.    = 185
Processed input data KB/sec/core = 46
Average record size (Bytes)     = 351
```

These statistics are based on the contents of the `/opt/tnf/cna/stat/*/loader_script/loader.log` log file. Note that at the moment the script can only parse one file, i.e. one active interface.

5.6.6 Performance

The file `/opt/tnf/cna/stat/{interface}/vmstat.stat`, where `{interface}` is the interface type, contains the memory and CPU statistics measured during the applications' current life time, as measured by the vmstat command. The monitoring daemon's pid is stored in `/opt/tnf/cna/control/{interface}/vmstat.pid`, and can be used to monitor its status. The daemon is killed during the stop command and both files are reset upon a full restart.

5.6.7 Streams logs

Every time a Streams job is stopped, either by the “stop” command or by the automated recovery mechanism outlined in the next section, its Streams logs are archived to `/opt/tnf/cna/stat/{interface}/StreamsLogsJob{id}.tgz`, where `{interface}` is the interface type and `{id}` is the (former) Streams job identifier. Refer to the Streams documentation for what the individual log files mean.

5.7 Recovery mechanisms

All applications aside from the Centralised DB Loader (which is configured to be automatically restarted by Streams) are monitored every minute during the application's life time; if a Streams job fails or reports a status different from “run” (fully operational), the job is restarted. The monitoring daemon is started automatically by the “start” command, and is killed by the “stop” command; the process has the same name as the management script, *cna.sh*, and its process ID is stored in `/opt/tnf/cna/control/{interface}/health.pid`, which can be used to monitor its status, e.g. with

```
ps -fp `cat /opt/tnf/cna/control/{interface}/health.pid`
```

If no process is listed in the command's output, the monitoring process might have been killed or terminated unexpectedly: the process might have to be killed manually to examine Streams logs that

would be removed during a restart. In this case there is no embedded mechanism to restart it, and it's recommended to manually restart the applications to restore the functionality.

The file `/opt/tnf/cna/stat/{interface}/health.log` contains the health monitor logs and can be used to detect instabilities in the system. Streams-related instabilities might have to be analysed using the individual job and PE logs (see related Streams documentation).

In the following example content:

```
08/26/15 07:56:37 === Started monitoring instance cna_lte
08/26/15 07:57:48 DBLoader (job #16): status Running/yes/run
08/26/15 07:57:58 LookupManager (job #14): status Running/yes/run
08/26/15 07:57:58 ite.0 - P1 (job #-): status -/terminated
streamtool submitjob /opt/tnf/cna/bin/ITE_LTE.sab -C data-
directory=/opt/tnf/file_loader/vedia/storage/Streams-ITE/P1 -P probeId=P1 -P ite.jobName=ite.0
-P netroFileMask='netro_([0-9]{14})\\.log\\.gz'\
         -P pagebasedFileMask='pbqoe_([0-9]{14})\\.log\\.gz'\
         -P appSplitterEnabled=false\
         -P numContextsUP=3\
         -P numContextsUPS=1\
         -P numContextsCPCPS=1\
         -P loadUnknownSubscribers=false\
         -P filterCPsuccessCodes=true\
         -P CpsuccessCodes=16,17,18,19\
         -P ignorePageBased=false\
         -P roamersFilter=subscriber\
         -P global.applicationControlDirectory=/opt/tnf/cna/control/lte
CDISC0079I The following number of applications were submitted to the cna_lte instance: 1. The
instance is in the cna domain.
CDISC0080I The 17 job was submitted for the following application: ITE_LTE.sab. The job was
submitted to the cna_lte instance in the cna domain.
CDISC0020I Submitted job IDs: 17
08/26/15 07:59:24 DBLoader (job #16): status Running/yes/run
08/26/15 07:59:32 LookupManager (job #14): status Running/yes/run
08/26/15 07:59:40 ite.0 - P1 (job #17): status Running/yes/run
```

We can see that the applications were started at 07:56, but at 07:57 the ITE LTE application had been detected as terminated; it was immediately restarted (the individual parameters depend on the content of the ITE configuration file), and at 07:59 all applications are healthy.

In case of a failure of an ITE job, as in the above example, the last 5 processed files are also resubmitted for collection; the ITE application will automatically identify TDRs that had been already aggregated and possibly written and will not output them again to prevent duplicates.

In case of a failure of the LookupManager a full restart is automatically performed.

5.8 Housekeeping

The following files have to be cleaned periodically using a set of crontab commands:

- Feed files and their symlinks downloaded by the Collector:
  By default in `/opt/tnf/cna/tdrs/`
  Clean files not updated more than 2 hours.

- Feed files (or their symlinks) processed by the ITE or RawTDR jobs, and also the stat files:
  By default also in `/opt/tnf/cna/tdrs/`
  Clean files not updated more than 5 days.

- Aggregated or RawTDR dataset files in generated by ITE or RawTDR jobs:
  By default in `/opt/tnf/cna/loaderDir/`
  Clean files not updated more than 2 hours.

- Statistics files generated by all Streams applications:
  By default in `/opt/tnf/cna/stat/`
  Clean files not updated more than 5 days.

- Old provisioning files already loaded into the Lookup Manager
  By default in `/opt/tnf/cna/control/gn/provisioning/archive/`
  Clean files not updated more than 5 days.