EOS Manual

V 0.1.0

IT-DSS (C) CERN 2011

November 2, 2011

Contents

1 EOS Overview & Introduction

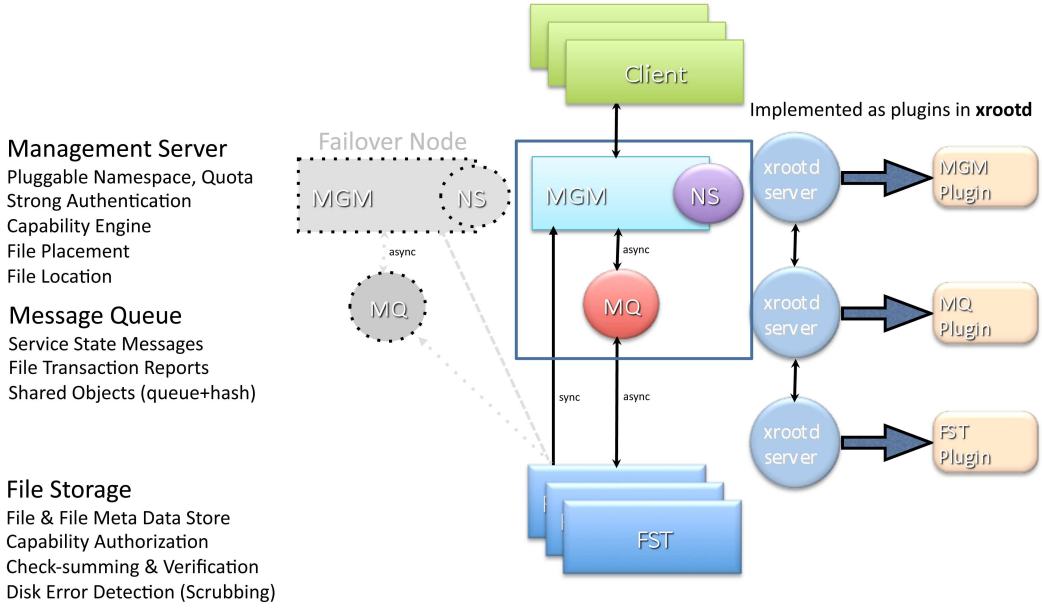
1.1 What is EOS

The EOS project was started in April 2010 in the IT DSS group. The main goal of the project is to provide fast and reliable disk only storage technology for CERN LHC use cases. EOS provides remote access via xroot protocol and can be mounted by a single user or shared in multi user mode with strong authentication via a FUSE implementation. EOS is not a POSIX filesystem but close enough to POSIX to fullfill 99% of the typical use cases. The performance is not optimized to give maximum performance for a single client as a mounted file system but rather a good compromise to allow large scaling installations with low latency and high overall performance. A key feature of EOS is dynamic storage reconfiguration with automatized draining and filesystem rebalancing.

1.1.1 Features

- High performance in-memory namespace - (e.g. ~100 kHz meta data read ops) with a memory limited namespace size (e.g. 100 Mio. files with 128 GB of RAM)
- Strong Authentication via Kerberos & X509 certificates for remote and mounted file access
- Quota System with User + Group Volume+Inode Quota based on quota nodes inside the namespace
- Round-robin file access/placement scheduler with dynamic adaption to filesystem load parameters
- File-level checksumming with software hash functions Adler32, SHA1, MD5, CRC32 and hardware hash function CRC32C (via SSE4.2 instruction set)
- Block-level checksumming (algorithm as before) for 4k, 64k, 128k, 256k, 512k & 1M blocksize
- File HA via network RAID-1 replication (1-16 replica per file on distinctive nodes)
- Namespace HA Active-Passive failover based on DNS alias
- Hardware Grouping into Space & attached scheduling Groups
- EOS shell as user & administrator interface

- Easy-to-use admin tools for life-cycle management & rapid configuration
- Meta Data & IO accounting
- User/Group/Host based Access Control Interface
- Directory-based (non-standard) additive ACLs
  - support for write-once, delete never directories as sandboxes
  - support to define ACLs via CERN EGROUPs
- Configurable Redirection on ENOENT and ENONET (file missing or inaccessible)



1.2 Server Components

1.2.1 MGM - Management & Meta Data Server

The central entry point for all clients it the MGM. In the current implementation only a single instance of an MGM can be active. For high-availability the deployment of an active-passive MGM pair is possible (see later). The MGM runs the in-memory namespace of EOS and is responsible for file, filesystem and node management and configuration. The in-memory namespace is based on the GOOGLE SparseHash implementation and as a rough estimate you need ~1kB of memory per file in the namespace. Files and directories used by the MGM are:

Location	Significance
/etc/xrd.cf.mgm	xrootd server configuration file
/etc/sysconfig/eos	instance configuration file
/var/eos/md/files.<hostname>.mdlog	changelog file cont. file meta data of the name space
/var/eos/md/directories.<hostname>.mdlog	changelog file cont. directory meta data of the name space
/var/eos/config/<hostname>	directory cont. configuration files and configuration changelog
/etc/eos.keytab	keytab file for 'sss' authentication (shared secret)
/var/log/eos/mgm/xrdlog.mgm	eos/xrootd MGM log file
/var/eos/md/so.mgm.dump	Dump of the MGM shared object hash/queues

The MGM service is operated via standard service scripts as

```
service eos start|stop|conrestart|restart|status mgm
```

The service runs under the *daemon* account on the default xrootd port 1094 and all files and directories are owned by this account.

1.2.2 MQ - Message Queue Service

The MQ broker receives and delivers asynchronous messages between MGM and FSTs and report messages. The MQ service can run on the same node like the MGM service and all connecting services (MGM+FST) authenticate via 'sss' (shared secret) keys (/etc/eos.keytab) to the message broker. It supports point-to-point messages and fan-out messages which are replicated to a group of receivers e.g. the MGM can send broadcast messages to all FSTs. All message queues are in-memory and not persistent in case of daemon restart. The asynchronous messaging of EOS however does not rely on reliable messaging. The MQ broker supports advisory messages which are used in EOS to set FST nodes online/offline and to implement the FST heartbeat. This is provided by two types of advisory messages: connect/disconnect & client query messages (whenever a client asks for messages in his queues).

The relevant files for the MQ service are summarized in the following table:

Location	Significance
/etc/xrd.cf.mq	xrootd server configuration file
/etc/sysconfig/eos	instance configuration file
/var/log/eos/mq/xrdlog.mq	eos/xrootd MQ log file
/var/log/eos/mq/proc/stats	current broker statistics file

The broker statistcis file summarizes the current broker activity.

```
[root@eos ~]# cat /var/log/eos/mq/proc/stats
mq.received 7678057
```

```
mq.delivered 81349236
mq.fanout 81348926
mq.advisory 75840764
mq.undeliverable 154670
mq.droppedmonitoring 2023
mq.total 74094518
mq.queued 10
mq.nqueues 163
mq.backloghits 0
mq.in_rate 16.277212
mq.out_rate 188.735770
mq.fan_rate 188.735770
mq.advisory_rate 167.365688
mq.undeliverable_rate 0.000000
mq.droppedmonitoring_rate 0.000000
mq.total_rate 163.571001
```

The tags of the statistics file are mainly self-explaining. In this example 163 nodes are connected to the message broker and the broker sends messages at 163 Hz. A growing number of queued messages (mq.queued) hints to some problem/dead-lock in one of the message receivers (typically the MGM). The message broker frequency is correlated to file read- or write-open activity: every open-close sequence creates one report message which are send as monitoring messages to report queues (a monitoring message is silently dumped if no receiver has subscribed to pick it up - standard messages trigger an error in the client-sender if no receiver is subscribed). Broker clients are automatically disconnected after an idle time of 100 seconds to provide a reliable heartbeat system and avoid message pileup in case of hanging clients. By default the MQ service is configured to run on port 1097 and runs preferably on the same node like the MGM service.

The MQ service is operated via standard service scripts as

```
service eos start|stop|conrestart|restart|status mq
```

1.2.3 FST - File Storage Server

The file storage server is the data storage component in EOS. It manages many local storage parititions which have to be configured in the management node. The FST cannot be contacted for IO directly by a client. Every client has to go via a MGM and one redirection to open a file on a FST node. Each FST has an in-memory meta data store which is populated in the startup phase equivalent to the MGM changelog implementation. Some meta data is tagged additionally as external attributes in the file system. The FST incorporates the standard XROOT IO server, asynchronous messaging threads, transfer multiplexer, transfer and checksum scanner threads per filesystem. File deletion & verification is implemented via asynchronous messages coming from the MGM node. Replica movements are implemented via asynchronous queues which are supervised

by a transfer multiplexer distributing them into individual transfer threads per filesystem. Scanner threads rescan all the files found on the data partitions and verify file and blockchecksum (if defined for a particular file). The result of the scan are tagged as external attributes to the file.

Location	Significance
/etc/xrd.cf.fst	xrootd server configuration file
/etc/sysconfig/eos	instance configuration file
/var/log/eos/fmd.<unix-tst>.<fsid>.mdlog	FST meta data changelog file
/var/log/eos/so.fst.dump	Dump of the FST shared object hash/queues

The FST service is operated via standard service scripts as

```
service eos start|stop|conrestart|restart|status fst
```

1.2.4 EOS Shared Objects

Backbone of MGM & FST services is a shared object implementation with distributed hashes and queues using the asynchronous messaging provided by the MQ services. A distributed hash is defined via a map of key/value pairs and a broadcast queue in the broker to where changes of hash values are broadcasted. The MGM uses fully qualified targets as broadcast queue e.g. the filesystem state if broadcasted only to the FST node managing the filesystem, while the FST broadcasts changes on a filesystem with a wildcard address (/eos/*/mgm) to all MGM nodes ins the system (remark: currently we only support one active MGM). Each shared hash key/value pair has the last change time & change id attached. Queues are represented via the hash implementation using a numerical index as key values (1,2,3 ...).

1.2.5 SYNC - Metadata replication service

The SYNC service is a dedicated xrootd server running by default on port 1096 which is used as a target server for replication of configuration and meta data changelog files from other MGM server. The SYNC service is operated via standard service scripts as

```
service eos start|stop|conrestart|restart|status sync
```

1.2.6 Generic EOS service script

The EOS service script is able to do bulk management of MGM, MQ, FST & SYNC services defined for the node. The main configuration file for EOS services is /etc/sysconfig/eos and the list of services has to be specified in this file like:

```
export XRD_ROLES="mq mgm sync"
```

The multi-role script can be called like

```
service eos start|stop|conrestart|restart|status
```

to operate on all services defined as XRD_ROLES.

1.3 Client Components

1.3.1 Remote Access Client (xrootd clients)

EOS can be accessed via the standard xrootd clients:

- xrdcp
  - command line copy tool
- xrd
  - meta data operations & deletion
- XrdClient/XrdClientAdmin
  - C++ classes for file & meta data operations
- TXNetFile/TXNetSystem
  - ROOT C++ class for file & meta data operations

1.3.2 Mounted File System Clients

Standard single user FUSE Client

The single user FUSE client (eosfsd process) creates a user private FUSE mount using credentials of the user executing the mount command. A mount/umount can be done via the EOS shell:

```
eos -b [root://<hostname>] fuse mount|umount <local mountpoint>
```

The server endpoint can also be defined via the environment variable:

```
export EOS_MGM_URL=root://<myeoshost>
```

Low-Level multi user FUSE Client

The low level FUSE client (eosd process) creates a shared multi-user mount based on a FUSE low-level implementation. It can be started via a service script:

```
service eosd start | stop | restart | condrestart | status
```

The service script allows to mount a single or multiple eos pools on individual mounts-point. The main mount is configured in the configuration file

```
/etc/sysconfig/eos:
export EOS_FUSE_MOUNT_POINT=/eos/main/ # mount point
export EOS_FUSE_MGM_ALIAS=localhost # MGM hostname of the instance
to mount
```

Additional mounts e.g. of the 'dev' & 'pro' EOS instance can be added by adding to

```
/etc/sysconfig/eos:
export EOS_FUSE_MOUNTS=''main pro dev''
```

and defining two new sysconfig files

```
/etc/sysconfig/eos.dev:
export EOS_FUSE_MOUNT_POINT=/eos/dev/ # mount point
export EOS_FUSE_MGM_ALIAS=eosdev.example.org # MGM hostname of the
dev instance to mount
/etc/sysconfig/eos.pro:
export EOS_FUSE_MOUNT_POINT=/eos/pro/ # mount point
export EOS_FUSE_MGM_ALIAS=eospro.example.org # MGM hostname of the
pro instance to mount
```

By default the FUSE mount configures 128k kernel-readahead and maximum writes of 4M. The files systems are name e.g. 'eosmain' 'eosdev' or 'eospro' in the previous example. Here is a list of additional configuration parameters for /etc/sysconfig/eos[.XX] files:

```
export EOS_NOACCESS=0|1 # default 1 - currently access is not called
at all
export EOS_KERNELCACHE=0|1 # default 1 - let the kernel buffercache
cache files on that fuse mount
export EOS_DIRECTIO=0|1 # default 0 - bypasses 4k page chunking and
the buffer cache for large pipes and big read/writes
export EOS_DEBUG # default undefined - enables EOS FUSE debug & XRD
Client debug - need run eosd in interactive mode with '-d' to get
the debug on the console
```

1.4 Extern Storage Connectors

1.4.1 gridFTP

The EOS gridFTP door is provided by the 'eos-gridftp' service. The gridFTP server forks a private process for each mapped client and this process bridges the gridFTP protocol to xrootd protocol using the XrdPosix interface of the xrootd client. The interface allows normal open/read/write/stat/rm/mkdir/rmdir/close semantics and additionally supports the checksum-get command. Throughput can be scaled easily by deploying several gridFTP gateway nodes. The following table shows the locations of config and log files.

Location	Significance
/var/log/eos/gridftp/ globus-gridftp-server.log	default transfer log file
/var/log/eos/gridftp/ globus-gridftp-server-debug.log	debug log file with var. verbosity
/etc/sysconfig/eos-gridftp-server	configuration file

The service is handled with the corresponding service script:

```
service eos-gridftp start|stop|restart|condrestart|status
```

The service configuration file is /etc/sysconfig/eos-gridftp-server and should define atleast the following variables:

```
GLOBUS_TCP_PORT_RANGE=20000,21000 # GLOBUS TCP port range
GLOBUS_UDP_PORT_RANGE=20000,21000 # GLOBUS UDP port range
export XROOTD_VMP=eosdev.cern.ch:1094:/ # virtual mountpoint e.g.
MGM URL + root directory
export XRDPOSIX_RCSZ=0 # disable the XROOT client readcache => we
see errors immedeatly after each read/write request
```

Other variables to specify:

export GLOBUS_LOCATION=/opt/globus #alternative GLOBUS installation directory

export GLOBUS_GRIDFTP_PORT=2811 # alternative GLOBUS gridftp port

export GLOBUS_GRIDFTP_OPTIONS="" # options passed to the gridftp command

export GLOBUS_GRIDFTP_DEBUGLEVEL="warn,error,info" # debug level used in logging

The following RPM packages are required for a gridFTP gateway:

```
eos-dsi
eos-client
xrootd-client
xrootd-libs
vdt_globus_essentials
vdt_globus_data_server
```

Each gridFTP node needs a valid host certificate, grid-map file and CA certificates according to standard GLOBUS installations.

1.4.2 SRM

The SRM service is provided by BeStMan2 operating on a FUSE mount (eosd). Because of the stateful protocol it is not possible to have several load-balanced SRM nodes. Moreover be aware that the observed performance is a mixture of 1 Hz Get, 1 Hz Put/PutDone, 10Hz Ls requests aso.

The service is handled with the corresponding service script:

```
service eos-gridftp start|stop|restart|condrestart|version|status
```

Location	Significance
/var/log/bestman2/bestman2.out	bestman2 STDOUT
/var/log/bestman2/event.srm.log.<#>	SRM command log
/opt/bestman2/conf/bestman2.rc	configuration file

This is an example file for bestman2.rc:

```
############################################################
# BeStMan server GATEWAY mode configuration
# installed on Thu Mar 17 12:17:50 CET 2011
# administrative guide on http://sdm.lbl.gov/bestman
############################################################
############################################################ EventLogLocation=/var/log/bestman2
# Modify eventLogLevel to adjust logging level (DEBUG | INFO) eventLogLevel=INFO
############################################################
# Provide blocking for user access to the list of the local directories #
# default:  /;/etc;/var ## All definitions will include the default blocked list #
# e.g.  localPathListToBlock=/;/etc;/var
############################################################
# Provide permission for user access to the list of the local directories #
# If a path is listed on both blocked and allowed list, ## blocked is taken priority.  #
# e.g.  localPathListAllowed=/home;/project
############################################################
securePort=8443
CertFileName=/etc/grid-security/bestman2/hostcert.pem
KeyFileName=/etc/grid-security/bestman2/hostkey.pem
############################################################
# supportedProtocolList can be defined #
# When different from the same hostname that BeStMan runs on #
# Or, multiple transfer servers are supported #
# supportedProtocolList=gsiftp://host1.domain.tld;gsiftp://host2.domain.tld
############################################################
### For Xrootd, xrootdTokenCompName=oss.cgroup may be needed
############################################################
```

```
pathForToken=true
checkSizeWithFS=true
checkSizeWithGsiftp=false
accessFileSysViaSudo=true
############################################################
# MaxMappedIDCached limits how many mapped ids can be cached at a given time #
# LifetimeSecondsMappedIDCached limits how long mapped ids are cached #
# LifetimeSecondsMappedIDCached=0 :  permanently cache ## LifetimeSecondsMappedIDCached<0
:  never cache #
# LifetimeSecondsMappedIDCached>0 :  cache for the duration
############################################################
MaxMappedIDCached=1000
LifetimeSecondsMappedIDCached=1800
GridMapFileName=/etc/grid-security/grid-mapfile
GridMapFileName=/etc/grid-security/grid-mapfile
############################################################
# Do Not edit below unless you really really understand the entries
############################################################
# GATEWAY mode entries
############################################################
disableSpaceMgt=true
useBerkeleyDB=false
noCacheLog=true
############################################################
FactoryID=srm/v2/server
noEventLog=false
NoDirBrowsing=true
defaultChecksumType=adler32
hexChecksumCommand=/usr/sbin/eos-srm-checksum
localPathListAllowed=/eos
showChecksumWhenListingFile=true
staticTokenList=ATLASDATADISK[desc:ATLASDATADISK][owner:atlas][retention:REPLICA][latency:ONLINE]\
[path:/eos/atlas/atlasdatadisk][usedBytesCommand:/usr/sbin/eos-srm-used-bytes][totalBytesCommand:/usr/sbin/eos-sr
ATLASSCRATCHDISK[desc:ATLASSCRATCHDISK][owner:atlas][retention:REPLICA][latency:ONLINE]\
[path:/eos/atlas/atlasscratchdisk][usedBytesCommand:/usr/sbin/eos-srm-used-bytes][totalBytesCommand:/usr/sbin/eos
ATLASGROUPDISK[desc:ATLASGROUPDISK][owner:atlas][retention:REPLICA][latency:ONLINE]\
[path:/eos/atlas/atlasgroupdisk][usedBytesCommand:/usr/sbin/eos-srm-used-bytes][totalBytesCommand:/usr/sbin/eos-s
ATLASEOSDATADISK[desc:ATLASEOSDATADISK][owner:atlas][retention:REPLICA][latency:ONLINE]\
[path:/eos/atlas/atlaseosdatadisk][usedBytesCommand:/usr/sbin/eos-srm-used-bytes][totalBytesCommand:/usr/sbin/eos
ATLASEOSSCRATCHDISK[desc:ATLASEOSSCRATCHDISK][owner:atlas][retention:REPLICA][latency:ONLINE]\
[path:/eos/atlas/atlaseosscratchdisk][usedBytesCommand:/usr/sbin/eos-srm-used-bytes][totalBytesCommand:/usr/sbin/
supportedProtocolList=gsiftp://eosatlassftp.cern.ch
```

The following RPM packages are required for the SRM service:

```
eos-fuse
eos-srm
eos-bestman2
eos-client-0.1.0-rc40a
java-1.6.0-openjdk
java-1.6.0-openjdk-devel
xrootd-libs
xrootd-client
```

Each SRM node needs a valid host certificate, grid-map file and CA certificates according to standard GLOBUS installations.

1.5 High Availability & Backup

1.5.1 MGM Failover in Master-Slave Configuration

1.5.2 Meta Data Backup & Replication

2 Installation & Configuration

2.1 Installation

2.1.1 Requirements

Management Nodes

File Storage Nodes

SRM Node

gridFTP Nodes

2.1.2 Source Installation

2.1.3 Binary Installation

2.2 Steps to Install EOS

2.3 Configuring EOS

2.3.1 Configuration Files

2.3.2 Configuring a Single Node EOS Storage

2.3.3 Scaling EOS Storage - Multi Node Storage

2.3.4 Configuring EOS Failover

3 Using EOS

3.1 EOS Shell - User Commands

3.1.1 Permission System

3.1.2 Namespace Interface

3.1.3 File Layouts

3.1.4 File-Level Checksumming

3.1.5 Block-Level Checksumming

3.1.6 Quota

4 EOS Operations

4.1 EOS Shell - Admin Commands

5 EOS Maintenance

6 EOS Monitoring

7 Upgrading EOS