

# The CERN Tape Archive: Administrator Guide

Vladimír Bahyl    Germán Cancio    Eric Cano    Michael Davis    Steven Murray

December 1, 2017

# Contents

<b>Contents</b>	<b>1</b>
<b>1 Introduction</b>	<b>2</b>
1.1 CTA Overview	2
1.2 Authentication between CTA Servers	2
<b>2 Install CTA</b>	<b>3</b>
2.1 Get the CERN Tape Archive Repository	3
2.2 Configure yum	3
2.3 Install XRoot and EOS	4
2.4 Install CTA RPMs	4
2.5 [Optional] Compile CTA from Source	5
2.6 Configure the ObjectStore	5
2.6.1 Configuration for a test system	6
2.6.2 Configuration for a production system	6
2.7 Configure the Catalogue	7
2.8 Configure CTA Front End	8
2.8.1 CTA Front End Kerberos Authentication	9
2.8.2 CTA Front End SSS Authentication	9
2.8.3 CTA Front End XRoot Configuration	11
2.8.4 Create a CTA Admin Host and User	12
2.9 Install the Tape Server	12
2.9.1 Install the Tape Library	12
2.9.2 Install the Remote Media Changer Daemon ( <code>cta-rmcd</code> )	12
2.9.3 Configure the CTA Tape Server Daemon	13
2.10 Start the XRoot daemon with CTA Front End plugin	14
2.11 Configure EOS Workflows	14
2.12 Removing CTA	16
2.13 To Do	16
2.13.1 RPM and Configuration File Names	16
2.13.2 Describe properly RPM dependencies (in the <code>.spec</code> files).	16
2.13.3 <code>ctacli.sh</code> contains comment: <code>&lt;FQDN&gt;:&lt;TCPPort&gt;</code> yet the value mentioned does not have the domain specified.	16
2.13.4 PLEASE make sure the <code>'status='</code> string is ALWAYS on the same position in the <code>cta-taped</code> logs.	17
<b>3 Start CTA</b>	<b>18</b>
3.1 Debugging Authentication Problems	18
3.2 Trying out some commands	18
3.3 Running <code>tapeserverd</code>	19
3.3.1 Tape labelling	19
3.3.2 Preparing the CTA environment for migrations	19

<b>4</b>	<b>Administrator Commands</b>	<b>20</b>
4.1	cta admin	20
4.2	cta adminhost	21
4.3	cta user	21
4.4	cta tapepool	21
4.5	cta archiveroute	22
4.6	cta logicallibrary	22
4.7	cta tape	22
4.8	cta storageclass	23
4.9	cta list...	23
4.10	Questions administrators need to be able to answer	23
<b>5</b>	<b>User Commands</b>	<b>24</b>
<b>A</b>	<b>Install a Local EOS Instance</b>	<b>25</b>
A.1	Install XRoot and EOS RPMs	25
A.2	Create a Kerberos keytab file	25
A.3	Create a Simple Shared Secret keytab file	27
A.4	Configure EOS sysconfig	27
A.5	Configure XRoot mgm	28
A.6	Configure XRoot fst	29
A.7	Start XRoot daemons	29
A.8	Enable EOS authentication mechanisms	29
A.9	Configure the default EOS space	29
A.10	Test the EOS Installation	30
A.11	Installing EOS in Docker	30
<b>B</b>	<b>Install the Virtual Tape Library (mhVTL)</b>	<b>32</b>
B.1	Install mhVTL	32
B.2	Configure mhVTL	32
B.2.1	udev rules	33
B.3	Start mhVTL	34
B.4	Virtual Tape Library Commands	35
<b>C</b>	<b>Tape Format</b>	<b>37</b>
C.1	Overview	37
C.2	Volume Label (VOL <sub><i>n</i></sub> )	37
C.3	Header Label (HDR <sub><i>n</i></sub> )	38
C.4	User Header Label (UHL <sub><i>n</i></sub> )	39
C.5	Data Records	39
C.6	End of File (EOF <sub><i>n</i></sub> )	40
C.7	User Trailer Label (UTL <sub><i>n</i></sub> )	41
C.8	Checksums	41

# Chapter 1

## Introduction

The CERN Tape Archive (CTA) is a tape storage back-end to EOS.

This document is a guide for administrators to install and administer a CTA system.

### 1.1 CTA Overview

Documentation should explain and separate the various functions of individual servers:

- EOS
- CTA FrontEnd
- ObjectStore
- Database
- Tape Server

### 1.2 Authentication between CTA Servers

Section [2.8](#) and Figure [2.1](#) explain why multiple Kerberos and Simple Shared Secret (SSS) keys are necessary. Probably we should move the conceptual information here and leave the instructions in Chapter [2](#).

# Chapter 2

## Install CTA

This chapter describes how to install CTA together with a local EOS instance. These instructions assume that the target system is CERN CentOS 7 (CC7). In principle CTA can be installed on SLC6 and the steps should be more or less the same. Installation follows these steps:

1. Configure yum (Section [2.2](#))
2. Install XRoot and EOS (Section [2.3](#))
3. Install CTA RPMs (Section [2.4](#))
4. Configure the ObjectStore (Section [2.6](#))
5. Configure the Catalogue (Section [2.7](#))
6. Configure the CTA Front End (Section [2.8](#))
7. Install the Tape Server (Section [2.9](#))

### 2.1 Get the CERN Tape Archive Repository

Some of the files required for installation are held in the CTA git repository. It is necessary to install the CTA repo until it is puppetized.

Clone the [CTA git repository](#) (using Kerberos authentication):

```
# cd ~  
# git clone https://:@gitlab.cern.ch:8443/cta/CTA.git
```

Alternative URLs using HTTPS or SSH authentication:

```
# git clone https://gitlab.cern.ch/cta/CTA.git  
# git clone ssh://git@gitlab.cern.ch:7999/cta/CTA.git
```

### 2.2 Configure yum

Ensure the yum priorities plugin is installed, to permit assigning different priorities to each yum repository:

```
# yum install yum-plugin-priorities
```

Ensure that the `cernonly` repo is enabled in `CentOS-CERN.repo`:

```
# yum-config-manager --enable cernonly
```

Copy the extra yum repo files from the CTA git repo:

```
# cd ~/CTA/continuousintegration/docker/ctafontend/cc7/etc/yum.repos.d
# yum-config-manager --add-repo=eos.repo
# yum-config-manager --add-repo=cta-ci.repo
# yum-config-manager --add-repo=castor.repo
# cd ~/CTA/continuousintegration/buildtree_runner/vmBootstrap
```

EOS RPMs are installed from the `eos` repo. The repo file has `eos-citrine` (disabled) and `eos-citrine-commit` (enabled). What is the difference between these two repos?

XRoot RPMs are installed from the `epel` repo. If this is not installed:

```
# yum install epel-release
```

CTA release 0.5 will use the Scalable Service Interface (SSI), which is not yet packaged with the released version of XRoot. So we will need our own XRoot repo.

## 2.3 Install XRoot and EOS

CTA and EOS communicate using the XRoot protocol. The CTA project requires XRoot version 4 or higher. The EOS instance must use the same version of XRoot.

To avoid problems with incompatible EOS/XRoot versions, install a yum version lock list:

```
# cd ~/CTA/continuousintegration/docker/ctafontend/cc7/etc/yum/pluginconf.d/
# cat versionlock.list >>etc/yum/pluginconf.d/versionlock.list
```

Now install the RPMs:

```
# yum install xrootd-client xrootd-debuginfo xrootd-server
```

If EOS has not been installed, follow the instructions in [Appendix A](#) to install a local EOS instance.

## 2.4 Install CTA RPMs

To install CTA:

```
# yum -y install cta-catalogueutils cta-debuginfo cta-frontend \
    cta-objectstore-tools mt-st mtx lsscsi sg3_utils
# yum clean packages
```

## 2.5 [Optional] Compile CTA from Source

If the correct version of the required CTA RPMs is not available, they can be built from source as described in this section.

First, build the source RPM of the CTA project by running `cmake` against the CTA sources that were installed in Section 2.1:

```
# cd ~
# mkdir CTA_build
# cd CTA_build
# cmake -DPackageOnly:Bool=true ../CTA
# make cta_srpm
```

Use this to install all the dependencies which are required to build CTA:

```
# yum-builddep RPM/SRPMS/cta-0-0.src.rpm
```

Delete the contents of the `CTA_build` directory, then re-run `cmake` to prepare to build the complete CTA project (not just the source RPMs):

```
# cd ..
# rm -rf CTA_build/
# mkdir CTA_build
# cd CTA_build
# cmake ../CTA
```

Build CTA by running `make`:

[ There is an issue with the dependencies of some of the auto-generated source code. The workaround is to type `make` twice. ]

```
# make
# make
```

Build the CTA RPMs:

```
# make cta_rpm
```

Install the CTA RPMs:

```
# yum -y install RPM/RPMS/x86_64/cta-*
```

## 2.6 Configure the ObjectStore

There are two types of ObjectStore: Virtual File System (VFS) or **Ceph** distributed storage system.

## 2.6.1 Configuration for a test system

In a test system, configure the ObjectStore as a VFS. Initialise the new ObjectStore VFS and set global `rwX` permissions:

```
# export OBJECTSTORETYPE=file
# export OBJECTSTOREURL=$(cta-objectstore-initialize | sed 's/^.*file:\/\\\/\\\/')
# chmod -R 0777 $OBJECTSTOREURL
# ls -ld $OBJECTSTOREURL
drwxrwxrwx. 2 root root 240 May  9 11:22 /tmp/jobStoreVFSg4Mqz4/
```

Add it as a configuration parameter in the `cta-taped.conf` and `cta-frontend.conf`, as in the following example:

```
ObjectStore BackendPath /tmp/jobStoreVFSOKJCjW
```

## 2.6.2 Configuration for a production system

In a production system, configure the ObjectStore to use Ceph. To install Ceph:

```
# yum-config-manager --enable ceph
# yum -y install ceph-common
```

The Ceph client version must match the version of the Ceph server you are connecting to. If there are problems connecting, check the required version number with the Ceph server administrator. At the time of writing, the required version is:

```
# ceph --version
ceph version v11.0.0-2590-g08becd3 (08becd34a82b2c541f0aeeaf28c4038faf268d47)
```

Specify the location of the Ceph cluster monitor daemon in `/etc/ceph/ceph.conf`. To use the CERN Ceph cluster monitor:

```
[global]
    mon host = cephmon.cern.ch:6790
```

Configure the following environment variables to specify the ObjectStore. For convenience, save these in a file (e.g. `/etc/sysconfig/ceph.conf`) which can be sourced from the shell:

```
# Initialise ObjectStore configuration with:
#
# . /etc/sysconfig/ceph.conf
#
# Declare which type of ObjectStore you have. Possible values: file or ceph
export OBJECTSTORETYPE=ceph

# Define the ObjectStore ID, which corresponds to the Ceph user name. The Ceph
# service manager should provide this value.
export OBJECTSTOREID=eoscta

# Define the ObjectStore pool name on Ceph. The Ceph service manager should
```



```
# provide this value.
export OBJECTSTOREPOOL=eoscta_metadata

# Define the ObjectStore namespace name which will be prepended to all objects
# in the ObjectStore. The CTA service manager defines this value. Usually it is
# kept as "cta-ns".
export OBJECTSTORENAMESPACE=cta-ns

# This is just to construct the URL from the above values
export
    OBJECTSTOREURL=rados://${OBJECTSTOREID}@${OBJECTSTOREPOOL}:${OBJECTSTORENAMESPACE}
```

Create the Ceph keyring for this CTA instance (/etc/ceph/ceph.client.\$OBJECTSTOREID.keyring). Fill in the values which correspond to the environment variables defined above:

```
[client.OBJECTSTOREID]
key = CEPH_OBJECTSTORE_SECRET_KEY
caps mon = "allow r"
caps osd = "allow rwx pool=OBJECTSTOREPOOL namespace=OBJECTSTORENAMESPACE"
```

There is also a configuration file /etc/ceph/rbdmap but this appears to contain only comments:

```
# RbdDevice      Parameters
#poolname/imagename id=client,keyring=/etc/ceph/ceph.client.keyring
```

Initialise the ObjectStore:

```
# cta-objectstore-initialize $OBJECTSTOREURL
```

To list the ObjectStore content:

```
# rados -p $OBJECTSTOREPOOL --id $OBJECTSTOREID --namespace $OBJECTSTORENAMESPACE \
ls
cta-objectstore-initialize-p06253947b39467.cern.ch-192840-20170512-12:56:25 root
driveRegister-cta-objectstore-initialize-p06253947b39467.cern.ch-192840-20170512-12:5
agentRegister-cta-objectstore-initialize-p06253947b39467.cern.ch-192840-20170512-12:5
schedulerGlobalLock-cta-objectstore-initialize-p06253947b39467.cern.ch-192840-2017051
```

To delete the ObjectStore content:

```
# rados -p $OBJECTSTOREPOOL --id $OBJECTSTOREID --namespace $OBJECTSTORENAMESPACE \
ls | xargs -itoto rados -p $OBJECTSTOREPOOL --id $OBJECTSTOREID --namespace \
$OBJECTSTORENAMESPACE rm toto
```

## 2.7 Configure the Catalogue

The Catalogue is stored in an Oracle database. The database connection string is specified in /etc/cta/cta\_catalogue\_db.conf in the format oracle:username/password@database.

CTA can also be configured to connect to SQLite, using either the `in_memory` or `sqlite:filename` option. These options are intended for unit tests only. There are various reasons why SQLite is not suitable for system tests: Resource starvation; SQLite and Oracle handle locking differently; and test coverage (we need to use Oracle part of the system that will be used in production).

`/etc/cta/cta_catalogue_db.conf` should contain exactly one connection string. For example, to configure CTA to use database `devdb12` with login name `cta_test` and password `MYSECRET`:

```
oracle:cta_test/MYSECRET@devdb12
```

To use this configuration to create a new database schema:

```
# cta-catalogue-schema-create /etc/cta/cta-catalogue.conf
```

To drop an existing database schema:

```
# cta-catalogue-schema-drop /etc/cta/cta-catalogue.conf
WARNING
You are about to drop the schema of the CTA catalogue database
Database name: devdb12
Are you sure you want to continue?
Please type either "yes" or "no" > yes
DROPPING the schema of the CTA catalogue database
Dropped table CTA_CATALOGUE
Dropped table ARCHIVE_ROUTE
Dropped table TAPE_FILE
Dropped table ARCHIVE_FILE
Dropped table TAPE
Dropped table REQUESTER_MOUNT_RULE
Dropped table REQUESTER_GROUP_MOUNT_RULE
Dropped table ADMIN_USER
Dropped table ADMIN_HOST
Dropped table STORAGE_CLASS
Dropped table TAPE_POOL
Dropped table LOGICAL_LIBRARY
Dropped table MOUNT_POLICY
Dropped sequence ARCHIVE_FILE_ID_SEQ
```

## 2.8 Configure CTA Front End

Create the CTA Front End configuration file `/etc/cta/cta-frontend.conf`. The ObjectStore Backend should point to the `$OBJECTSTOREURL` specified in [Section 2.6](#):

```
ObjectStore BackendPath rados://cta-id@cta-tapepool:cta-ns
Catalogue NumberOfConnections 1
Log URL file:/cta-frontend.log
```

Create the corresponding logfile:

```
# touch /cta-frontend.log
# chmod a+w /cta-frontend.log
```

Create the CTA user:

```
# useradd cta
```

The CTA Front End requires both Kerberos and SSS authentication: Kerberos is used to authenticate user archive/retrieve commands and admin commands. SSS is used to authenticate communication between the Front End and CTA/EOS `mgm`. See Figure 2.1 for an overview.

### 2.8.1 CTA Front End Kerberos Authentication

CTA uses one of the Kerberos keys from `/etc/krb5.keytab`. If this file does not exist, see Appendix A.2 for details of how to create it. This file should be copied to the CTA Front End keytab which should be owned by the user/group which will run the CTA Front End XRoot daemon:

```
# cd /etc
# cp krb5.keytab krb5.keytab.cta
# chown cta:cta krb5.keytab.cta
```

Check the contents of the new keytab:

```
# echo -e "read_kt krb5.keytab.cta\nlist\nquit" | ktutil
```

### 2.8.2 CTA Front End SSS Authentication

There will be one EOS instance per User (Atlas, CMS, etc.), each of which can send archive and retrieve requests to the CTA Front End. Each EOS instance should have its own Simple Shared Secret (SSS) key<sup>1</sup>.

The EOS instance name is used as the user name for the SSS key. In the case of a local EOS instance (See Appendix A), this can be found in the `eos_env` configuration file:

```
# grep EOS_INSTANCE_NAME /etc/sysconfig/eos_env
EOS_INSTANCE_NAME=eosdev
```

Use the instance name to create a SSS key for communication between the CTA and the EOS instance and add it to the keytab for the CTA Front End:

```
# cd /etc
# xrdsssadmin -k cta_eosdev -u eosdev -g cta add ctafrontend_server_sss.keytab
xrdsssadmin: Keyfile 'ctafrontend_server_sss.keytab' does not exist. Create it? (y
| n): y
xrdsssadmin: 1 key out of 1 kept (0 expired).
# chown cta ctafrontend_server_sss.keytab
# chmod 600 ctafrontend_server_sss.keytab
```

The `-k` option specifies the key name that the `-u` (user) and `-g` (group) options will be applied to, overriding the default `nobody/nogroup`.

If the keyname ends with `+`, SSS tokens may be forwarded when encrypted by the associated key, i.e. the key can be used by a different host from the one that encrypted the SSS token. This

---

<sup>1</sup>In principle, each instance should have a unique key. In practice, it may be that all instances share the same key.

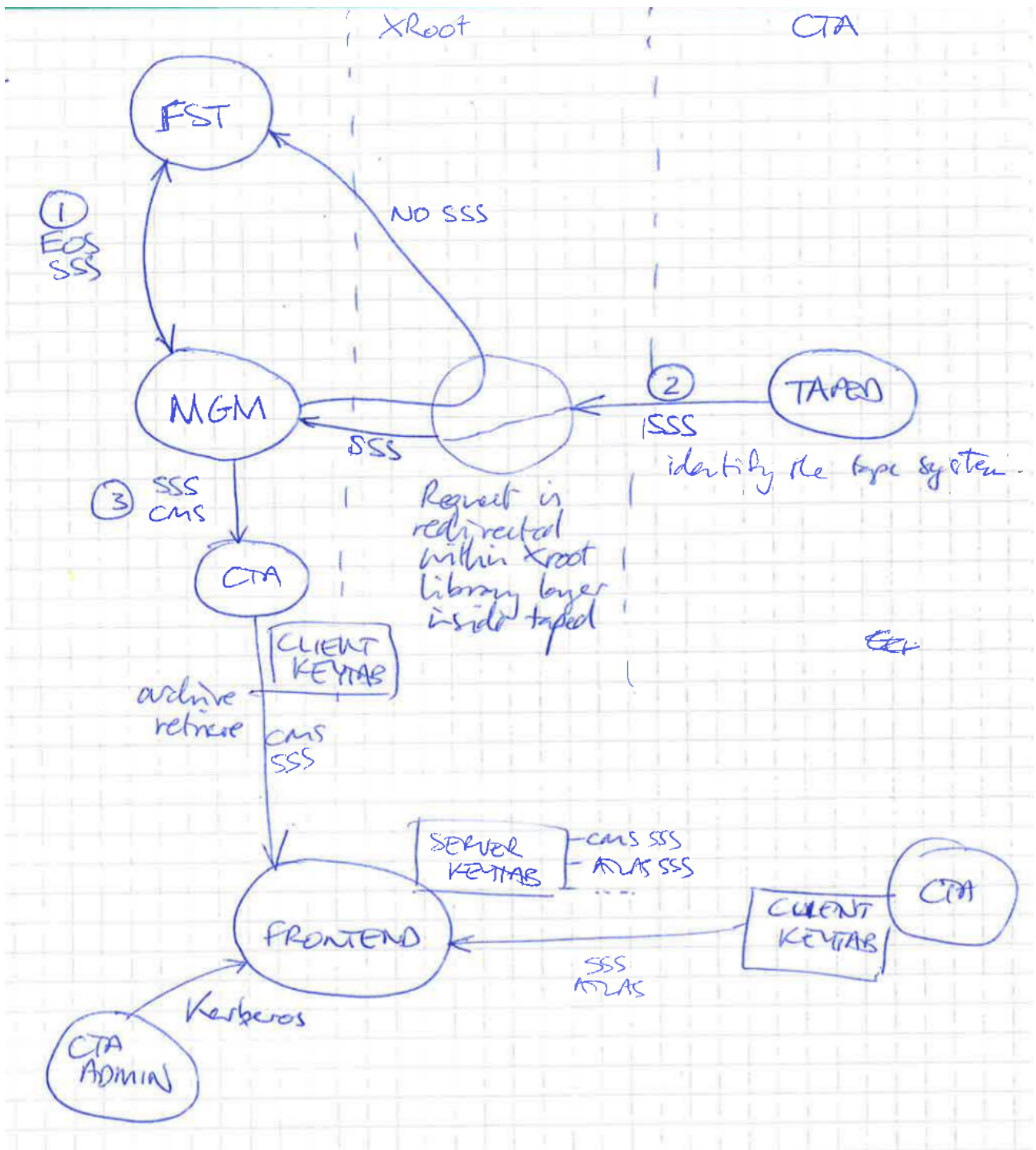


Figure 2.1: Authentication Mechanisms in CTA

is required in certain situations, for example when tunnelling through a NAT device or when creating keys for use in the Kubernetes environment.

In production, the `+` should be omitted, as forwardable keys are inherently less secure. Allowing forwarded tokens makes it impossible to detect man-in-the-middle attacks or stolen SSS tokens.

The SSS key that we have just created also needs to be available to the EOS client, so we copy the server keytab into the client keytab:

```
# cp ctafrontend_server_sss.keytab ctafrontend_client_sss.keytab
# chmod 600 ctafrontend_client_sss.keytab
# chown cta ctafrontend_client_sss.keytab
```

As mentioned above, in principle there should be one key per EOS instance, so `ctafrontend_client_sss.keytab` may contain more than one key. However, as the XRoot client only uses the **last** key added to the keytab, the entire keytab can be safely copied to the client (so long as the key for that instance is the last one that was added).

### 2.8.3 CTA Front End XRoot Configuration

The EOS XRoot daemons (`fst`, `mgm`, `mq`) have their configuration files under `/etc/xrd.cf.<daemon>`. Note that the CTA XRoot configuration files are not in the same place, instead they are under `/etc/xrootd/<daemon>.cfg`.

Edit `/etc/cta/cta-frontend-xrootd.conf` and ensure that both Kerberos and SSS are enabled and using the correct keys. This is specified using the `sec.protocol` and `sec.protbinding` configuration lines. The file should finish looking something like this:

```
# Load the plugin XrdCtaOfs
xrootd.fslib libXrdCtaOfs.so

# Use the security module
xrootd.seclib libXrdSec.so

# Protocol specification
# The xroot server process needs to be able to read the keytab file
sec.protocol krb5 /etc/krb5.keytab.cta cta/cta-frontend@TEST.CTA
sec.protocol sss -s /etc/ctafrontend_server_sss.keytab -c
    /etc/ctafrontend_client_sss.keytab

# Only Kerberos 5 and sss are allowed
sec.protbinding * only sss krb5

# Allow copying from absolute paths
all.export /

# Turn off asynchronous i/o
xrootd.async off

# Use a port other than 1094, already used by EOS xroot server
xrd.port 10956
```

Configure the CTA Command Line Interface (CLI) to talk to the same host:port as specified above:

```
# The CTA frontend address in the form <FQDN>:<TCPPort>
<host>.cern.ch:10956
```

## 2.8.4 Create a CTA Admin Host and User

In order to use the `cta admin` command (see Section 4.1), the admin host and user must already be defined. The `cta-catalogue-admin-host-create` and `cta-catalogue-admin-user-create` commands are provided to avoid the circular problem of not being able to use `cta admin` to create the admin host and user, by connecting directly to the catalogue database.

```
# cta-catalogue-admin-host-create /etc/cta/cta-catalogue.conf -n adminhost.cern.ch
-c "Administrative Host"
# cta-catalogue-admin-user-create /etc/cta/cta-catalogue.conf -u adminl -c
"Administrative User"
```

## 2.9 Install the Tape Server

### 2.9.1 Install the Tape Library

If the CTA instance is not using real tape hardware, follow the directions in Appendix B to install the mhVTL Virtual Tape Library.

### 2.9.2 Install the Remote Media Changer Daemon (`cta-rmcd`)

The Remote Media Changer Daemon (`cta-rmcd`) is a TCP/IP server which controls the robots in the tape libraries. `cta-rmcd` must run locally for security reasons: it only listens to the localhost network. It includes a device driver for the drive and one for the library.

Install `cta-rmcd`:

```
# yum install -y castor-rmc-client castor-rmc-server initscripts
```

[ The dependency on `initscripts` will be removed when `cta-rmcd` is ported from `init.d` to `systemd`. ]

The configuration file `/etc/sysconfig/cta-rmcd` should contain the following settings:

```
DAEMON_COREFILE_LIMIT=unlimited
RUN_RMCD=yes
RMCD_OPTIONS=/dev/smc
```

Start `cta-rmcd`:

```
# systemctl start cta-rmcd
```

Now the command `smc` can be used to load/unload tapes (instead of `mtx`).

### 2.9.3 Configure the CTA Tape Server Daemon

Create `/etc/cta/cta.conf` and configure the number of buffers and the ObjectStore URL. The following example configures 10 buffers of 5 Mb. (The buffer size in this case is the compile-time default, see values below). Replace `OBJECTSTOREURL` with the value defined in Section 2.6):

```
taped BufferCount 10
ObjectStore BackendPath OBJECTSTOREURL
```

Define the tape drives in `/etc/cta/TPCONFIG`. The four columns are Tape Drive, Logical Library, No-rewind SCSI Device, SCSI Media Changer Device:

```
VDSTK1 VLSTK /dev/nst0 smc0
VDSTK2 VLSTK /dev/nst1 smc1
```

Now start the Tape Server. `cta-taped` is an XRootD client, linked with `libXrdCl.so`. This library needs to be told which authentication method to use: set `XrdSecPROTOCOL` to use `sss` authentication (to talk to the `mgm`) and also `unix` authentication (to talk to the `fst`). Also specify the location of the keytab in `XrdSecSSSKT`:

```
# XrdSecPROTOCOL=sss,unix XrdSecSSSKT=/etc/cta-taped.keytab cta-taped
# cta listdrivestates
smc0 devbox.cern.ch VDSTK1 Down NoMount 0 1495195920 0 0 0.000000
smc1 devbox.cern.ch VDSTK2 Down NoMount 0 1495195920 0 0 0.000000
```

Log messages are sent to `rsyslog`. When `cta-taped` starts, it reports its configuration. This is set from compile-time defaults, overwritten by any values set in `/etc/cta/cta.conf` (see above):

```
# grep cta-taped /var/log/messages
TID="430" MSG="cta-taped started" version="0-0" foreground="0" logToStdout="0"
logToFile="0" logFilePath="" configFileLocation="/etc/cta/cta.conf"
helpRequested="0"
TID="430" MSG="Configuration entry" category="taped" key="LogMask" value="DEBUG"
source="Compile time default"
TID="430" MSG="Configuration entry" category="taped" key="TpConfigPath"
value="/etc/cta/TPCONFIG" source="Compile time default"
TID="430" MSG="Configuration entry" category="taped" key="BufferSize"
value="5248000" source="Compile time default"
TID="430" MSG="Configuration entry" category="taped" key="BufferCount" value="10"
source="/etc/cta/cta.conf:1"
TID="430" MSG="Configuration entry" category="taped" key="ArchiveFetchBytesFiles"
maxBytes="80000000000" maxFiles="500" source="Compile time default"
TID="430" MSG="Configuration entry" category="taped" key="ArchiveFlushBytesFiles"
maxBytes="32000000000" maxFiles="200" source="Compile time default"
TID="430" MSG="Configuration entry" category="taped" key="RetrieveFetchBytesFiles"
maxBytes="80000000000" maxFiles="500" source="Compile time default"
TID="430" MSG="Configuration entry" category="taped" key="NbDiskThreads"
value="10" source="Compile time default"
TID="430" MSG="Configuration entry" category="taped"
key="WatchdogIdleSessionTimer" value="10" source="Compile time default"
TID="430" MSG="Configuration entry" category="taped" key="WatchdogMountMaxSecs"
value="900" source="Compile time default"
TID="430" MSG="Configuration entry" category="taped"
key="WatchdogNoBlockMoveMaxSecs" value="1800" source="Compile time default"
TID="430" MSG="Configuration entry" category="taped" key="WatchdogScheduleMaxSecs"
value="60" source="Compile time default"
TID="430" MSG="Configuration entry" category="general" key="ObjectStoreURL"
value="/tmp/jobStoreVFSAgmdey" source="/etc/cta/cta.conf:2"
```

```
TID="430" MSG="Configuration entry" category="general" key="FileCatalogConfigFile"
value="/etc/cta/cta-catalogue.conf" source="Compile time default"
TID="430" MSG="Configuration entry" category="TPCONFIG Entry" unitName="VDSTK1"
logicalLibrary="VLSTK" devFilename="/dev/nst0" librarySlot="smc0"
source="/etc/cta/TPCONFIG:1"
TID="430" MSG="Configuration entry" category="TPCONFIG Entry" unitName="VDSTK2"
logicalLibrary="VLSTK" devFilename="/dev/nst1" librarySlot="smc1"
source="/etc/cta/TPCONFIG:2"
```

This is followed by the startup messages:

```
TID="430" MSG="Set log mask" logMask="DEBUG"
TID="430" MSG="Set process capabilities" capabilities="= cap_setgid,cap_setuid+ep
cap_sys_rawio+p"
TID="431" MSG="Got dumpable attribute of process" dumpable="true"
TID="431" MSG="Set process capabilities" capabilities="= cap_sys_rawio+p"
TID="431" MSG="Adding handler for subprocess" SubprocessName="signalHandler"
TID="431" MSG="Adding handler for subprocess" SubprocessName="drive:VDSTK1"
TID="431" MSG="Adding handler for subprocess" SubprocessName="drive:VDSTK2"
TID="431" MSG="Adding handler for subprocess" SubprocessName="garbageCollector"
TID="431" MSG="Subprocess handler requested forking" SubprocessName="drive:VDSTK1"
TID="431" MSG="Subprocess handler will fork" SubprocessName="drive:VDSTK1"
TID="431" MSG="Subprocess handler requested forking" SubprocessName="drive:VDSTK2"
TID="432" MSG="In child process. Running child." SubprocessName="drive:VDSTK1"
TID="431" MSG="Subprocess handler will fork" SubprocessName="drive:VDSTK2"
TID="431" MSG="Subprocess handler requested forking"
SubprocessName="garbageCollector"
```

## 2.10 Start the XRoot daemon with CTA Front End plugin

Start the XRoot daemon, specifying the name of the XRoot instance (`-n`), the name of the configuration file we just created (`-c`), use IPv4 protocol (`-I v4`), run in the background (`-b`) and where to put the log file (`-l`)<sup>2</sup>:

```
# su - cta
[~]$ xrootd -n cta -c /etc/cta/cta-frontend-xrootd.conf -I v4 -b -l
/tmp/ctafontend.log
```

Once the XRoot daemon is running, the CTA CLI will be able to send commands and receive results (See Chapters 4 and 5).

## 2.11 Configure EOS Workflows

The workflows define how EOS will talk to the CTA Front End. The following EOS workflows should be registered:

- Archive workflow (executed on CLOSEW)
- Retrieve workflow (executed on PREPARE)
- Delete Archive workflow (executed on DELETE)

Create a workflow namespace directory. This will be used to store the workflow attributes but will not store any files:

<sup>2</sup>For full documentation of command line options, see [http://xrootd.org/doc/dev44/xrd\\_config.htm](http://xrootd.org/doc/dev44/xrd_config.htm)



```
# eos mkdir -p /eos/dev/proc/cta/workflow
```

Link the attributes of the data directory to the attributes of the workflow directory. This causes the workflow configuration to be inherited by the data directory:

```
# eos attr link /eos/dev/proc/cta/workflow /eos/users/test
```

Configure the three workflows mentioned above. Archive workflow:

```
# eos attr set "sys.workflow.closew.default=bash:shell:cta 'cta archive
--user <eos::wfe::username>
--group <eos::wfe::rgroupname>
--diskid <eos::wfe::fid>
--instance eoscta
--srcurl root://devbox.cern.ch/<eos::wfe::path>
--size <eos::wfe::size>
--checksumtype <eos::wfe::checksumtype>
--checksumvalue <eos::wfe::checksum>
--storageclass <eos::wfe::cxattr:CTA_StorageClass>
--diskfilepath <eos::wfe::path>
--diskfileowner <eos::wfe::username>
--diskfilegroup <eos::wfe::groupname>
--recoveryblob:base64 <eos::wfe::base64:metadata>
--diskpool default --throughput 10000 1>&2'" .
```

Retrieve workflow:

```
# eos attr set "sys.workflow.sync::prepare.default=bash:shell:cta 'cta retrieve
--user <eos::wfe::username>
--group <eos::wfe::rgroupname>
--id <eos::wfe::fxattr:sys.archiveFileId>
--dsturl <eos::wfe::turl>
--diskfilepath <eos::wfe::path>
--diskfileowner <eos::wfe::username>
--diskfilegroup <eos::wfe::groupname>
--recoveryblob:base64 <eos::wfe::base64:metadata>
--diskpool default --throughput 10000 1>&2'" .
```

Delete Archive workflow:

```
# eos attr set "sys.workflow.delete.default=bash:shell:cta 'cta deletearchive
--user <eos::wfe::username>
--group <eos::wfe::rgroupname>
--id <eos::wfe::fxattr:sys.archiveFileId> 1>&2'" .
```

Ensure that the EOS workflow engine is switched on:

```
# eos space config default space.wfe=on
# eos space status default
```

The workflow configuration can be tested by manually triggering the EOS workflows:

```
# eos file workflow <filename> default closew
# eos file workflow <filename> default prepare
# eos file workflow <filename> default delete
```

## 2.12 Removing CTA

To remove the CTA Front-end and associated packages:

```
# yum remove oracle-instantclient12.1-basic oracle-instantclient-tnsnames.ora \
    oracle-instantclient12.1-devel oracle-instantclient12.1-meta cta-lib \
    cta-frontend protobuf3 protobuf3-compiler cryptopp libcephfs1 ceph-common \
    python-cephfs
```

## 2.13 To Do

### 2.13.1 RPM and Configuration File Names

We should decide on uniform naming conventions:

CTA packages sometimes have `utils` in the name

(`cta-catalogueutils-0-116192gitelfedfea.el7.cern.x86_64.rpm`,

`cta-mediachangerutils-0-116192gitelfedfea.el7.cern.x86_64.rpm`)

and sometimes have `-tools`

(`cta-objectstore-tools-0-116192gitelfedfea.el7.cern.x86_64.rpm`).

CTA configuration files sometimes use underscore

(`/etc/cta/cta_catalogue_db.conf.example`)

and sometimes use hyphen (`/etc/cta/cta-frontend.conf`).

### 2.13.2 Describe properly RPM dependencies (in the `.spec` files).

- `protobuf3` package is needed
- Ceph packages are needed
- `oracle-instantclient12.1-meta` is needed (has Provides declarations)

### 2.13.3 `ctacli.sh` contains comment: `<FQDN>:<TCPPort>` yet the value mentioned does not have the domain specified.

Can aliases (in general) be used (such as: `ctafrontend.cern.ch`)? In which circumstances YES and NO?

## 2.13.4 PLEASE make sure the 'status=' string is ALWAYS on the same position in the cta-taped logs.

Either at the end or immediately after the 'MSG=' key. Example of 3 (!!!) different positions within 2 hours:

```
2017-04-13T14:32:59.548438+02:00 cta-taped[250]: LVL="Info" TID="250" MSG="Tape
session finished" TPVID="I08155" mountType="Retrieve" mountTime="21.423454"
positionTime="8.541020" waitInstructionsTime="0.141429"
waitFreeMemoryTime="0.000004" waitDataTime="0.000000"
waitReportingTime="0.000427" checksummingTime="0.000000"
readWriteTime="0.010911" flushTime="0.000000" unloadTime="23.826163"
unmountTime="5.488744" encryptionControlTime="0.003776" transferTime="0.152771"
totalTime="59.295118" deliveryTime="30.183826" drainingTime="0.000000"
dataVolume="420" filesCount="1" headerVolume="480"
payloadTransferSpeedMBps="0.000007" driveTransferSpeedMBps="0.000015"
status="success"
2017-04-13T14:43:56.399355+02:00 cta-taped[250]: LVL="Info" TID="250" MSG="Tape
session finished" TPVID="I08155" mountType="Archive" status="success"
mountTime="15.523610" positionTime="8.509446" waitInstructionsTime="0.151270"
waitFreeMemoryTime="0.000000" waitDataTime="0.003547"
waitReportingTime="0.000239" checksummingTime="0.912911"
readWriteTime="7.656642" flushTime="10.106426" unloadTime="34.299974"
unmountTime="5.494463" encryptionControlTime="0.004026"
transferTime="18.831035" totalTime="82.511620" deliveryTime="82.511620"
drainingTime="0.000000" dataVolume="1048576000" filesCount="1"
headerVolume="480" payloadTransferSpeedMBps="12.708222"
driveTransferSpeedMBps="12.708228"
2017-04-13T15:56:48.411636+02:00 cta-taped[250]: LVL="Info" TID="250" MSG="Tape
session finished" TPVID="I08155" mountType="Archive" Error_tapeUnload="1"
Error_tapeMountForWrite="1" status="failure" mountTime="0.000000"
positionTime="0.000000" waitInstructionsTime="0.152993"
waitFreeMemoryTime="0.000000" waitDataTime="0.000000"
waitReportingTime="0.000000" checksummingTime="0.000000"
readWriteTime="0.000000" flushTime="0.000000" unloadTime="0.000000"
unmountTime="0.000000" encryptionControlTime="0.593565" transferTime="0.152993"
totalTime="61.499234" deliveryTime="61.499234" drainingTime="0.000000"
dataVolume="0" filesCount="0" headerVolume="0"
payloadTransferSpeedMBps="0.000000" driveTransferSpeedMBps="0.000000"
```

This REALLY helps a human to read the logs! I would propose to simply order all keys alphabetically and that's it.

## Chapter 3

# Start CTA

Documentation should contain "Hello world" example:

That is how to migrate one files to tape and how to get it back. This should include:

- declaration of a library in CTA catalogue
- declaration of tape pools and a tape
- putting file into EOS
- checking that the file is there
- list pending archives
- see tape mounted (like castor "tpstat" command)
- show the "m" bit to confirm the file is on tape
- remove disk copy on EOS
- retrieve file from tape
- see tape mounted (like castor "tpstat" command)
- show a disk copy of a file somewhere on disk (example: <FST server:/<FST path>/00000000/00000007)

### 3.1 Debugging Authentication Problems

Force the `cta` command line tool to use Kerberos authentication:

```
export XrdSecPROTOCOL=krb5
```

Get debugging info if KDC authentication does not work:

```
export XrdSecDEBUG=1
```

### 3.2 Trying out some commands

Now CTA should be up and running and we can try out some commands from the `cmdline` directory:

```

$ cta ll add -n IBMJD3 -m "IBM lib 3"
$ cta tp add -n cms_raw_08 -p 15 -m "CMS raw"
$ cta ta add -v G12345 -l IBMJD3 -t cms_raw_08 -c 10000 -m "Tape"
$ cta sc add -n single -i 2 -c 1 -m "A single copy class"
$ cta ar add -s single -c 1 -t cms_raw_08 -m "Route to cms"
$ cta mkdir /cta
$ cta ssc /cta single
$ cta csc /cta
$ cta ssc /cta single
$ cta ls /
$ cta a "eos://eos/passwd" /cta/file5

```

See Chapter 5 for a complete reference for the user commands.

## 3.3 Running tapeserverd

### 3.3.1 Tape labelling

Tapes can be labelled with the following commands (depends on the tape library and the tape drive types):

```

$ echo "VOL1V31001" CASTOR
$ mtx -f /dev/smc load 1 0
$ dd if=label.file of=/dev/nst0 bs=80 count=1
$ mt -f /dev/nst0 rewind
$ mtx -f /dev/smc unload 1 0

```

### 3.3.2 Preparing the CTA environment for migrations

The library and tapes need to be registered for the migrations (use appropriate names for your setup):

```

# create library
$ cta logicallibrary add --name VLSTK --comment "ctasystest"
# create tape pool
$ cta tapepool add --name ctasystest --partialtapesnumber 5 --encrypted false --
# add tape to the tapepool and library
$ cta tape add --logicallibrary VLSTK --tapepool ctasystest --capacity 1000000000
--comment "ctasystest" --vid V31001 --disabled false --full false
# create storage class
$ cta storageclass add --instance root --name ctaStorageClass --copynb 1 --comment
# create archive route
$ cta archiveroute add --instance root --storageclass ctaStorageClass --copynb 1
--tapepool ctasystest --comment "ctasystest"
# create mount policy
$ cta mountpolicy add --name root --archivepriority 1 --minarchiverequeststage 1 \
--retrievepriority 1 --minretrieverequeststage 1 --maxdrivesallowed 1 --comment "ct
# create requester mount rule
$ cta requestermountrule add --instance root --name root --mountpolicy ctasyste
--comment "ctasystest"

```

## Chapter 4

# Administrator Commands

These are the commands that only administrators are allowed to use to operate CTA. For each command there is a short version and a long one, example: `op/operator`. Subcommands (`add`, `rm`, `ls`, `ch`, `reclaim`) do not have short versions.

- `cta ad/admin`
- `cta ah/adminhost`<sup>1</sup>
- `cta us/user`<sup>2</sup>
- `cta tp/tapepool`<sup>3</sup>
- `cta ar/archiveroute`
- `cta ll/logicallibrary`
- `cta ta/tape`
- `cta sc/storageclass`<sup>4</sup>
- `cta lpa/listpendingarchives`
- `cta lpr/listpendingretrieves`
- `cta lds/listdrivestates`<sup>5</sup>

The detailed list of the commands with their subcommands and parameters follows.

### 4.1 `cta admin`

This command is used to manage the administrators of the system. These are special users allowed to perform the admin commands described in this section.

---

<sup>1</sup>hostgroups also? not for the prototype

<sup>2</sup>or maybe power users? or egroups? will see later, for the moment just users

<sup>3</sup>listing the tapepools should include also stats such as total number of tapes and number of free tapes to help operator

<sup>4</sup>storage classes should be as static as possible, no change nor deletion if there is at least 1 directory using it

<sup>5</sup>more or less the “drive” part of the current `showqueues -x`

<i>Command &amp; Subcommand</i>	<i>Parameters</i>
	--uid/-u <uid>
cta ad/admin add	--gid/-g <gid>
	--comment/-m <"comment">
cta ad/admin ch	--uid/-u <uid>
	--gid/-g <gid>
	--comment/-m <"comment">
cta ad/admin rm	--uid/-u <uid>
	--gid/-g <gid>
cta ad/admin ls	

## 4.2 cta adminhost

This command is used to manage the administrator hosts. These are specific machines from which administrators may issue the admin commands described in this section.

<i>Command &amp; Subcommand</i>	<i>Parameters</i>
cta ah/adminhost add	--name/-n <host_name>
	--comment/-m <"comment">
cta ah/adminhost ch	--name/-n <host_name>
	--comment/-m <"comment">
cta ah/adminhost rm	--name/-n <host_name>
cta ah/adminhost ls	

## 4.3 cta user

This command is used to manage the “normal” users of CTA. These are typically physicists and experiment data storage managers, who are able to perform only the actions described in the next section (*archive, retrieve, etc.*).

<i>Command &amp; Subcommand</i>	<i>Parameters</i>
	--uid/-u <uid>
cta us/user add	--gid/-g <gid>
	--comment/-m <"comment">
cta us/user ch	--uid/-u <uid>
	--gid/-g <gid>
	--comment/-m <"comment">
cta us/user rm	--uid/-u <uid>
	--gid/-g <gid>
cta us/user ls	

## 4.4 cta tapepool

This command is used to manage the tape pools, which are logical sets of tapes. These are useful to manage the life cycle of tapes tolabel → supply → user pool → erase → tolabel.

<i>Command &amp; Subcommand</i>	<i>Parameters</i>
	--name/-n <tapepool_name>
cta tp/tapepool add	--partialtapesnumber/-p <number_of_partial_tapes> --comment/-m <"comment">
	--name/-n <tapepool_name>
cta tp/tapepool ch	--partialtapesnumber/-p <number_of_partial_tapes> --comment/-m <"comment">
cta tp/tapepool rm	--name/-n <tapepool_name>
cta tp/tapepool ls	

## 4.5 cta archiveroute

This command is used to manage the archive routes, which are the policies linking name space entries to tape pools.

<i>Command &amp; Subcommand</i>	<i>Parameters</i>
	--storageclass/-s <storage_class_name>
cta ar/archiveroute add	--copynb/-c <copy_number> --tapepool/-t <tapepool_name> --comment/-m <"comment">
	--storageclass/-s <storage_class_name>
cta ar/archiveroute ch	--copynb/-c <copy_number> --tapepool/-t <tapepool_name> --comment/-m <"comment">
cta ar/archiveroute rm	--storageclass/-s <storage_class_name> --copynb/-c <copy_number>
cta ar/archiveroute ls	

## 4.6 cta logicallibrary

This command is used to manage the logical libraries. These are logical groupings of tapes and drives based on physical location and tape drive capabilities. Basically a tape can be accessed by a drive if it is in the same physical library and if the drive is capable of reading or writing the tape, in that case we typically have that that tape and that drive are in the same logical library.

<i>Command &amp; Subcommand</i>	<i>Parameters</i>
cta ll/logicallibrary add	--name/-n <logical_library_name> --comment/-m <"comment">
cta ll/logicallibrary ch	--name/-n <logical_library_name> --comment/-m <"comment">
cta ll/logicallibrary rm	--name/-n <logical_library_name>
cta ll/logicallibrary ls	

## 4.7 cta tape

This command is used to manage the tapes. These are the physical containers of data.



<i>Command &amp; Subcommand</i>	<i>Parameters</i>
cta ta/tape add	--vid/-v <vid>
	--logicallibrary/-l <logical_library_name>
	--tapepool/-t <tapepool_name>
	--capacity/-c <capacity_in_bytes> --comment/-m <"comment">
cta ta/tape ch	--vid/-v <vid>
	--logicallibrary/-l <logical_library_name>
	--tapepool/-t <tapepool_name>
	--capacity/-c <capacity_in_bytes> --comment/-m <"comment">
cta ta/tape rm	--vid/-v <vid>
cta ta/tape reclaim	--vid/-v <vid>
cta ta/tape ls	

## 4.8 cta storageclass

This command is used to manage the storage classes. These can be associated with CTA directories and they determine the number of tape copies the files in the directory should have.

<i>Command &amp; Subcommand</i>	<i>Parameters</i>
cta sc/storageclass add	--name/-n <storage_class_name>
	--copynb/-c <number_of_tape_copies>
	--comment/-m <"comment">
cta sc/storageclass ch	--name/-n <storage_class_name>
	--copynb/-c <number_of_tape_copies>
	--comment/-m <"comment">
cta sc/storageclass rm	--name/-n <storage_class_name>
cta sc/storageclass ls	

## 4.9 cta list...

This set of commands is used to list the pending archives and retrieves as well as the state of each tape drive (its status –up or down–, its contents, etc.).

<i>Command</i>	<i>Parameters</i>
cta lpa/listpendingarchives	--tapepool/-t <tapepool_name>
cta lpr/listpendingretrieves	--vid/-v <vid>
cta lds/listdrivestates	

## 4.10 Questions administrators need to be able to answer

Questions that administrators need to be able to answer easily using commands above (not necessarily for the prototype, but to keep in mind):

1. Why is data not going to tape?
2. Why is data not coming out of tapes?
3. Which user is responsible for system overload?

## Chapter 5

# User Commands

For most commands there is a short version and a long one. Due to the limited number of USER commands it is not convenient (nor intuitive) to use subcommands here (anyway it could be applied only to storage classes).

- `cta lsc/liststorageclass`<sup>1</sup>
- `cta ssc/setstorageclass <dirpath> <storage_class_name>`
- `cta csc/clearstorageclass <dirpath>`
- `cta mkdir <dirpath>`
- `cta chown <uid> <gid> <dirpath>`<sup>2</sup>
- `cta rmdir <dirpath>`
- `cta ls <dirpath>`
- `cta a/archive <src1> [<src2> [<src3> [...]]] <dst>`
- `cta r/retrieve <src1> [<src2> [<src3> [...]]] <dst>`
- `cta da/deletearchive <dst>`<sup>3</sup>
- `cta cr/cancelretrieve <dst>`<sup>4</sup>

---

<sup>1</sup>this command might seem a duplicate of the corresponding admin command but it actually shows a subset of fields (name and number of copies)

<sup>2</sup>we may want to add `chmod` later on

<sup>3</sup>this works both on ongoing and finished archives, that is why it's called "delete"

<sup>4</sup>this clearly works only on ongoing retrieves, obviously does not delete destination files, that's why it's called "cancel"

# Appendix A

## Install a Local EOS Instance

This Appendix describes how to install a local EOS instance.

The steps described below are automated by Steve's `reinstalleos_eosctatape` script<sup>1</sup>.

### A.1 Install XRoot and EOS RPMs

The yum repos, priorities and version lock list should be configured as described in Sections 2.2 and 2.3. Next, install the RPMs:

```
# yum install eos-client eos-server xrootd-client xrootd-debuginfo xrootd-server
  heimdal-server heimdal-workstation
```

Now the list of installed EOS and XRoot RPMs should look something like this:

```
# rpm -qa | egrep 'eos|xrootd|heimdal' | sort
eos-client-4.1.11-20170118171644git24cd94a.el7.x86_64
eos-server-4.1.11-20170118171644git24cd94a.el7.x86_64
heimdal-libs-1.6.0-0.9.20140621gita5adc06.el7.x86_64
heimdal-server-1.6.0-0.9.20140621gita5adc06.el7.x86_64
heimdal-workstation-1.6.0-0.9.20140621gita5adc06.el7.x86_64
libmicrohttpd-0.9.38-eos.wves.el7.cern.x86_64
xrootd-4.4.1-1.el7.x86_64
xrootd-client-4.4.1-1.el7.x86_64
xrootd-client-libs-4.4.1-1.el7.x86_64
xrootd-debuginfo-4.4.1-1.el7.x86_64
xrootd-libs-4.4.1-1.el7.x86_64
xrootd-selinux-4.4.1-1.el7.noarch
xrootd-server-4.4.1-1.el7.x86_64
xrootd-server-libs-4.4.1-1.el7.x86_64
```

### A.2 Create a Kerberos keytab file

The instructions below are for machines on structured cabling which have a fixed IP address. Machines without a fixed IP address—including virtual machines or Docker pods—are not recognised by the CERN Kerberos servers. To install a local EOS instance in this case, we need to run our own KDC server. See Section A.11 for more information on how to do this.

The EOS `mgm` in the XRoot daemon authenticates users using a key from the Kerberos keytab.

<sup>1</sup>`reinstalleos_eosctatape` calls `create_eos_and_taped_keytabs` to create the Simple Shared Secret (SSS) keys for the EOS instance and the tape server.

If `/etc/krb5.keytab` does not already exist, we need to create a new EOS service principal in the `kdc`, and install the key in the keytab.

In the case of machines at CERN with a fixed IP address, there is a package for this<sup>2</sup>:

```
# yum install cern-get-keytab
```

This can be used to create the host and EOS service kerberos keys and host and CTA service key as follows:

```
# rm -f /etc/krb5.keytab
# cern-get-keytab --service eosdev --force
# rm /etc/krb5.keytab
# cern-get-keytab --service cta --force
# echo -e "read_kt /etc/krb5.keytab\nlist\nquit" | ktutil
```

Extract the key for the EOS principal to a new keytab, which should be owned by user `daemon` so that it is readable by the `mgm`:

```
# rm -f /etc/krb5.keytab.eosdev
# cp /etc/krb5.keytab /etc/krb5.keytab.eosdev
# chown daemon:daemon /etc/krb5.keytab.eosdev
# echo -e "read_kt /etc/krb5.keytab.eosdev\nlist\nquit" | ktutil
```

Note that this process re-generates a new version of every other key for this host, which might require the client users to `kdestroy` their corresponding tickets in caches.

Note that the `ktutil` utility can also be run in interactive mode. Here is a more verbose way to extract the keys:

```
# ktutil
ktutil: read_kt /etc/krb5.keytab
ktutil: list
slot KVNO Principal
-----
1 14 devbox$@CERN.CH
2 14 devbox$@CERN.CH
3 14 devbox$@CERN.CH
4 14 eoscta/devbox.cern.ch@CERN.CH
5 14 eoscta/devbox.cern.ch@CERN.CH
6 14 eoscta/devbox.cern.ch@CERN.CH
ktutil: delent 1
ktutil: delent 1
ktutil: delent 1
ktutil: list
slot KVNO Principal
-----
1 14 eoscta/devbox.cern.ch@CERN.CH
2 14 eoscta/devbox.cern.ch@CERN.CH
3 14 eoscta/devbox.cern.ch@CERN.CH
ktutil: write_kt /etc/krb5.keytab.eos
ktutil: quit
```

---

<sup>2</sup>For more details, see <http://linux.web.cern.ch/linux/docs/kerberos-access.shtml>

## A.3 Create a Simple Shared Secret keytab file

The `cta` user must exist before creating the SSS keys. This should be done automatically when the CTA RPMs are installed.

The EOS `mgm` and `fst` nodes authenticate to each other using the Simple Shared Secret (SSS) mechanism. The `xrdsssadmin` tool is used to create a keytab containing the Tape Server key and EOS instance key:

```
# xrdsssadmin -k cta-taped -u cta -g cta add /etc/eos.keytab
xrdsssadmin: Keyfile '/etc/eos.keytab' does not exist. Create it? (y | n): y
xrdsssadmin: 1 key out of 1 kept (0 expired).
# xrdsssadmin -k eosdev -u daemon -g daemon add /etc/eos.keytab
xrdsssadmin: 2 keys out of 2 kept (0 expired).
# chown daemon:daemon /etc/eos.keytab
```

Then create a second keytab containing only the Tape Server key.

```
# cp /etc/eos.keytab /etc/cta-taped.keytab
# xrdsssadmin -k eosdev del /etc/cta-taped.keytab
xrdsssadmin: 1 key out of 2 kept (0 expired).
# chown cta:cta /etc/cta-taped.keytab
```

Note that XRoot clients which parse a multi-line SSS keytab file use the last line in the file as their key.

## A.4 Configure EOS `sysconfig`

It looks like there is a new config file `/etc/eos.systemd.conf` which overlaps with (or supersedes?) `/etc/sysconfig/eos_env`. Check which of these is the correct one to use.

Create the `/etc/sysconfig/eos_env` file based on the example installed by the `eos-server` RPM:

```
# cp /etc/sysconfig/eos_env.example /etc/sysconfig/eos_env
```

Edit the file and comment out the lines for the SYNC and FED daemons. This reduces the XRoot daemon roles to the minimum set of three (`mq`, `mgm` and `fst`). *i.e.*, there will be three XRoot daemons running for EOS on the local development box.

```
#sync=sync
#fed=fed
```

Continue editing the file to set the name of the EOS instance and replace all the hostnames with the fully-qualified domain name of the local development box. The resulting hostname entries should look something like the following (where `devbox.cern.ch` should be replaced with the FQDN of the development box where EOS is being installed):

```
EOS_INSTANCE_NAME=eosdev
EOS_GEOTAG="devbox.cern.ch"
EOS_MGM_HOST=devbox.cern.ch
```

```
EOS_MGM_HOST_TARGET=devbox.cern.ch
EOS_MGM_MASTER1=devbox.cern.ch
EOS_MGM_MASTER2=devbox.cern.ch
EOS_MGM_ALIAS=devbox.cern.ch
EOS_MAIL_CC="your.name@cern.ch"
```

## A.5 Configure XRoot mgm

SSS will be used for communication between the CTA Tape Server and EOS. In a production system, Kerberos is also required for communication between the user and EOS.

This is configured in the `/etc/xrd.cf.mgm` file which was installed by the `eos-server` RPM. Edit this file and comment out the UNIX- and GSI-based authentication mechanisms and leave only Simple Shared Secret (SSS) and Kerberos (KRB) authentication enabled. Then edit the Kerberos protocol to use the EOS-specific kerberos keytab file as shown below:

```
# UNIX authentication
#sec.protocol unix
# SSS authentication
sec.protocol sss -c /etc/eos.keytab -s /etc/eos.keytab
# KRB authentication
sec.protocol krb5 /etc/krb5.keytab.eosdev eosdev/<host>@CERN.CH
# GSI authentication
#sec.protocol gsi -crl:0 -cert:/etc/grid-security/daemon/hostcert.pem ...
```

Set the order of authentication mechanisms on all hosts to be Kerberos followed by Simple Shared Secret:

```
#sec.protbinding localhost.localdomain unix sss
#sec.protbinding localhost unix sss
sec.protbinding * only krb5 sss
```

Configure the EOS mgm instance name:

```
mgmofs.broker root://devbox.cern.ch:1097//eos/
mgmofs.instance eosdev
mgmofs.cfgredishost devbox.cern.ch
```

Finally, ensure that the EOS namespace plugin will be loaded:

```
#-----
# Set the namespace plugin implementation
#-----
mgmofs.nslib /usr/lib64/libEosNsInMemory.so
```

Create a local directory for EOS mgm. It seems this is not actually used for anything, but creating it suppresses a spurious error in the EOS logs.

```
# mkdir -p /mgm
# chown daemon:daemon /mgm/
```

## A.6 Configure XRoot fst

In `/etc/xrd.cf.fst`, replace references to `localhost` with the FQDN of the local development box:

```
all.manager devbox.cern.ch 2131
fstofs.broker root://devbox.cern.ch:1097//eos/
```

Create a local directory to be used to store files by the EOS `fst`:

```
# mkdir -p /fst
# chown daemon:daemon /fst
```

## A.7 Start XRoot daemons

Start the XRoot daemons that will run the EOS `mgm`, `mq` and `fst` plugins:

```
# systemctl start eos
```

The logs for the XRoot daemons will be created under `/var/log/eos/fst`, `/var/log/eos/mgm` and `/var/log/eos/mq`.

## A.8 Enable EOS authentication mechanisms

Enable the Kerberos and Simple Shared Secret authentication mechanisms within EOS (as opposed to XRoot):

```
# eos vid enable sss
# eos vid enable krb5
```

## A.9 Configure the default EOS space

Register the directory for the default EOS space:

```
# echo 'EOS_MGM_URL=root://devbox.cern.ch' > /etc/sysconfig/eos
# eosfstregister -r /fst default:1
#####
# <eosfstregister> v1.0.0
#####
/fst : uuid=abae0ba6-ffc5-491f-9ef3-291f291493af fsid=undef
success: mapped 'abae0ba6-ffc5-491f-9ef3-291f291493af' <=> fsid=1
```

`/usr/sbin/eosfsregister` is a convenience script for registering an EOS `fst` node with an EOS `mgm` node. It parses the deprecated configuration file `/etc/sysconfig/eos` to determine the location of the EOS `mgm` node. For the time being, we need to create this file even though it is not used by EOS directly.

`eosfsregister` is not maintained by the EOS developers, and the same work can be done by standard EOS commands. We will modify our CTA install/configure/setup scripts to use the standard commands instead of `eosfsregister`.

Enable the WorkFlow Engine (WFE) for the default EOS space, enable default EOS space, bring the EOS `fst` node on-line:

```
# eos space config default space.wfe=on
# eos space set default on
# eos node set devbox.cern.ch on
```

Wait until the EOS disk filesystem comes online:

```
# eos fs ls /fst | grep online
devbox.cern.ch (1095) 1 /fst default.0 ...evbox.cern.ch booted rw nodrain online
unknown raid
```

Create the `/eos` directory within the EOS namespace, map it to the EOS `default` space, set the number of replicas to 1:

```
# eos attr -r set default=replica /eos
# eos attr -r set sys.forced.nstripes=1 /eos
```

## A.10 Test the EOS Installation

Perform a simple write and read test of the new EOS installation:

```
# eos mkdir /eos/users/test
# eos chmod 777 /eos/users/test
# xrdcp /etc/motd root://devbox.cern.ch//eos/users/test/
# xrdcp root://devbox.cern.ch//eos/users/test/motd /tmp/eostest
# diff /tmp/eostest /etc/motd
```

## A.11 Installing EOS in Docker

The instructions above are for installing EOS on real hardware. To install EOS in a Docker pod, note that you must:

1. Create a Docker network to allow DNS/reverse DNS queries to work.
2. Use a privileged Docker container and mount the `cgroup` to allow `systemd` to run.

To run `systemd` on Docker host `bandersnatch` on network `wonderland`, the command would be:

```
$ sudo docker network create wonderland
$ sudo docker run --net=wonderland --name bandersnatch --hostname
bandersnatch.wonderland --privileged -v /sys/fs/cgroup:/sys/fs/cgroup:ro -d
ctatest /usr/sbin/init
```

See also the [eos-docker](#) GitLab project, prepared by the IT-ST-AD section.



The EOS `mgm` startup script checks if we are running under `systemd`, but the test it performs does not work inside a Docker container, as (a) `pidof` is part of `sysvtools-init`, which is not installed and (b) the `systemd` process is started as `init` (a symbolic link to `systemd`):

```
/usr/sbin/pidof systemd
```

This has been reported to the EOS team to fix. In the meantime, here is a hack to work around the problem:

```
# yum install -y sysvinit-tools
# ln -s /usr/bin/sleep /tmp/systemd
# /tmp/systemd 1000 &
# systemctl start eos
```

## Appendix B

# Install the Virtual Tape Library (mhVTL)

Normally CTA will be used with physical Tape Libraries. If a physical Tape Library is not available—for example during development and testing—a Virtual Tape Library (VTL) can be used instead. A VTL is a data storage virtualization technology which presents a storage component (usually hard disk storage) as tape libraries or tape drives.

This Appendix describes how to install Mark Harvey’s Virtual Tape Library **mhVTL**.

### B.1 Install mhVTL

Install the RPMs for the mhVTL software and kernel module:

```
# yum-config-manager --enable castor
# yum install -y mhvtl-utils kmod-mhvtl
```

Also install the command-line utilities to access the virtual tape drives. `mt` is used to control magnetic tape drives (rewinding, ejecting, skipping files and blocks, etc.). `mtx` is used to control the robotic mechanism in tape libraries:

```
# yum install -y mt-st mtx
```

### B.2 Configure mhVTL

Create the configuration directory `/etc/mhvtl`. The main configuration file is `/etc/mhvtl/mhvtl.conf`:

```
# Home directory for configuration files
MHVTL_CONFIG_PATH=/etc/mhvtl

# Default media capacity in Mb
CAPACITY=1000

# Verbosity level [0|1|2|3]
VERBOSE=1

# Set kernel module debugging [0|1]
VTL_DEBUG=0
```

Configure tape libraries and drives as virtual SCSI devices in `/etc/mhvtl/device.conf`:

```

VERSION: 5

Library: 10 CHANNEL: 00 TARGET: 00 LUN: 00
Vendor identification: STK
Product identification: VLSTK
Unit serial number: VLSTK
NAA: 30:22:33:44:ab:00:08:00

Drive: 11 CHANNEL: 00 TARGET: 1 LUN: 00
Library ID: 10 Slot: 1
Vendor identification: STK
Product identification: T10000B
Unit serial number: VDSTK1
NAA: 30:22:33:44:ab:00:09:00
Compression: factor 1 enabled 1
Compression type: zlib

Drive: 12 CHANNEL: 00 TARGET: 2 LUN: 00
Library ID: 10 Slot: 2
Vendor identification: STK
Product identification: T10000B
Unit serial number: VDSTK2
NAA: 30:22:33:44:ab:00:09:00
Compression: factor 1 enabled 1
Compression type: zlib

```

Next, create a configuration file per library to map the drives to the library and define the other elements of the library. For the example above, the configuration file will be `/etc/mhvtl/library_contents.10`. This library is defined with two drives, one robot picker and one Media Access Port (MAP). There are eight tape slots containing five tapes:

```

Drive 1:
Drive 2:

Picker 1:

MAP 1:

Slot 1: V31001TA
Slot 2: V31002TA
Slot 3: V31003TA
Slot 4: V31004TA
Slot 5: V31005TA
Slot 6:
Slot 7:
Slot 8:

```

### B.2.1 udev rules

Create `/etc/udev/rules.d/00-cta.rules`. Add rules to create symbolic links to the first tape drive and the SCSI media changer device (IBM tape library specific):

```

# Create symlink /dev/tape pointing to the (first) tape drive
SUBSYSTEM=="scsi_tape", KERNEL=="nst0", SYMLINK:="tape"

# Create symlink /dev/smc pointing to the SCSI media changer
SUBSYSTEM=="scsi_generic", KERNEL=="sg*", ATTRS{type}=="8", SYMLINK:="smc"

```

Also add rules for permissions for the SCSI devices. This example sets global read/write permissions to all devices; in production, you should configure more nuanced rules based on the `st` group. This group is created automatically by the castor RPMs:

```
# Set the permissions and group of the tape devices
KERNEL=="nst*", MODE:="0666"
KERNEL=="st*", MODE:="0666"
KERNEL=="sg*", MODE:="0666"
```

To apply the rules without rebooting:

```
# sudo udevadm trigger
```

## B.3 Start mhVTL

Start the mhVTL service:

```
# systemctl start mhvtl
```

The service requires the mhVTL kernel module `mhvtl.ko`, which should be loaded automatically by `systemd`.

After starting the mhVTL daemon, the files for the virtual tapes will be created automatically:

```
# ls -lRF /opt/mhvtl/
/opt/mhvtl/:
total 0
drwxrwx---. 7 vtl vtl 86 May 15 09:58 10/

/opt/mhvtl/10:
total 0
drwxrwx---. 2 vtl vtl 42 May 15 09:58 V31001TA/
drwxrwx---. 2 vtl vtl 42 May 15 09:58 V31002TA/
drwxrwx---. 2 vtl vtl 42 May 15 09:58 V31003TA/
drwxrwx---. 2 vtl vtl 42 May 15 09:58 V31004TA/
drwxrwx---. 2 vtl vtl 42 May 15 09:58 V31005TA/

/opt/mhvtl/10/V31001TA:
total 4
-rw-rw----. 1 vtl vtl    0 May 15 09:58 data
-rw-rw----. 1 vtl vtl    0 May 15 09:58 indx
-rw-rw----. 1 vtl vtl 1536 May 15 09:58 meta
...
```

Validate that the Virtual Tape Library is working:

```
# mtx -f /dev/smc inquiry
Product Type: Medium Changer
Vendor ID: 'STK'
Product ID: 'VLSTK'
Revision: '0105'
Attached Changer API: No
```

```
# mtx -f /dev/smc status
Storage Changer /dev/smc:2 Drives, 9 Slots ( 1 Import/Export )
Data Transfer Element 0:Empty
Data Transfer Element 1:Empty
Storage Element 1:Full :VolumeTag=V31001TA
Storage Element 2:Full :VolumeTag=V31002TA
Storage Element 3:Full :VolumeTag=V31003TA
Storage Element 4:Full :VolumeTag=V31004TA
Storage Element 5:Full :VolumeTag=V31005TA
Storage Element 6:Empty
Storage Element 7:Empty
Storage Element 8:Empty
Storage Element 9 IMPORT/EXPORT:Empty
```

## B.4 Virtual Tape Library Commands

Look for the medium changer (mediumx):

```
# lsscsi -g | grep mediumx
```

My medium changer is /dev/sg3, yours may well be something else:

```
# mtx -f /dev/sg3 status
```

Mount the tape in storage element 1 into data transfer element 0:

```
# mtx -f /dev/sg3 load 1 0
# mtx -f /dev/sg3 status
```

Check the status of the drive:

```
# mt -f /dev/nst0 status
```

Rewind the tape:

```
# mt -f /dev/nst0 rewind
```

Read the first block from the tape:

```
# dd if=/dev/nst0 bs=262144 count=1
```

Successively repeating the above command will read out subsequent blocks from the tape in the following format:

- VOL1 label (80 byte block)
- HDR1 label (80 byte block)
- HDR2 label (80 byte block)

- UHL1 label (80 byte block)
- End of tape file (0 records out)
- First block of user's file (256 KiB)
- Second block of user's file (256 KiB)
- ...
- Last block of user's file ( $\leq 256$  KiB)
- End of tape file (0 records out)
- EOF1 label (80 byte block)
- EOF2 label (80 byte block)
- UTL1 label (80 byte block)
- End of tape file (0 records out)

Rewind the tape:

```
# mt -f /dev/nst0 rewind
```

Eject the tape:

```
# mt -f /dev/nst0 eject
```

Unload the tape to storage element 1 from data transfer element 0:

```
# mtx -f /dev/sg3 unload 1 0
```

# Appendix C

## Tape Format

### C.1 Overview

CTA uses the same AUL file format as CASTOR<sup>1</sup>. This format is based on [ANSI INCITS 27-1987](#) and is described in detail on the [Tape Labels, ANSI and IBM](#) web page (last updated in 2008).

The AUL format has the following descriptors:

- Volume Label (VOL1)
- Header Blocks: Headers (HDR1, HDR2) and User Header Labels (UHL1)
- Trailer Blocks: User Trailer Labels (UTL1)<sup>2</sup>

Each of these descriptor labels is contained in an 80-byte tape block of ASCII text. Empty bytes are stored as spaces (0x20). The label descriptor must begin with the 4-byte identifier. Labels are terminated by a file mark: Tape Mark (TM) or End of File (EOF).

Table C.1: AUL label format

VOL1	HDR1	HDR2	UHL1	TM	DATA	TM	EOF1	EOF2	UTL1	TM
one data file										

Volumes that have just been initialised contain no data records, just a single ‘header label group’:

Table C.2: AUL prelabeled tape with one HDR1

VOL1	HDR1(PRELABEL)	TM
------	----------------	----

### C.2 Volume Label (VOL<sub>n</sub>)

The very first label record on a labelled volume is VOL1. If this label is incorrect, you will not advance at all.

<sup>1</sup>CASTOR used several file formats over time, but by 2013, only the AUL format was in use.

<sup>2</sup>The UHLs and UTLs are defined in ANSI X 3.27. The general description of the ANSI fields was documented in IBM’s z/OS documentation.

The format is shown in Table C.3. Example for beginning of the tape:

```

00000000  56 4f 4c 31 56 35 32 30 30 31 20 20 20 20 20 20 |VOL1V52001      |
00000010  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 |              |
00000020  20 20 20 20 20 43 41 53 54 4f 52 20 20 20 20 20 |          CASTOR  |
00000030  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 |              |
00000040  20 20 20 20 20 20 20 20 20 20 20 20 20 20 33 33 |                      3|

```

Table C.3: The structure of the volume label

VOL1			
Bytes	Length	Offset	Content
0-3	4	0x00	Volume label indicator: the characters <b>VOL1</b>
4-9	6	0x04	Volume serial number (VSN) (e.g., “AB1234”)
10	1	0x0A	Accessibility (left as empty space)
11-23	13	0x0B	Reserved (spaces)
24-36	13	0x18	Implementation identifier (left as empty spaces)
37-50	14	0x25	Owner identifier (the string “CASTOR” or STAGESUPERUSER name, padded with spaces)
51-78	28	0x33	Reserved (spaces)
79	1	0x4F	Label standard level (1, 3 and 4 are listed as valid in IBM’s documentation. CASTOR uses ASCII ‘3’)

### C.3 Header Label (HDR<sub>n</sub>)

HDR1 and HDR2 are normally found together at the beginning of a dataset.

The format for HDR1 is shown in Table C.4 and the format for HDR2 is shown in Table C.5. Example for the empty tape with PRELABEL and one HDR1:

```

00000000  56 4f 4c 31 56 35 32 30 30 31 20 20 20 20 20 20 |VOL1V52001      |
00000010  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 |              |
00000020  20 20 20 20 20 72 6f 6f 74 20 20 20 20 20 20 20 |          root    |
00000030  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 |              |
00000040  20 20 20 20 20 20 20 20 20 20 20 20 20 20 33 33 |                      3|
00000050  48 44 52 31 50 52 45 4c 41 42 45 4c 20 20 20 20 |HDR1PRELABEL    |
00000060  20 20 20 20 20 56 35 32 30 30 31 30 30 30 31 30 |          V5200100010|
00000070  30 30 31 30 30 30 31 30 30 30 31 33 32 33 34 30 |0010001000132340|
00000080  31 33 32 33 34 20 30 30 30 30 30 30 43 41 53 54 |13234 000000CAST|
00000090  4f 52 20 32 2e 31 2e 31 33 20 20 20 20 20 20 20 |OR 2.1.13       |

```

Example of HDR1 for the second file on the tape:

```

00000000  48 44 52 31 31 32 41 31 36 30 43 33 38 20 20 20 |HDR112A160C38   |
00000010  20 20 20 20 20 56 35 32 30 30 31 30 30 30 31 30 |          V5200100010|
00000020  30 30 32 30 30 30 31 30 30 30 31 32 30 34 31 30 |0020001000120410|
00000030  31 32 30 34 31 20 30 30 30 30 30 30 43 41 53 54 |12041 000000CAST|
00000040  4f 52 20 32 2e 31 2e 31 32 20 20 20 20 20 20 20 |OR 2.1.12       |

```

Example of HDR2 for the first file on the tape:

```

00000000  48 44 52 32 46 30 30 30 30 30 30 30 30 30 30 20 |HDR2F0000000000 |
00000010  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 |              |
00000010  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 |              |
00000030  20 20 30 30 20 20 20 20 20 20 20 20 20 20 20 20 |          00      |
00000040  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 |              |

```



Table C.4: The structure of the HDR1, EOF1 labels

HDR1, EOF1			
Bytes	Length	Offset	Content
0-3	4	0x00	Header label: the characters “HDR1 or EOF1”
4-20	17	0x04	File identifier: hexadecimal CASTOR NS file ID. <code>nsgetpath -x</code> can be used to find the CASTOR full path name. Aligned to left. In case of prelabeled tape ‘PRELABEL’ is used instead of file ID.
21-26	6	0x15	The volume serial number of the tape.
27-30	4	0x1B	File section number: a number (0001 to 9999) that indicates the order of the volume within the multivolume aggregate. This number is always 0001 for a single volume data set.
31-34	4	0x1F	File sequence number: a number that indicates the relative position of the data set within a multiple data set group (aggregate). CASTOR uses modulus for fseq by 10000
35-38	4	0x23	Generation number: ‘0001’ in CASTOR.
39-40	2	0x27	Version number of generation: ‘00’ in CASTOR.
41-46	6	0x29	Creation date: Date when allocation begins for creating the data set. The date format is <code>cyydd</code> , where: <code>c</code> = century (blank=19; 0=20; 1=21; etc.) <code>yy</code> = year (00-99) <code>ddd</code> = day (001-366)
47-52	6	0x2F	Expiration date: year and day of the year when the data set may be scratched or overwritten. The data is shown in the format <code>cyydd</code> . It is always advisable to set the expiration date when a volume is being initialised (‘prelabelled’) to be a date before the current date, so that writing to the tape is immediately possible.
53	1	0x35	Accessibility: a code indicating the security status of the data set and ‘space’ means no data set access protection.
54-60	6	0x36	Block count: This field in the trailer label shows the number of data blocks in the data set on the current volume. This field in the header label is always ‘000000’.
60-72	13	0x3C	System code of creating system: a unique code that identifies the system. CASTOR with CASTOR BASEVERSION number string.
73-79	7	0x49	Reserved

## C.4 User Header Label (UHL<sub>n</sub>)

The format for UHL1 is shown in Table C.6. Example for the second file on the tape:

```

00000000  55 48 4c 31 30 30 30 30 30 30 30 30 32 30 30 |UHL10000000000200|
00000010  30 30 32 36 32 31 34 34 30 30 30 30 32 36 32 31 |0026214400002621|
00000020  34 34 43 45 52 4e 20 20 20 20 4c 58 43 32 44 45 |44CERN      LXC2DE|
00000030  56 35 44 32 53 54 4b 20 20 20 20 20 54 31 30 30 |V5D2STK      T100|
00000040  30 30 42 20 58 59 5a 5a 59 5f 42 31 20 20 20 20 |00B XYZZY_B1    |

```

## C.5 Data Records

After a ‘header label group’, data records follow of any length and in any number. Eventually, an EOF will appear and then a ‘trailer label group’ is expected.

The data block size is configurable but in practice a block size of 256 KiB has been used everywhere.

Table C.5: The structure of the HDR2, EOF2 labels

HDR2, EOF2			
Bytes	Length	Offset	Content
0-3	4	0x00	Header label: the characters “HDR2 or EOF2”
4	1	0x04	Record format. An alphabetic character that indicates the format of the records in the associated data set. For the AUL it could be only: F - fixed length (U - was used for HDR2 for prelabeled tapes)
5-9	5	0x05	Block length in bytes (maximum). For the block size greater than 100000 the value is 00000.
10-14	5	0x0A	Record length in bytes (maximum). For the record size greater than 100000 the value is 00000.
15	1	0x0F	Tape density. Depends on the tape density values are following: ‘2’ for D800, ‘3’ for D1600, ‘4’ for D6250
16-33	18	0x10	Reserved
34	2	0x22	Tape recording technique. The only technique available for 9-track tape is odd parity with no translation. For a magnetic tape subsystem with Improved Data Recording Capability, the values are: ‘P’ - Record data in compacted format, ‘ ’ - Record data in standard uncompact format. For CASTOR is ‘P’ if the drive configured to use compression (i.e. xxxGC)
35-49	14	0x24	Reserved
50-51	2	0x32	Buffer offset ‘00’ for AL and AUL tapes
52-79	28	0x34	Reserved

Table C.6: The structure of the UHL1, UTL1 labels

UHL1, UTL1			
Bytes	Length	Offset	Content
0-3	4	0x00	User header label: the characters “UHL1 or UTL1”.
4-13	10	0x04	Actual file sequence number ( ‘0’ padded from left ).
14-23	10	0x0E	Actual block size ( ‘0’ padded from left ).
24-33	10	0x18	Actual record length ( ‘0’ padded from left ).
34-41	8	0x22	Site : a part of the domain name uppercase.
42-51	10	0x2A	Tape mover host name uppercase without domain name.
52-59	8	0x34	Drive manufacturer.
60-67	8	0x3C	Drive model (first 8 bytes from the field PRODUCT IDENTIFICATION in the SCSI INQUIRY replay).
68-79	12	0x44	Drive serial number.

## C.6 End of File (EOF<sub>n</sub>)

EOF1 and EOF2 are normally found together at the end of a dataset.

Note that an End of Volume (EOV<sub>n</sub>) label will appear instead of EOF<sub>n</sub> if this is the final label group on the volume, but the dataset continues on another volume. EOV1 and EOV2 are only expected together and at the end of a volume.

## C.7 User Trailer Label (UTL<sub>*n*</sub>)

The format for UTL1 is the same as UHL1 (Table C.6).

## C.8 Checksums

When a file is written to tape, an Adler32 checksum is computed on the file. The main advantages of Adler32 are that it is faster to compute than CRC32 or MD5, and it is distributive when computing the checksum for a multi-block file. This checksum is not stored on the tape; it is stored as metadata in the Catalogue.

**Note:** The tape drives also compute a CRC32 checksum on each block, which is checked in firmware. This checksum is not seen by the software.