The XtreemFS Installation and User Guide

Version 1.0

XtreemFS is available from the XtreemFS website (www.XtreemFS.org).

Contents

Chapter 1

Quick Start

This is the very short version to help you set up a local installation of XtreemFS.

1. Download XtreemFS RPMs/DEBs and install

   (a) Download the RPMs or DEBs for your system from the XtreemFS website

   (b) open a root console (`su` or `sudo`)

   (c) install with `rpm -Uhv xtreemfs-client-1.0.x.rpm xtreemfs-server-1.0.x.rpm`

2. Start the Directory Service:
   `/etc/init.d/xtreemfs-dir start`

3. Start the Metadata Server:
   `/etc/init.d/xtreemfs-mrc start`

4. Start the OSD:
   `/etc/init.d/xtreemfs-osd start`

5. If not already loaded, load the FUSE kernel module:
   `modprobe fuse`

6. Depending on your distribution, you may have to add users to a special group to allow them to mount FUSE file systems. In openSUSE users must be in the group `trusted`, in Ubuntu in the group `fuse`. You may need to log out and log in again for the new group membership to become effective.

7. You can now close the root console and work as a regular user.

8. Wait a few seconds for the services to register at the directory service. You can check the registry by opening the DIR status page in your favorite web browser http://localhost:30638.

9. Create a new volume with the default settings:
   `xtfs_mkvol localhost/myVolume`

10. Create a mount point:
    `mkdir ~/xtreemfs`

11. Mount XtreemFS on your computer:

    ```
    xtfs_mount localhost/myVolume ~/xtreemfs
    ```

12. Have fun ;-)

13. To un-mount XtreemFS:
    ```
    xtfs_umount ~/xtreemfs
    ```

You can also mount this volume on remote computers. First make sure that the ports 32636, 32638 and 32640 are open for incoming TCP connections. You must also specify a hostname that can be resolved by the remote machine! This hostname has to be used instead of `localhost` when mounting.

Chapter 2

What is XtreemFS

2.1 About XtreemFS

With XtreemFS you are about to install a modern *distributed and replicated file system*. As a distributed file system, XtreemFS stores your file data on several servers and you can simply scale your file system by adding more hosts. XtreemFS is a full-featured file system that supports the full POSIX file interface, including *extended attributes* (xattrs). In case of concurrent access by several distributed programs, XtreemFS provides you currently with NFS close-to-open consistency.

With version 1.0 XtreemFS also supports replication of files. The so called read-only replication allows you to have multiple copies of immutable files. XtreemFS also supports partial replicas which helps you to save disk capacity and network bandwidth; only data that is requested by clients is stored in partial replicas.

XtreemFS has been designed for deployment in *wide-area environments* connected by the Internet. This means that it allows you to mount an XtreemFS volume from any location, given the right permissions; but it also implies that file system installations can span multiple locations or data centers.

In a normal UNIX environment, XtreemFS has full permission and *POSIX ACL* support. XtreemFS can also be integrated into *X.509-based security architectures*. Access policies (as well several other policies) are pluggable and can be easily extended. If you deploy XtreemFS as part of an *XtreemOS* installation, you will benefit from its transparent integration with the XtreemOS *Virtual Organization (VO)* infrastructure in the form of dynamic user mappings and automatic mounting of home volumes.

If you need *high-performance* access to your files, XtreemFS can help you with support for *file striping*: XtreemFS can store a file across several storage servers and *access* the parts *in parallel*. The size of an individual stripe and the number of storage servers used can be configured on a per-file or per-directory basis.
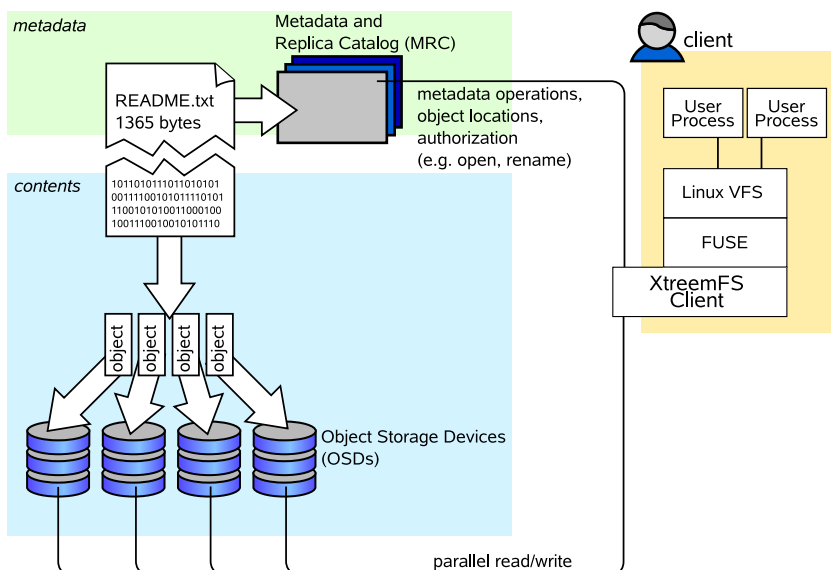
Figure 2.1: The XtreemFS architecture and components.

2.2   XtreemFS Architecture

XtreemFS implements an *object-based file system architecture* (Fig. 2.1). The name of this architecture comes from the fact that an object-based file system splits file content into a series of fixed-size *objects* and stores them on its storage servers. In contrast to block-based file systems, the size of such an object can vary from file to file.

The *metadata* of a file (such as the file name or file size) is stored separate from the file content on a Metadata server. This metadata server organizes file system metadata as a set of *volumes*, each of which implements a separate file system namespace in form of a directory tree.

2.2.1   The Components of XtreemFS

An XtreemFS installation contains three types of servers that can run on one or several machines (Fig. 2.1):

- DIR - Directory Service
  The directory service is the central registry for all services in XtreemFS. The MRC uses it to discover storage servers.

- MRC - Metadata and Replica Catalog
  The MRC stores the directory tree and file metadata such as file name, size or modification time. Moreover, the MRC authenticates users and authorizes access to files.

- OSD - Object Storage Device
  An OSD stores arbitrary objects of files; clients read and write file data on OSDs.

These servers are connected by the *client* to a file system. A client *mounts* one of the volumes of the MRC in a local directory. It translates file system calls into RPCs sent to the respective servers.

The client is implemented as a *FUSE user-level driver* that runs as a normal process. FUSE itself is a kernel-userland hybrid that connects the user-land driver to Linux' *Virtual File System (VFS)* layer where file system drivers usually live.

2.2.2 Security

As usual, XtreemFS security differentiates between authentication and authorization. *Authentication* is the process of verifying a user's or client's identity, e.g. validating and reading an X.509 certificate. In contrast, *authorization* is the process of checking if a user has the permission to execute a certain operation, e.g. write access to a file.

By default, XtreemFS uses unauthenticated and unencrypted TCP connections. However, *SSL* can be enabled in all XtreemFS services and the client. Using SSL requires that all users and services provide valid X.509 certificates. Any data sent over a SSL connection is encrypted. Using SSL, however, will increase the resource consumption of all components, especially for connection setup (SSL handshake).

2.3 Policies

Many facets of the behavior of XtreemFS can be configured by means of policies. A policy defines how a certain task is performed, e.g. how the MRC selects a set of OSDs for a new file, or how it distinguishes between an authorized and an unauthorized user when files are accessed. Various policies have been defined that cover different aspects.

2.3.1 OSD Selection Policies

When a *new file* is created, the MRC must decide which OSDs to use for storing the file content. Based on the required number of OSDs defined in the file's striping policy, an OSD Selection Policy is responsible for selecting the most suitable OSDs. OSD selection policies are assigned at volume granularity. Currently, there are the following policies:

- Random OSD Selection (policy id 1)
  Randomly selects OSDs from the list of all available OSDs that are alive and have more than 2GB of free space left.

- Proximity-based OSD Selection (policy id 2)
  Selects a group of OSDs that are close to each other. The distance is determined by the IP address, i.e. OSDs on the same subnet are preferred. This policy is particularly useful for striping, since it is desirable to have all OSDs at the same site.

- DNS based OSD Selection (policy id 3)
  The FQDN of the client and all OSDs is compared and the maximum match
  (from the end of the FQDN) is used to sort the OSDs. The policy selects the
  OSDs with the highest number of characters that match. This policy can be
  used to automatically select OSDs which are close to the client, if the length
  of the match between two DNS entries also indicate a low latency between
  two machines. The minimum match can be modified by setting the xattr
  'xtreemfs.osdsel_policy_args' to an integer value larger 0. OSDs with
  a match smaller than the minimum match are never used.

2.3.2   Replica Selection Policies

When a client opens a file with more than one replica, the MRC uses a replica selec-
tion policy to sort the list of replicas for the client. Clients will always use the first
replica in the list. If a replica is not available, the client will automatically select the
next replica from the list. Replica selection policies can be used to sort the replica
lists, e.g. to ensure that clients use replicas which are close to them. By default, the
MRC returns an unmodified list of replicas. Section 4.5 describes how to set and
display the policy of a volume.

- Default Replica Selection (policy id 1)
  The MRC returns an unmodified list of replicas to the client. This is the default
  policy.

- DNS based Replica Selection (policy id 3)
  This policy is identical to the DNS based OSD Selection (see Sec. 2.3.1). The
  list of replicas is sorted from the longest FQDN name match to the shortest.

- Datacenter Map Replica Selection (policy id 4)
  This policy uses a statically configured datacenter map that describes the dis-
  tance between datacenters. It works only with IPv4 addresses at the moment.
  Each datacenter has a list of matching IP addresses and networks which is used
  to assign clients and replicas (OSDs) to datacenters. The replica list is sorted
  according to the distance between replica and client starting from the closest
  (smallest distance). Machines in the same datacenter have a distance of 0.

  This policy requires a datacenter map configuration file in /etc/xos/xtreemfs/datacentermap
  on the MRC machine which is loaded at MRC startup. This config file must
  contain the following parameters:

  - datacenters=A,B,C
    A comma separated list of datacenters. Datacenter names may only con-
    tain a-z, A-Z, 0-9 and _.

  - distance.A-B=100
    For each pair of datacenters, the distance must be specified. As distances
    are symmetric, it is sufficient to specify A to B.

  - addresses.A=192.168.1.1,192.168.2.0/24
    For each datacenter a list of matching IP addresses or networks must be
    specified.

```
– max_cache_size=1000
```
Sets the size of the address cache that is used to lookup IP-to-datacenter matches.

A sample datacenter map could look like this:

```
datacenters=BERLIN,LONDON,NEW_YORK
distance.BERLIN-LONDON=10
distance.BERLIN-NEW_YORK=140
distance.LONDON-NEW_YORK=110
addresses.BERLIN=192.168.1.0/24
addresses.LONDON=192.168.2.0/24
addresses.NEW_YORK=192.168.3.0/24,192.168.100.0/25
max_cache_size=100
```

2.3.3 Striping Policies

XtreemFS allows the content of a file to be distributed among several storage devices (OSDs). This has the benefit, that the file can be read or written in parallel on multiple servers which increases the bandwidth. The more OSDs are used, the higher the bandwidth available for reading or writing. The number of OSDs is called the striping width.

A striping policy is a rule that defines how the objects are distributed on the available OSDs. Currently, XtreemFS implements only the RAID0 policy which simply stores the objects in a round robin fashion on the OSDs. The RAID0 policy has two parameters. The striping width defines to how many OSDs the file is distributed. The stripe size defines the size of each object.

When using a striping width of 1, the files are not striped but each file is stored on a single OSD. In that case, you can use any OSD Selection Policy which suits your needs. For striped files (i.e. a striping width larger than 1) we recommend to use the Proximity-based OSD selection policy, because the OSDs onto the files are striped should reside on the same network for better performance and data availability.

Striping over several OSDs enhances the read and write bandwidth of a file, the bandwidth increases the larger the striping width. Please note, that striping also increases the probability of data loss. A striped file will become corrupted even if a single OSDs it is stored on has a disk crash.

2.3.4 Authorization - Access Policies

User authorization is managed by means of Access Policies. An access policy defines the access rights for any user on any file or directory contained in a volume. When creating a new volume, the access policy has to be chosen, which cannot be changed in the future. Various access policies can be used:

- Authorize All Policy (policy Id 1)
  No authorization - everyone can do everything. This policy is useful if performance of metadata operations matters more than security, since no recursive evaluation of access policies is done.

- POSIX ACLs & Permissions (policy Id 2)
  This access policy implements the traditional POSIX permissions commonly
  used on Linux, as well as POSIX ACLs, an extension that provides for ac-
  cess control at the granularity of single users and groups. POSIX permissions
  should be used as the default, as it guarantees maximum compatibility with
  other file systems.

- Volume ACLs (policy Id 3)
  Volume ACLs provide an access control model similar to POSIX ACLs & Per-
  missions, but only allow one ACL for the whole volume. This means that
  there is no recursive evaluation of access rights which yields a higher perfor-
  mance at the price of a very coarse-grained access control.

2.3.5   Pluggable Policies

Administrators may extend the set of existing policies by defining *plug-in policies*.
Such policies are Java classes that implement a predefined policy interface. Currently,
the following policy interfaces exist:

- `org.xtreemfs.common.auth.AuthenticationProvider`
  can be used to implement an individual mechanism to authenticate users and
  groups

- `org.xtreemfs.mrc.ac.FileAccessPolicy`
  can be used to implement an individual access control model on files, directo-
  ries and volumes

- `org.xtreemfs.mrc.osdselection.OSDSelectionPolicy`
  can be used to implement an individual policy for allocating OSDs to newly
  created files

Note that there may only be one authentication provider per MRC, while file access
policies and OSD selection policies may differ for each volume. The former one
is identified by means of its class name (property `authentication_provider`, see
Sec. 3.2.4), while volume-related policies are identified by ID numbers. It is therefore
necessary to add a member field

```
 public static final long POLICY_ID = 4711;
```

to all such policy implementations, where 4711 represents the individual ID num-
ber. Administrators have to ensure that such ID numbers neither clash with ID
numbers of built-in policies (1-9), nor with ID numbers of other plug-in policies.
When creating a new volume, IDs of plug-in policies may be used just like built-in
policy IDs.

Plug-in policies have to be deployed in the directory specified by the MRC config-
uration property `policy_dir`. The property is optional; it may be omitted if no
plug-in policies are supposed to be used. An implementation of a plug-in policy can
be deployed as a Java source or class file located in a directory that corresponds to
the package of the class. Library dependencies may be added in the form of source,
class or JAR files. JAR files have to be deployed in the top-level directory. All source
files in all subdirectories are compiled at MRC start-up time and loaded on demand.

Chapter 3

XtreemFS Services

3.1 Installation

When installing XtreemFS, you can choose from two different installation sources: you can download one of the *pre-packaged releases* that we create for most Linux distributions or you can install directly from the *source tarball*. In the pre-packaged release, the server and the client parts are split into separate packages.

3.1.1 Prerequisites

For the pre-packaged release, you will need Sun Java JRE 1.6.0 or newer to be installed on the system.

When building XtreemFS directly from the source, you need a Sun Java JDK 1.6.0 or newer, Ant 1.6.5 or newer and gmake.

3.1.2 Installing from Pre-Packaged Releases

On RPM-based distributions (RedHat, Fedora, SuSE, Mandriva, XtreemOS) you can install the package with

```
$> rpm -i xtreemfs-server-1.0.x.rpm
```

For Debian-based distributions, please use the `.deb` package provided and install it with

```
$> dpkg -i xtreemfs-server-1.0.x.deb
```

Both packages will also install `init.d` scripts for an automatic start-up of the services. Use `insserv xtreemfs-dir`, `insserv xtreemfs-mrc` and `insserv xtreemfs-osd`, respectively, to automatically start the services during boot.

3.1.3   Installing from Sources

Extract the tarball with the sources. Change to the top level directory and execute

```
$> make server
```

3.2   Configuration

Generally, the configuration files of XtreemFS are located in `/etc/xos/xtreemfs/` if you installed from packages.

3.2.1   A Word about UUIDs

XtreemFS uses UUIDs (Universally Unique Identifiers) to be able to identify services and their associated state independently from the machine they are installed on. This implies that you cannot change the `uuid` of a MRC or OSD after it has been used for the first time!

The Directory Service keeps a mapping from UUID to a port number and IP address or hostname. Currently, each UUID can only be assigned to a single endpoint; the netmask must be "*" which means that this mapping is valid in all networks. Upon first start-up, OSDs and MRCs will create the mapping if it does not exist. They will use the first available network device with a public address.

Changing the IP address, hostname or port is possible at any time. Due to the caching of UUIDs in all components it can take some time until the new UUID mapping is used by all OSDs, MRCs and clients. The TTL defines how long an XtreemFS component is allowed to keep entries cached. The default value is 3600 seconds (1 hour). It should be set to shorter durations if services change their IP address frequently.

To create a globally unique UUID you can use tools like `uuidgen`. During installation the post-install script will automatically create a UUID for each OSD and MRC if it does not have a UUID assigned.

3.2.2   Automatic DIR Service Discovery

The OSD and MRC discover the DIR service automatically by default. On startup they will broadcast requests to the local LAN and wait up to 10s for a response from a DIR. This works only in a local LAN environment as broadcast messages are not routed to other networks. Local firewalls on the computers on which the services are running can also prevent the automatic discovery from working. The services will select the first DIR which responded which can lead to non-deterministic behaviour if multiple DIR services are present.

Security: The automatic discovery is a potential security risk when used in untrusted environments as any user can start-up DIR services.

A statically configured DIR address and port can be used to disable DIR discovery in the OSD and MRC (see Sec. 3.2.4, `dir_service`). By default. the DIR responds to UDP broadcasts. To disable this feature, set `discover = false` in the DIR service config file.

3.2.3 Authentication

XtreemFS has an interface which allows MRC administrators to choose the way of authenticating users. Basically, an MRC has two sources of information on users. The first one is the user id and group ids sent by the client along with each request. In addition, the MRC can use information included in the certificates if SSL is enabled. The Authentication Providers are modules that implement different methods for retrieving the user and group IDs to use.

UNIX uid/gid - NullAuthProvider

The NullAuthProvider is the default Authentication Provider. It simply uses the user ID and group IDs sent by the XtreemFS client. This means that the client is trusted to send the correct user/group IDs.

The XtreemFS Client will send the user ID and group IDs of the process which executed the file system operation, not of the user who mounted the volume!

The superuser is identified by the user ID `root` and is allowed to do everything on the MRC. This behavior is similar to NFS with `no_root_squash`.

Plain SSL Certificates - SimpleX509AuthProvider

XtreemFS supports two X.509 certificate "types" which can be used by the client. When mounted with a service/host certificate the XtreemFS client is regarded as a trusted system component. The MRC will accept any user ID and groups sent by the client and use them for authorization as with the NullAuthProvider. This setup is useful for volumes which are used by multiple users.

The second certificate type are regular user certificates. The MRC will only accept the user name and group from the certificate and ignore the user ID and groups sent by the client. Such a setup is useful if users are allowed to mount XtreemFS from untrusted machines.

Both certificates are regular X.509 certificates. Service and host certificates are identified by a Common Name (`CN`) starting with `host/` or `xtreemfs-service/`, which can easily be used in existing security infrastructures. All other certificates are assumed to be user certificates.

If a user certificate is used, XtreemFS will take the Distinguished Name (`DN`) as the user ID and the Organizational Unit (`OU`) as the group ID.

Superusers must have `xtreemfs-admin` as part of their Organizational Unit (`OU`).

XtreemOS Certificates - XOSAuthProvider

In contrast to plain X.509 certificates, XtreemOS embeds additional user information as extensions in XtreemOS-User-Certificates. This authentication provider uses this information (global UID and global GIDs), but the behavior is similar to the SimpleX509AuthProvider.

The superuser is identified by being member of the `VOAdmin` group.

3.2.4   List of Configuration Options

All configuration parameters that may be used to define the behavior of the different services are listed in this section. Unless marked as optional, a parameter has to occur (exactly once) in a configuration file.

`admin_password` *optional*

Services	DIR, MRC, OSD
Values	String
Default	empty
Description	Defines the admin password that must be sent to authorize requests like volume creation, deletion or shutdown.

`authentication_provider`

Services	MRC
Values	Java class name
Default	`org.xtreemfs.common.auth.NullAuthProvider`
Description	Defines the Authentication Provider to use to retrieve the user identity (user ID and group IDs). See Sec. 3.2.3 for details.

`capability_secret`

Services	MRC, OSD
Values	String
Default	-
Description	Defines a shared secret between the MRC and all OSDs. The secret is used by the MRC to sign capabilities, i.e. security tokens for data access at OSDs. In turn, an OSD uses the secret to verify that the capability has been issued by the MRC.

`checksums.enabled`

Services	OSD
Values	true, false
Default	false
Description	If set to true, the OSD will calculate and store checksums for newly created objects. Each time a checksummed object is read, the checksum will be verified.

`checksums.algorithm`

Services	OSD
Values	Adler32, CRC32
Default	Adler32
Description	Must be specified if `checksums.enabled` is enabled. This property defines the algorithm used to create OSD checksums.

`database.dir`

Services	DIR, MRC
Values	absolute file system path to a directory
Default	DIR: `/var/lib/xtreemfs/dir/database`,
	MRC: `/var/lib/xtreemfs/mrc/database`
Description	The directory in which the Directory Service or MRC will store their databases. This directory should never be on the same partition as any OSD data, if both services reside on the same machine. Otherwise, deadlocks may occur if the partition runs out of free disk space!

`database.log`

Services	MRC
Values	absolute file system path
Default	MRC: `/var/lib/xtreemfs/mrc/dblog`
Description	The directory the MRC uses to store database logs. This directory should never be on the same partition as any OSD data, if both services reside on the same machine. Otherwise, deadlocks may occur if the partition runs out of free disk space!

`debug.level` *optional*

Services	DIR, MRC, OSD
Values	0, 1, 2, 3, 4, 5, 6, 7
Default	6
Description	The debug level determines the amount and detail of information written to logfiles. Any debug level includes log messages from lower debug levels. The following log levels exist:

> 0 - fatal errors
>
> 1 - alert messages
>
> 2 - critical errors
>
> 3 - normal errors
>
> 4 - warnings
>
> 5 - notices
>
> 6 - info messages
>
> 7 - debug messages

debug.categories *optional*

Services	DIR, MRC, OSD
Values	all, lifecycle, net, auth, stage, proc, db, misc
Default	all
Description	Debug categories determine the domains for which log messages will be printed. By default, there are no domain restrictions, i.e. log messages form all domains will be included in the log. The following categories can be selected:

> all - no restrictions on the category
>
> lifecycle - service lifecycle-related messages, including startup and shutdown events
>
> net - messages pertaining to network traffic and communication between services
>
> auth - authentication and authorization-related messages
>
> stage - messages pertaining to the flow of requests through the different stages of a service
>
> proc - messages about the processing of requests
>
> db - messages that are logged in connection with database accesses
>
> misc - any other log messages that do not fit in one of the previous categories
>
> Note that it is possible to specify multiple categories by means of a comma or space-separated list.

dir_service.host

Services	MRC, OSD
Values	hostname or IP address
Default	.autodiscover
Description	Specifies the hostname or IP address of the directory service (DIR) at which the MRC or OSD should register. The MRC also uses this directory service to find OSDs. If set to .autodiscover the service will use the automatic DIR discovery mechanism (see Sec. 3.2.2).

dir_service.port

Services	MRC, OSD
Values	1 .. 65535
Default	32638
Description	Specifies the port on which the remote directory service is listening. Must be identical to the listen_port in your directory service configuration.

discover *optional*

Services	DIR
Values	true, false
Default	true
Description	If set to true the DIR will received UDP broadcasts and advertise itself in response to XtreemFS components using the DIR automatic discoery mechanism. If set to false, the DIR will ignore all UDP traffic. For details see Sec. 3.2.2.

geographic_coordinates

Services	DIR, MRC, OSD
Values	String
Default	empty
Description	Specifies the geographic coordinates which are registered with the directory service. Used e.g. by the web console.

http_port

Services	DIR, MRC, OSD
Values	1 .. 65535
Default	30636 (MRC), 30638 (DIR), 30640 (OSD)
Description	Specifies the geographic coordinates which are registered with the directory service. Used e.g. by the web console.

listen.address *optional*

Services	OSD
Values	IP address
Default	-
Description	If specified, defines the interface to listen on. If not specified, the service will listen on all interfaces (any).

listen.port

Services	DIR, MRC, OSD
Values	1 .. 65535
Default	DIR: 32638, MRC: 32636, OSD: 32640
Description	The port to listen on for incoming connections (TCP). The OSD uses TCP and UDP on the specified port. Make sure to configure your firewall to allow incoming TCP and UDP traffic on the specified port.

`local_clock_renewal`

Services	MRC, OSD
Values	milliseconds
Default	50
Description	Reading the system clock is a slow operation on some systems (e.g. Linux) as it is a system call. To increase performance, XtreemFS services use a local variable which is only updated every `local_clock_renewal` milliseconds.

`no_atime`

Services	MRC
Values	true, false
Default	true
Description	The POSIX standard defines that the atime (timestamp of last file access) is updated each time a file is opened, even for read. This means that there is a write to the database and hard disk on the MRC each time a file is read. To reduce the load, many file systems (e.g. ext3) including XtreemFS can be configured to skip those updates for performance. It is strongly suggested to disable atime updates by setting this parameter to true.

`no_fsync` *optional*

Services	MRC
Values	true, false
Default	false
Description	By default, the MRC will write all file-modifying operations (such as create file, delete etc.) to disk followed by a `fsync` to ensure data is written to the hard disk. While this ensures maximum data safety in case of crash of the MRC server, it also reduces the performance of the MRC. Set this to true, if you want much higher performance at the risk of losing some recent file operations in case of a server crash.

`object_dir`

Services	OSD
Values	absolute file system path to a directory
Default	`/var/lib/xtreemfs/osd/`
Description	The directory in which the OSD stores the objects. This directory should never be on the same partition as any DIR or MRC database, if both services reside on the same machine. Otherwise, deadlocks may occur if the partition runs out of free disk space!

osd_check_interval

Services	MRC
Values	seconds
Default	300
Description	The MRC regularly asks the directory service for suitable OSDs to store files on (see OSD Selection Policy, Sec. 2.3.1). This parameter defines the interval between two updates of the list of suitable OSDs.

remote_time_sync

Services	MRC, OSD
Values	milliseconds
Default	30,000
Description	MRCs and OSDs all synchronize their clocks with the directory service to ensure a loose clock synchronization of all services. This is required for leases to work correctly. This parameter defines the interval in milliseconds between time updates from the directory service.

report_free_space

Services	OSD
Values	true, false
Default	true
Description	If set to true, the OSD will report its free space to the directory service. Otherwise, it will report zero, which will cause the OSD not to be used by the OSD Selection Policies (see Sec. 2.3.1).

ssl.enabled

Services	DIR, MRC, OSD
Values	true, false
Default	false
Description	If set to true, the service will use SSL to authenticate and encrypt connections. The service will not accept non-SSL connections if ssl.enabled is set to true.

ssl.service_creds

Services	DIR, MRC, OSD
Values	path to file
Default	DIR: /etc/xos/xtreemfs/truststore/certs/ds.p12, MRC: /etc/xos/xtreemfs/truststore/certs/mrc.p12, OSD: /etc/xos/xtreemfs/truststore/certs/osd.p12
Description	Must be specified if ssl.enabled is enabled. Specifies the file containing the service credentials (X.509 certificate and private key). PKCS#12 and JKS format can be used, set ssl.service_creds.container accordingly. This file is used during the SSL handshake to authenticate the service.

ssl.service_creds.container

Services	DIR, MRC, OSD
Values	pkcs12 or JKS
Default	pkcs12
Description	Must be specified if `ssl.enabled` is enabled. Specifies the file format of the `ssl.service_creds` file.

ssl.service_creds.pw

Services	DIR, MRC, OSD
Values	String
Default	-
Description	Must be specified if `ssl.enabled` is enabled. Specifies the password which protects the credentials file `ssl.service_creds`.

ssl.trusted_certs

Services	DIR, MRC, OSD
Values	path to file
Default	/etc/xos/xtreemfs/truststore/certs/xosrootca.jks
Description	Must be specified if `ssl.enabled` is enabled. Specifies the file containing the trusted root certificates (e.g. CA certificates) used to authenticate clients.

ssl.trusted_certs.container

Services	DIR, MRC, OSD
Values	pkcs12 or JKS
Default	JKS
Description	Must be specified if `ssl.enabled` is enabled. Specifies the file format of the `ssl.trusted_certs` file.

ssl.trusted_certs.pw

Services	DIR, MRC, OSD
Values	String
Default	-
Description	Must be specified if `ssl.enabled` is enabled. Specifies the password which protects the trusted certificates file `ssl.trusted_certs`.

uuid

Services	MRC, OSD
Values	String, but limited to alphanumeric characters, - and .
Default	-
Description	Must be set to a unique identifier, preferably a UUID according to RFC 4122. UUIDs can be generated with `uuidgen`. Example: `eacb6bab-f444-4ebf-a06a-3f72d7465e40`.

3.2.5 Configuring SSL Support

In order to enable certificate-based authentication in an XtreemFS installation, services need to be equipped with X.509 certificates. Certificates are used to establish a mutual trust relationship among XtreemFS services and between the XtreemFS client and XtreemFS services.

It is not possible to mix SSL-enabled and non-SSL services in an XtreemFS installation!

Each XtreemFS service needs a certificate and a private key in order to be run. Once they have been created and signed, the credentials may need to be converted into the correct file format. XtreemFS services also need a *trust store* that contains all trusted Certification Authority certificates.

By default, certificates and credentials for XtreemFS services are stored in

```
/etc/xos/xtreemfs/truststore/certs
```

Converting PEM files to PKCS#12

The simplest way to provide the credentials to the services is by converting your signed certificate and private key into a PKCS#12 file using `openssl`:

```
$> openssl pkcs12 -export -in ds.pem -inkey ds.key \
   -out ds.p12 -name "DS"
$> openssl pkcs12 -export -in mrc.pem -inkey mrc.key \
   -out mrc.p12 -name "MRC"
$> openssl pkcs12 -export -in osd.pem -inkey osd.key \
   -out osd.p12 -name "OSD"
```

This will create three PKCS12 files (`ds.p12`, `mrc.p12` and `osd.p12`), each containing the private key and certificate for the respective service. The passwords chosen when asked must be set as a property in the corresponding service configuration file.

Importing trusted certificates from PEM into a JKS

The certificate (or multiple certificates) from your CA (or CAs) can be imported into a Java Keystore (JKS) using the Java keytool which comes with the Java JDK or JRE.

Execute the following steps for each CA certificate using the same keystore file.

```
$> keytool -import -alias rootca -keystore trusted.jks \
   -trustcacerts -file ca-cert.pem
```

This will create a new Java Keystore `trusted.jks` with the CA certificate in the current working directory. The password chosen when asked must be set as a property in the service configuration files.

Note: If you get the following error

```
$> keytool error: java.lang.Exception: Input not an X.509 certificate
```

you should remove any text from the beginning of the certificate (until the `---BEGIN CERTIFICATE---` line).

Sample Setup

Users can easily set up their own CA (certificate authority) and create and sign certificates using `openssl` for a test setup.

1. Set up your test CA.

   (a) Create a directory for your CA files

   ```
   $> mkdir ca
   ```

   (b) Create a private key and certificate request for your CA.

   ```
   $> openssl req -new -newkey rsa:1024 -nodes -out ca/ca.csr \
      -keyout ca/ca.key
   ```

   Enter something like XtreemFS-DEMO-CA as the common name (or something else, but make sure the name is different from the server and client name!).

   (c) Create a self-signed certificate for your CA which is valid for one year.

   ```
   $> openssl x509 -trustout -signkey ca/ca.key -days 365 -req \
      -in ca/ca.csr -out ca/ca.pem
   ```

   (d) Create a file with the CA's serial number

   ```
   $> echo "02" > ca/ca.srl
   ```

2. Set up the certificates for the services and the XtreemFS Client.
   Replace *service* with `dir`, `mrc`, `osd` and `client`.

   (a) Create a private key for the service.
       Use XtreemFS-DEMO-*service* as the common name for the certificate.

   ```
   $> openssl req -new -newkey rsa:1024 -nodes    -out service.req    -keyout
   ```

   (b) Sign the certificate with your demo CA.
       The certificate is valid for one year.

   ```
   $> openssl x509 -CA ca/ca.pem -CAkey ca/ca.key    -CAserial ca/ca.srl -req
   ```

   (c) Export the service credentials (certificate and private key) as a PKCS#12 file.
       Use "passphrase" as export password. You can leave the export password empty for the XtreemFS Client to avoid being asked for the password on mount.

   ```
   $> openssl pkcs12 -export -in service.pem -inkey    service.key    -out se
   ```

   (d) Copy the PKCS#12 file to the certificates directory.

   ```
   $> mkdir -p /etc/xos/xtreemfs/truststore/certs
   $> cp service.p12 /etc/xos/xtreemfs/truststore/certs
   ```

3. Export your CA's certificate to the trust store and copy it to the certificate dir.
   You should answer "yes" when asked "Trust this certificate".
   Use "passphrase" as passphrase for the keystore.

```
$> keytool -import -alias ca -keystore trusted.jks\
    -trustcacerts -file ca/ca.pem
$> cp  trusted.jks /etc/xos/xtreemfs/truststore/certs
```

4. Configure the services. Edit the configuration file for all your services. Set the
   following configuration options (see Sec. 3.2 for details).
   ```
   ssl.enabled = true
   ssl.service_creds.pw = passphrase
   ssl.service_creds.container = pkcs12
   ssl.service_creds = /etc/xos/xtreemfs/truststore/certs/service.p12
   ssl.trusted_certs = /etc/xos/xtreemfs/truststore/certs/trusted.jks
   ssl.trusted_certs.pw = passphrase
   ssl.trusted_certs.container = jks
   ```

5. Start up the XtreemFS services (see Sec. 3.3.1).

6. Create a new volume (see Sec. 3.3.3 for details).

```
$> xtfs_mkvol --pkcs12-file-path=\
    /etc/xos/xtreemfs/truststore/certs/client.p12 localhost/test
```

7. Mount the volume (see Sec. 4.2 for details).

```
$> xtfs_mount --pkcs12-file-path=\
    /etc/xos/xtreemfs/truststore/certs/client.p12 localhost/test /mnt
```

3.3 Management

This section covers all tools and functionality for XtreemFS management and tracing. In general, the use of management tools is restricted to superusers.

3.3.1 Starting and Stopping the XtreemFS services

If you installed a *pre-packaged release* you can start, stop and restart the services with the `init.d` scripts:

```
$> /etc/init.d/xtreemfs-ds start
$> /etc/init.d/xtreemfs-mrc start
$> /etc/init.d/xtreemfs-osd start
```

or

```
$> /etc/init.d/xtreemfs-ds stop
$> /etc/init.d/xtreemfs-mrc stop
$> /etc/init.d/xtreemfs-osd stop
```

Note that the Directory Service should be started first, in order to allow other services an immediate registration. Once a Directory Service and at least one OSD and MRC are running, XtreemFS is operational.

3.3.2   Web-based Status Page

The XtreemFS services all have a HTML status page which can be used to check if
the service is working correctly (Fig. 3.1). It can be displayed by opening the service
URL in your favorite web browser, e.g.
`http://my-mrc-host.com:30636/`. Make sure to use the right port, see `http_port`
in the service config file.

Figure 3.1: OSD status web page

3.3.3   Creating Volumes

Volumes can be created with the `xtfs_mkvol` command line utility. Please see `man`
`xtfs_mkvol` for a full list of options and usage.

When creating a volume, it is recommended to specify the access policy (see Sec.
2.3.4). If not specified, POSIX permissions/ACLs will be chosen by default. Access
policies cannot be changed afterwards.

An OSD selection policy (see Sec. 2.3.1) can also be specified per volume, but can
be changed anytime. By default, a random selection of available OSDs is assigned to
newly created files.

In addition, it is recommended to set a default striping policy (see Sec. 2.3.3). If no
per-file or per-directory default striping policy overrides the volume's default striping
policy, the volume's policy is used for new files and directories. If no volume policy
is explicitly defined, a RAID0 policy with a stripe size of 128kB and a width of 1 will
be assigned to the volume.

An example call to `xtfs_mkvol` for creating a volume with POSIX ACLs, 256kB
stripe size and a stripe width of 1 (which means no striping):

```
$> xtfs_mkvol -a POSIX -p RAID0 -s 256 -w 1 \
   my-mrc-host.com:32636/myVolume
```

3.3.4 Deleting Volumes

The `xtfs_rmvol` tool can be used to delete a volume. This also *deletes all files and data on that volume*! Please see `man xtfs_rmvol` for a full list of options and usage.

Example call to `xtfs_rmvol` to delete `myVolume`:

```
$> xtfs_rmvol my-mrc-host.com:32636/myVolume
```

Volume deletion is restricted to volume owners and superusers.

3.3.5 MRC Database Conversion

The format in which the MRC stores its data on disk may change with future XtreemFS versions. In order that XtreemFS server components may be updated without losing the whole content of the file system, it is possible to create a version-independent XML representation of the metadata stored in MRC database.

Such an XML representation can e.g. be created as follows:

```
$> xtfs_mrcdbtool -mrc my-mrc-host.com:32636 \
   dump /tmp/dump.xml
```

This call will create a file `dump.xml` containing the entire MRC database content in the `/tmp` directory at `my-mrc-host.com`.

To restore an MRC database from a dump, execute

```
$> xtfs_mrcdbtool -mrc my-mrc-host.com:32636 \
   restore /tmp/dump.xml
```

This will restore the database stored in `/tmp/dump.xml` at `my-mrc-host.com`. Note that for safety reasons, it is only possible to restore a database from a dump if the database of the running MRC does not have any content. To restore an MRC database, it is thus necessary to delete all MRC database files before starting the MRC.

3.3.6 Scrubbing and Cleanup

In real-world environments, errors occur in the course of creating, modifying or deleting files. This can cause corruptions of file data or metadata. Such things happen e.g. if the client is suddenly terminated, or loses connection with a server component. There are several such scenarios: if a client writes to a file but does not report file sizes received from the OSD back to the MRC, inconsistencies between the file size stored in the MRC and the actual size of all objects in the OSD will occur. If a client deletes a file from the directory tree, but cannot reach the OSD, orphaned objects

will remain on the OSD. If an OSD is terminated during an ongoing write operation, file content will become corrupted.

In order to detect and, if possible, resolve such inconsistencies, tools for scrubbing and OSD cleanup exist. To check the consistency of file sizes and checksums, the following command can be executed:

```
$> xtfs_scrub -dir oncrpc://my-dir-host.com:32638 xtreemfsVolume
```

This will scrub each file in the volume `myVolume`, i.e. check file size consistency and set the correct file size on the MRC, if necessary, and check whether an invalid checksum in the OSD indicates a corrupted file content. The `-dir` argument specifies the directory service that will be used to resolve service UUIDs. Please see `man xtfs_scrub` for further details.

A second tool scans an OSD for orphaned objects, which can be used as follows:

```
$> xtfs_cleanup -dir oncrpc://localhost:32638 \
    uuid:u2i3-28isu2-iwuv29-isjd83
```

The given UUID identifies the OSD to clean and will be resolved by the directory service defined by the `-dir` option (localhost:32638 in this example). The process will be started and can be stopped by setting the option `-stop`. To watch the cleanup progress use option `-i` for the interactive mode. For further informations see `man xtfs_cleanup`.

Chapter 4

The XtreemFS Client

4.1 Installation

As for the XtreemFS Services, there are two different installation sources for the XtreemFS Client: *pre-packaged releases* and *source tarballs*.

4.1.1 Prerequisites

For both installations you need FUSE 2.6 or newer, openSSL 0.9.8 or newer and a Linux 2.6 kernel. For optimal performance we suggest to use FUSE 2.8 with a kernel version 2.6.26 or newer.

To build the XtreemFS Client from sources, you need the openSSL headers (e.g. openssl-devel package), python $\geq$ 2.4, and gcc-c++ $\geq$ 4.2.

4.1.2 Installing from Pre-Packaged Releases

On RPM-based distributions (RedHat, Fedora, SuSE, Mandriva, XtreemOS) you can install the package with

```
$> rpm -i xtreemfs-client-1.0.x.rpm
```

For Debian-based distributions, please use the `.deb` package provided and install it with

```
$> dpkg -i xtreemfs-client-1.0.x.deb
```

4.1.3 Installing from Sources

Extract the tarball with the sources. Change to the top level directory and execute

```
$> make client
```

4.2   Mounting and Un-mounting

Before mounting XtreemFS volumes, please ensure that the FUSE kernel module is loaded. Please check your distribution's manual to see, if users must be in a special group (e.g. `trusted` in openSUSE) to be allowed to mount FUSE.

```
$> su
Password:
#> modprobe fuse
#> exit
```

To mount an XtreemFS volume use the `xtfs_mount` tool.

```
$> xtfs_mount remote.dir.machine/myVolume /xtreemfs
```

`remote.dir.machine` describes the host with the Directory Service at which the volume is registered; `myVolume` is the name of the volume name to be mounted. `/xtreemfs` is the directory on the local file system to which the XtreemFS volume will be mounted. For more options, please refer to `man xtfs_mount`.

The client will immediately go into background and won't display any error messages. Use the `-f` option to prevent the mount process from going into background and get all error messages printed to the console.

Access to a FUSE mount is usually restricted to the user who mounted the volume. To allow the root user or any other user on the system to access the mounted volume, the FUSE options `-o allow_root` and `-o allow_other` can be used with `xtfs_mount`. They are, however, mutually exclusive. In order to use these options, the system administrator must create a FUSE configuration file `/etc/fuse.conf` and add a line `user_allow_other`.

To check that a volume is mounted use the `mount` command. It ouputs a list of all mounts in the system. XtreemFS volumes are listed as `type fuse`:

```
/dev/fuse on /xtreemfs type fuse (rw,nosuid,nodev,user=userA)
```

Volumes are unmounted using the `xtfs_umount` tool.

```
$> xtfs_umount /xtreemfs
```

4.3   Reading XtreemFS-specific File Info

In addition to the regular file system information provided by the `stat` Linux utility, XtreemFS provides the `xtfs_stat` tool which displays XtreemFS specific information for a file or directory.

```
$> cd /xtreemfs
$> echo 'Hello World' > test.txt
$> xtfs_stat remote.mrc.machine/myVolume/test.txt
```

will produce output similar to the following:

```
-----
  type          = directory
  nlink         = 1
  size          = 0
  atime         = 2009-05-05T09:44:16.000Z
  mtime         = 2009-05-05T10:10:06.000Z
  ctime         = 2009-05-05T10:10:06.000Z
  owner user id = xtreemfs
  owner group id = users
  file_id       = 0fa6c684-4885-48b1-a678-babdfae8db37:1
  truncate epoch = 2031654
```

The fileID is the unique identifier of the file used on the OSDs to identify the file's objects. The owner/group fields are shown as reported by the MRC, you may see other names on your local system if there is no mapping (i.e. the file owner does not exist as a user on your local machine). Finally, the XtreemFS replica list shows the striping policy of the file, the number of replicas and for each replica, the OSDs used to store the objects.

4.4 Changing Striping Policies

It is not (yet) possible to change the striping policy of an existing file, as this would require moving and reformatting data among OSDs. However, individual striping policies can be assigned to new files (i.e. empty files) by changing the default striping policy of the parent directory or volume. For this purpose, XtreemFS provides the `xtfs_sp` tool. The tool can be used to change the striping policy that will be assigned to newly created files.

```
$> xtfs_sp --set -p RAID0 -w 4 -s 256 /xtreemfs
```

In addition, the tool can be used to retrieve the default striping policy of a volume or directory.

```
$> xtfs_sp --get /xtreemfs
```

The output will be similar to the following:

```
file:         /xtreemfs
policy:       STRIPING_POLICY_RAID0
stripe-size:  4
width (kB):   256
```

When creating a new file, XtreemFS will first check whether a default striping policy has been assigned to the parent directory. If this is not the case, the default striping policy for the volume will be used as the striping policy for the new file. Changing a volume's or directory's default striping policy requires superuser access rights or ownership of the directory or volume.

4.5   Changing Replica Selection Policy

The replica selection policy can only be set for the entire volume. The policies
are described in Sec. 2.3.2. To show and modify the policy for a volume, use the
`xtfs_repl` tool.

```
$> xtfs_repl --rsp_get /xtreemfs
```

displays the current replica selection policy used for the volume. To change the
policy to use the datacenter map for a volume, use

```
$> xtfs_repl --rsp_set dcmap /xtreemfs
```

If you want to use a custom (i.e. plug-in) policy, pass the id instead of the name.

4.6   Read-Only Replication

Read-only replication allows you to create copies of files on several OSDs. Since
the files are immutable (read-only) there is no overhead for coordinating the replicas.
XtreemFS supports partial and full replicas. A partial replica will only fetch the data
requested by a client (on demand). A full replica does the same, but it also fetches
the data in the background without client requests until all data of the file has been
replicated (background replication).

With XtreemFS all replicas are initially empty and can be used immediately by appli-
cations. If a replica does not have the requested data, it fetches it from another replica
and saves it locally for future requests. This helps you to reduce start-up times, and
saves network bandwidth and disk usage.

The XtreemFS client gets a list of replicas from the MRC when opening a file. This
list is sorted according to the volume's replica selection policy, i.e. the first replica in
the list is the "best" replica for the client. The client will use the first replica in the
list and automatically switches to the next one if it is not reachable.

So far XtreemFS only supports to manage replicas manually. Before the replication
can be used the file must be marked as read-only with the following command:

```
$> xtfs_repl --set_readonly local-path-of-file
```

After a file is marked as read-only replicas can be added. The tool supports differend
replica creation modes. The automatic mode retrieves a list of OSDs from the MRC
and chooses the best OSD according to the current replica selection policy. There is
also an interactive mode for selecting OSDs yourself. Finally, you can select an OSD
by sepcifying its UUID on the command line.

By default, partial replicas will be created. To create a full replica the option `-full`
must be set.

XtreemFS supports different transfer strategies which has an big impact on the speed
of the replication and the order in which objects are fetched. A transfer strategy
must be chosen for each replica.

To create a full replica and usage of the random transfer strategy, the following command must be used. The OSDs are selected according to the volume's replica selection policy. See Sec. 2.3.2 for a description of the policies.

```
$> xtfs_repl --add_auto --full --strategy random local-path-of-file
```

To list all replicas and OSDs of the file use:

```
$> xtfs_repl -l local-path-of-file
```

Removing also supports muliple modes. Again an interactive mode and a mode which is choosing the replica randomly. To remove a specific replica, the head-OSD (first OSD) of this replica must be given as an argument. At least one complete replica must exist, so the tool will not remove a complete replica, if no others exists. To remove the just created replica use the following command and change head-osd to the one listed by `xtfs_repl -l`:

```
$> xtfs_repl -r head-OSD local-path-of-file
```

For further options see:

```
$> xtfs_repl -h
```

Chapter 5

Troubleshooting and Support

5.1 Logfiles

The logfiles for the XtreemFS services are located in `/var/log/xtreemfs`. The client generates no output, unless the `-f` and `-d INFO` options are specified.

5.2 Support

Please visit the XtreemFS website at www.XtreemFS.org for links to the user mailing list and IRC channel.

5.3 Troubleshooting

Problem: The client hangs when opening/copying/creating a file but operations like `ls` or `mkdir` work.

Solution: This problem can occurr when an OSD uses a UUID which resolves to an address that the client cannot (correctly) resolve. If you use e.g. `localhost:32640` as the UUID for the OSD, the client will try to contact the local machine instead of the machine on which the OSD runs. Check the status page of your Directory Service and check the UUID of the OSDs.

Problem: `xtfs_mount` does not print an error message but the volume is not mounted (i.e. not listed in the output of `mount`).

Solution: The client `xtfs_mount` automatically goes into background and does not print any error messages or warinings. Use the `-f` flag when mounting to prevent the client from going into background. All error messages will be printed to the console.

Appendix A

XtreemOS Integration

XtreemFS Security Preparations

XtreemFS can be integrated in an existing XtreemOS VO security infrastructure. XtreemOS uses X.509 certificates to authenticate users in a Grid system, so the general setup is similar to a normal SSL-based configuration.

Thus, in an XtreemOS environment, certificates have to be created for the services as a first step. This is done by issuing a *Certificate Signing Request (CSR)* to the RCA server by means of the `create-server-csr` command. For further details, see the Section Using the RCA in the XtreemOS User Guide.

Signed certificates and keys generated by are RCA infrastructure are stored locally in PEM format. Since XtreemFS services are currently not capable of processing PEM certificates, keys and certificates have to be converted to PKCS12 and Java Keystore format, respectively.

Each XtreemFS service needs a certificate and a private key in order to be run. Once they have created and signed, the conversion has to take place. Assuming that certificate/private key pairs reside in the current working directory for the Directory Service, an MRC and an OSD (`ds.pem`, `ds.key`, `mrc.pem`, `mrc.key`, `osd.pem` and `osd.key`), the conversion can be initiated with the following commands:

```
$> openssl pkcs12 -export -in ds.pem -inkey ds.key \
   -out ds.p12 -name "DS"
$> openssl pkcs12 -export -in mrc.pem -inkey mrc.key \
   -out mrc.p12 -name "MRC"
$> openssl pkcs12 -export -in osd.pem -inkey osd.key \
   -out osd.p12 -name "OSD"
```

This will create three PKCS12 files (`ds.p12`, `mrc.p12` and `osd.p12`), each containing the private key and certificate for the respective service.

XtreemFS services need a *trust store* that contains all trusted Certification Authority certificates. Since all certificates created via the RCA have been signed by the XtreemOS CA, the XtreemOS CA certificate has to be included in the trust store. To create a new trust store containing the XtreemOS CA certificate, execute the following command:

```
$> keytool -import -alias xosrootca -keystore xosrootca.jks \
   -trustcacerts -file \
   /etc/xos/truststore/xtreemosrootcacert.pem
```

This will create a new Java Keystore xosrootca.jks with the XtreemOS CA cer-
tificate in the current working directory. The password chosen when asked will later
have to be added as a property in the service configuration files.

Once all keys and certificates have been converted, the resulting files should be
moved to /etc/xos/xtreemfs/truststore/certs as root:

```
# mv ds.p12 /etc/xos/xtreemfs/truststore/certs
# mv mrc.p12 /etc/xos/xtreemfs/truststore/certs
# mv osd.p12 /etc/xos/xtreemfs/truststore/certs
# mv xosrootca.jks /etc/xos/xtreemfs/truststore/certs
```

For setting up a *secured* XtreemFS infrastructure, each service provides the following
properties:

```
# specify whether SSL is required
ssl.enabled = true

# server credentials for SSL handshakes
ssl.service_creds = /etc/xos/xtreemfs/truststore/certs/\
service.p12
ssl.service_creds.pw = xtreemfs
ssl.service_creds.container = pkcs12

# trusted certificates for SSL handshakes
ssl.trusted_certs = /etc/xos/xtreemfs/truststore/certs/\
xosrootca.jks
ssl.trusted_certs.pw = xtreemfs
ssl.trusted_certs.container = jks
```

service.p12 refers to the converted file containing the credentials of the respective
service. Make sure that all paths and passphrases (xtreemfs in this example) are
correct.

Appendix B

Command Line Utilities

xtfs_cleanup Deletes orphaned objects on an OSD or creates new metadata objects for orphaned files.

xtfs_lsvol Lists the volumes on an MRC.

xtfs_mkvol Creates a new volume on an MRC.

xtfs_mount The XtreemFS client which mounts an XtreemFS volume locally on a machine.

xtfs_mrcdbtool Dumps an XML representation of the MRC database to a given directory in the MRC's local file system.

xtfs_repl Displays and controls the replication of files and the replication policies.

xtfs_rmvol Deletes a volume and all files on that volume from the MRC and the OSDs.

xtfs_stat Displays XtreemFS specific file information such as the striping policy and the OSDs.

xtfs_sp Displays and modifies the striping policy for a file, or the default striping policy for directories and volumes.

xtfs_scrub Examines all files in a volume for incorrect file sizes and checksums. In case of incorrect file sizes, file sizes are corrected at the MRC.

xtfs_umount Un-mounts a mounted XtreemFS volume.

Index