Project no. IST-033576

XtreemOS

Integrated Project
BUILDING AND PROMOTING A LINUX-BASED OPERATING SYSTEM TO SUPPORT VIRTUAL ORGANIZATIONS FOR NEXT GENERATION GRIDS

Advanced Guide: Installation and Administration

XtreemOS Technical Report # –NA–

Massimo Coppola

Report Registration Date: —

Version 0.01 / Last edited by Massimo Coppola / November 9, 2009

Project co-funded by the European Commission within the Sixth Framework Programme		
Dissemination Level		
PU	Public	√
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Revision history:

Version	Date	Authors	Institution	Section affected, comments
0.1	28/04/09	Massimo Coppola		Initial document
0.2	02/06/09	Ian Johnson	STFC	Revised descriptions of X-VOMS, corrected some minor typos

Contents

Chapter 1

Introduction

XtreemOS is a Linux-based operating system providing native support for virtual organizations (VOs) in next-generation grids. It represents a new approach to Grid and distributed computing, which overcomes many of the limitations of current middlewares. The XtreemOS Guides will let you make the first steps in using the system, and will lead you through all different installation phases, regardless that you are installing a full Grid platform, attempting an experiment with a few machines in your system, or just giving it a try on a Virtual Machine.

To better suit the different targets the Guides are split in a User's Guide, an Administrator's Guide, with several other related documents referenced from here.

1.1 What is in the Guides

The XtreemOS Administrator's and User's guides provide comprehensive guidelines for installing, configuring and operating an XtreemOS system.

The target of the **User Guide** are beginner users of the XtreemOS systems, who either:

1. already have access to an installed machine that is part of the XtreemOS network,

2. are trying a ready-made system-disk image on top of virtualization software such as VirtualBox, or

3. are using a live boot device that does not need installation (currently a CD-ROM, but USB flash device is planned).

User-level functionalities are explained and shown with examples, and common problems and basic configuration issues are either explained, or referenced in the Administration Guide.

The **Administration Guide** describes at length XtreemOS sytems installation procedures and system configuration, both using the graphical installer and discussing individual subsystem configuration files.

Both guides focus on the workstation and cluster (XtreemOS-SSI) flavours of XtreemOS, the distinction among them being almost completely confined to the Administration Guide. The mobile XtreemOS flavour is described in a separate document, which has as a prerequisite all the introductory part and the general concepts of the User's Guide.

1.1.1 Contents of the User Guide

- Chapter 2 describes what an XtreemOS grid looks like, its architecture and the core, resource and client services that must be deployed in order to operate the system
- Chapter 3 explains how to obtain the latest XtreemOS install CD and its software packages
- Chapter 4 describes how to operate an XtreemOS grid, its commands and the most common use cases
- Chapter 5 collects examples of practical use with small applications
- Chapter 6 collects frequently asked questions and common issues encountered during XtreemOS usage.

1.1.2 Contents of the Admininstration Guide

Chapters 2 and 3 are the same as the User Guide.

- Chapter 4.1 describes system installation from scratch using the graphical installer
- Chapter 5 explains how to run XtreemOS in just one machine, so you can get familiar with the main concepts and operations. TO BE REMOVED
- Chapter 6 explains how to set up a minimal XtreemOS grid in a quick and easy way
- Chapter 7 explains how to install and configure each XtreemOS service, in order to set up a full operative XtreemOS grid

- Chapter **??** collects frequently asked questions and common issues encountered during XtreemOS installation and administration.

Beside thes two guides, it is worth mentioning here the document which specifically targets the XtreemOS system flavour for mobile devices.

1.1.3 Note on conventions for showing command input

In the figures in this guide that show command input, commands which need root privileges to run are prefixed by a prompt 'Root# ' or '(root)# '. Long lines of command input may be entered on more than one line; the command is continued over to the next line by entering the backslash symbol so that the shell treats the input as one line. This is neccessary because of the constraints on column width in this Guide.

Consider the following command:

```
(root)# urpmi.addmedia xtreemos \
MIRROR/MandrivaLinux/devel/xtreemos/2009.0/i586/media/xtreemos/release/
```

This could entered to the shell by typing the backslash then a carriage return, and then entering the rest of the command. Alternatively, you could type the command as a single line.

1.2 Getting started

XtreemOS is a Linux-based operating system providing native support for virtual organizations (VOs) in next-generation grids. Unlike the traditional, middleware-based approaches, it is a prominent goal to provide seamless support for VOs, on all software layers involved, ranging from the operating system of a node, via the VO-global services, up to direct application support.

XtreemOS offers the following features and functionalities:

Application execution management

- Job basic manipulation : submit jobs, check job status, wait for job finalization, control jobs
- Send events to jobs
- Check, identify, reserve resources that match job requirements

- No global job scheduler is needed, the provided job management system is distributed

Virtual organization management

- A set of well-integrated security services
- Management of certificates
- Management of users, groups, roles, policies
- Management of resources
- Management of the complete lifecycle of VOs, operated via web and command-line interfaces[1]

Data management via the XtreemFS Grid file system

- Complete file system functionality across a Grid platform
- POSIX compliance
- Remote Grid authentication and authorization of users via XtreemOS certificates
- Volume creation and mount on any XtreemOS resource
- File striping and data replication, Grid file system monitoring

In the two System Guides you will see how these features are provided and how the necessary software is deployed within a Linux system. Of course, the most common ways of using XtreemOS will be connecting to an already existing XtreemOS platform, joining an XtreemOS platform with you machine, and building your own platform by installing a Linux distribution (like the Mandriva-XtreemOS) that already includes all the necessary XtreemOS packages by default.

1.2.1 Heterogeneous Devices

XtreemOS also integrates different *flavours* of the operating system, each one a different system distribution providing the XtreemOS functionalities on one of the different computing architectures that are commonly used in VOs.

- Stand-alone PCs (single CPU, or SMP, or multi-core) are supported by the standard XtreemOS distribution for laptops, which is the more heavily gorunded in the Linux operating system.

[1]Note that managing the VO can be achieved via simple, scriptable command-line tools by the Administrator, when logged on the core node running the X-VOMS database. For maximum flexibility of use, the web interface can be made accessible from everywhere.

- For clusters of Linux machines, the XtreemOS-SSI flavour combines VO support with a single system image (SSI) functionality. A custom, SSI-enabled kernel is adopted to provide the functionality of a single Linux SMP machine as big as the supporting cluster, with a very large memory space, fully shared devices, and custom support for high-performance file system management as well as near-zero overhead transparent process migration.

  Apart from providing the abstraction of a single huge computing resource, the SSI platfrom is seamlessly integrated within the XtreemOS platform, and it is distributed together with the single machine version.

- Linux mobile devices, finally, can also easily join the XtreemOS platform. The XtreemOS-MD flavour provides them with VO support and specially-tailored, lightweight services for application execution, common data access, and user management.

1.2.2 OGSA Compliance

In terms of the Open Grid Service Architecture (OGSA), as shown in the figure, XtreemOS is providing support on all layers involved in a virtual organization:

- On the *fabric layer*, XtreemOS provides VO-support by Linux kernel modules.
- On the *connectivity layer*, XtreemOS provides VO membership support for (compute and file) resources, application programs, and users.
- On the *resource layer*, XtreemOS provides application execution management.
- On the *collective layer*, XtreemOS provides the XtreemFS file system, and VO management services.
- On the *application layer*, finally, XtreemOS provides runtime support via the Simple API for Grid Applications (SAGA), next to native POSIX interfaces.

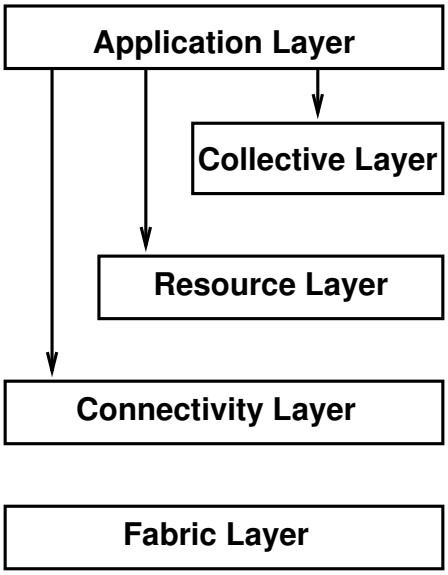

Figure 1.1: The layered Grid middleware architecture.

Chapter 2

Overview

In this chapter an overall description of the the XtreemOS system is given. We start from the three different kinds of nodes that compose the distributed platform, explaining their role and the additional system modules of XtreemOS that each kind needs with respect to a standard Linux system.

Section 2.2 discusses the different kind of cryptographic identity certificates that are exploited within XtreemOS to identify authorized users, and authenticate resources and services within the system.

Finally, section 2.3 will give a preliminary bird-eye description of the XtreemOS installation procedure.

2.1 Architecture

Within any XtreemOS system there are 3 different kinds of nodes (also called *configurations*):

- **Core** nodes, which are nodes providing the services that allow other nodes to provide resources to the XtreemOS system.
- **Resource** nodes, which are nodes providing resources to the XtreemOS system.
- **Client** nodes, which are nodes accessing the XtreemOS system without providing resources to it.

This is a logical functional division, whose purpose is to ease structuring the system and understandng its behaviour. Nothing precludes a certain machine from playing more than a single role, e.g. acting both as a client node and as a resource node, and this will be sometimes exactly the expected behaviour.

The important distinction in the platform is the one among core nodes and all other nodes. As it is for administrative accounts in a conventional operating system, it is generally advised to disallow arbitrary access on core nodes to users other than system administrators. Most services running on core nodes are critical for the functionality and security of the whole platform, thus security violations as well as core-node downtime would have a negative impact on the XtreemOS system. This is nevertheless possible and allows to even demo-install XtreemOS on a single machine that provides all the services by itself.

Returning to the main topic, the different purpose of each configuration also determines which components and services should be present on it. In the following sections, we list and briefly describe these services for the Core, Resource and Client node configurations.

2.1.1 Core services

Core VOM services provide the certificate issuing and signing authorities for the users and the resources. They also provide the means to edit VO-level policies and making policy decisions. They are the following:

X-VOMS is the database containing all the information related to the virtual organizations living in the XtreemOS system

VOLifeCycle is a web frontend to manage the full lifecycle of virtual organizations, users and resources in the XtreemOS system. This includes a web interface for creating XOS-Certificates and private keys.

Credential Distribution Authority (CDA) server provides a way for the user to get an XOS-Certificate via the command-line CDA client.

Resource Certification Authority (RCA) server is used to get a resource certificate, in order to authenticate the resource in the XtreemOS system. RCA server comprises of the following: the RCA server logic, the front-end for both the RCA server logic and the RCA database implemented for DIXI, and the service certificate signed by the organisations root certification authority or another authority with the organisations root CA in the signature chain

Virtual Organization Policy Service (VOPS) serves requests and forwards answers from/to resource discovery services and digitally signs its decisions before forwarding responses back to services. VOPS enforces user requests against VO level policies for gaining access to specific resource nodes. VOPS is a standalone security service

which provides Policy Administration Point (PAP), Policy Information Point (PIP), Policy Decision Point (PDP) to other XtreemOS services (e.g. AEM)

Core AEM services oversee the job submission process, select the nodes for the jobs and schedule their execution, and book-keep the jobs submitted so far.

Core XtreemFS services keep control of metadata as well as storage devices committed to the system and are the following:

Metadata Replica Catalog (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.

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

2.1.2 Resource services

Resource AEM services or **ExecMng** receive and execute the scheduled jobs, monitor the resources and the job execution. Moreover, resource-level information services support distributed information management, setting up essential overlay networks within the platform and providing higly scalable management of resources. The latest category includes:

Service/Resource Discovery System provides the node services related to distributed information management

Resource Selection Service provides distributed, scalable and efficient resource location.

ADS provides an implementation of the Overlay Weaver and Scalaris DHTs that are compatible with the Application Directory Service (ADS) as provided by the SRDS package

Resource XtreemFS services or **Object Storage Devices (OSDs)** store arbitrary objects of files; clients read and write file data on OSDs.

2.1.3 Client services

Client VOM services are the following:

Credential Distribution Authority (CDA) client creates a private key and accesses the CDA server in order to get the user's XOS-Certificate, containing their public key and VO attributes.

ssh-xos allows login of Grid users into the system, with their XOS-certificate and access to their home XtreemFS volumes.

Client AEM services are the following:

XATI and C-XATI provide the service clients, letting the user perform administration tasks and exploit the virtual organization services.

xconsole_dixi provides a command-line interface for retrieving the available resources and submitting jobs.

XtreemFS 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 Certificates used in XtreemOS

XtreemOS uses X.509 v3 certificates to provide a Public Key Infrastructure for authentication. The certificates carry the public key of an entity and may be distributed to other parties that wish to establish the authentiticy of the key holder. The corresponding private key is secured by the key holder (the user) and is used to sign messages from them that can be verified by the public key. The conventional suffix for a certificate file is `.crt`, for a private key it is `.key`. Certificate signing requests (CSR files) have a `.csr` suffix.

All XtreemOS users need the XtreemOS root CA certificate installed on their machine. This is something that should be done by their system administrator.

End-users mostly deal with the XOS-Certificate. This identifies them to the system, and carries details of which VOs they belong to. In addition, if an end-user needs to use the VOLife web service via SSL (recommneneded), they can either import the XtreemOS root CA certificate into their browser, or accept the certificate presented by the VOLife web application the first time they connect to it using SSL.

System administrators of core nodes need to request service certificates from the XtreemOS root CA. Service certificates are currently used in the following ways;

- to authenticate servers to clients
- to authenticate clients to servers
- to sign a server response sent back to a client.

Most of the applications are services that run on core nodes. The XtreemFS mount client can run on any kind of node, and can authenticate to an XtreemFS server by sending the identity of the machine it is running on (using an service certificate), or it can send the user's XOS-Certificate to the XtreemFS server to prove the user's identity.

Resource administrators describe their resources by requesting resource certificates from the Resource Certificate Authority.

The fundamental step in setting up the XtreemOS certificates is for the root CA administrator to create the root certificate and corresponding private key, which is then used to sign requests for service certificates from system administrators. This root certificate has to be installed on every machine in the XtreemOS Grid, to enable checking of received service certificates and XOS-certificates.

- The XtreemOS root certificate needs to be installed by a system administrator on every machine in an XtreemOS Grid. It goes into `/etc/xos/truststore/certs/xtreemos.crt`.
- The default location for the user's XOS certificate is `$HOME/.xos/truststore/certs/user.crt`, with the corresponding private key in `$HOME/.xos/truststore/private/user.key`.
- Host certificates on a core node reside in `/etc/xos/truststore/certs/<service>.crt`. The corresponding private key is in `/etc/xos/truststore/private/<service>.key`. the `<service>` can actually be `cda`, `vops`, `rca`.
- The storage of resource certificates is handled transparently by the AEM RCA client.

2.3 Installation and configuration process

The installation and configuration process of a XtreemOS grid should follow the following order (please refer to section **ADM**:6 for minimal setup routine, or to section **ADM**:7 for a more detailed view of the process):

1. Install XtreemOS and configure the Root CA on on a single machine (see section **ADM**:7.1). This should not be a multiuser machine, and should be in a physically secure location. This machine should, ideally, not run

Table 2.1: XtreemOS Certificate types

Certificate Type	Used By	Purpose	Created-By, See Doc Sect.
Root certificate	CA admin, End-user	Public Key of the XtreemOS root CA. Can be imported into browser as a trusted certificate in order to access VOLife via SSL (https)	Root CA, §**ADM**:7.1
XOS certificate	End-user	Contains public key and VO attributes.	CDA server, §**USE**:4.1.1
Service certificate	System admin	Used to authenticate the identity of applications running on core nodes, or to authenticate an XtreemFS client).	Root CA, §**ADM**:7.3.5
Resource certificate	AEM	Attest to the resources provided by a resource node	RCA Server, §**ADM**:7.3.4

any other XtreemOS services. To provide the ultimate in security (in order to avoid compromise of the Root CA), the machine need not be networked at all.

2. Install XtreemOS on a machine, and configure it as a core node to run the VO management services. Currently, the CDA and VOLife services have to be on the same machine as the X-VOMS database, unless remote database access has been configured. (Such configuration is outside the scope of this Guide.)

   Install the following VO Management services:

   - X-VOMS (section **ADM**:7.3.1).
   - VOLife (section **ADM**:7.3.3).
   - CDA (section **ADM**:7.3.2).

   This step requires using the Root CA to generate service certificates for the services (see section **ADM**:7.1.3).

   Upon completing this step, you are in a position to start the X-VOMS, VOLife and CDA services. Users can register with the VOLife web application, create and join VOs, and define the groups they belong to, and roles they have, in VOs. Users can obtain XOS-Certificates (by using the VOLife webapp or CDA client) which contain their VO attributes and public key.

3. Install and configure node-level information services. These have to be present on the same machine(s) configured in 2. They build up overlay

networks and provide services required by the AEM[1].

- RSS (section **ADM**:7.5)
- SRDS (section **ADM**:7.6).

Upon completing this step you will be able to install node-level AEM services.

4. Install and configure AEM services. These could go on the same machine configured in 2 Above or a different machine.
   - RCA (section **ADM**:7.3.4).
   - VOPS (section **ADM**:7.3.6).
5. Install and configure XtreemFS servers (section **ADM**:7.7).

[1]Note: currently the SRDS and RSS services are implemented as node-level AEM services.

Chapter 3

Getting XtreemOS

This chapter is about getting XtreemOS binaries for installation, and how to proceed in the different situations (install from CD, repositories, disk images). You can find in chapter 8 a quick-reference of network URLs where to get binaries, sources, documentation and support.

3.1 The Install CD

The XtreemOS 2nd Public Release is based on the Mandriva Linux 2009.0 stable release. It includes all needed basesystem software needed to be able to run a basic GNU/Linux system with an X server, and all tools needed to be able to rebuild packages or software. The second CD includes all tools developped by the XtreemOS consortium, and all sources of those software as Source RPM packages.

The XtreemOS Install CDs are available from Mandriva mirrors. Several mirrors are available, you can find a list on this page : http://www.xtreemos.eu/software/mirror-websites

The ISOs are located in the MandrivaLinux/devel/iso/xtreemos directory.

Any further information for XtreemOS 2nd Public Release?

3.2 Disk images for virtual machine software

For the XtreemOS 2nd Public Release, disk images usable with common virtualisation software (e.g. Vmware, VirtualBox) will be made available for the sake of letting the user try a simple one-node XtreemOS installation without having to configure the system at all.

information needed

3.3 Main supported installation methods

3.3.1 Installation from CD

The most common method for installing XtreemOS GNU/Linux is using the first CD. Just burn the image you have downloaded from a Mandriva mirror. Starting an install from CD is as simple as putting the disc (the first disc) into the drive and rebooting.

3.3.2 Installation from hard disk

There are two major ways of doing a hard disk installation: you can either install from a local mirror of the XtreemOS GNU/Linux tree (which you have previously created by downloading the entire tree from a public mirror using, for e.g., rsync), or you can install directly from the .ISO format images of the XtreemOS GNU/Linux CDs without burning them to disc. To install from a local mirror, select the hard disk installation method. Then select the drive and partition where the local mirror is stored. Finally, enter the path to the correct directory of the XtreemOS GNU/Linux CD1. To install from .ISO images on a local hard disk, select the hard disk installation method. Then select the drive and partition where the ISO image is stored. Then enter the path to the ISO image.

3.3.3 Installation from a local network

Select the appropriate method for the type of server you wish to install from: NFS, HTTP, FTP. For all methods, you must now enter your network configuration information (for a typical home user, select DHCP and all default settings; other users should know their settings, or consult your network manager).

For the HTTP and FTP methods, you will now be asked to configure a proxy, if appropriate. If not, simply leave the boxes blank. For the HTTP and FTP methods, select the ”Specify the mirror manually” option. At the next step, enter the path to the mirror as appropriate. The path to enter is the path to the correct architecture (i586 only for the moment).

For the NFS method, after configuring networking, you will be asked to enter the hostname or IP address of the NFS server, and the path containing the XtreemOS GNU/Linux installation files.

3.3.4 From a Mandriva system

The packages of the XtreemOS 2nd Public Release are built on a Mandriva Linux 2009.0 distribution. It is therefore possible to install the packages on a Mandriva Linux 2009.0 distribution, see the next section about setting up packages repositories.

3.4 Setting up package repositories

Once your XtreemOS machine is installed, it is recommend to set up the online packages repository, to be able to install packages updates for bugfix.

This release of XtreemOS is based on Mandriva 2009.0, so you need to setup both Mandriva 2009.0 repositories, and XtreemOS repositories.

The following command will add both the Mandriva 2009.0 and XtreemOS repositories, selecting automatically the fastest mirror :

```
(root)# urpmi.addmedia --distrib --mirrorlist '$MIRRORLIST'
```

3.4.1 Using a proxy

If you are behind a proxy, you might need to setup wget or curl to use the proxy to let urpmi download the packages. This can be done by setting the following environment variables :

```
ftp_proxy=http://proxy_ip:port/
http_proxy=http://proxy_ip:port/
```

If using wget, this can be set in `/etc/wgetrc`, with the following variables :

```
ftp_proxy=http://proxy_ip:port/
http_proxy=http://proxy_ip:port/
```

Should not need to specify either option below, the default seems to work OK
Use the `--curl` or `--wget` option in your urpmi commands to select whether wget or curl should be used to download the packages.

3.4.2 The different types or repositories

The previous command will setup different kind of repositories :

- **release** The release repository contains the packages as they were when when the distribution was first released. This repository is not updated after the release.
- **updates** The updates repository contains the updated stable packages
- **testing** The testing repository contains testing package updates. Those packages should be used very carefuly. Normal users do not want this repository, for this reason it is not enabled by default.
- **backports** The backports repository is not available on the XtreemOS packages. It contains backports of new versions of some software. It should be used carefuly, for this reason it is not enabled by default.

Warning: Some testing and backport repositories are setup but not enabled by default. Make sure you don't enable them, or this could create some problems. This release of XtreemOS is based on Mandriva 2009.0, make sure you don't setup repositories from an other release such as 2009.1.

3.4.3 Updating packages

It is important to keep your system up-to-date. The following command will install the latest packages from the repositories you have configure:

```
(root)# urpmi --auto-update
```

3.4.4 Installing a package from the testing repository

If you want to install a package from the XtreemOS Testing repository, you can use the following command :

```
(root)# urpmi.update XtreemOS_Testing
(root)# urpmi --media XtreemOS_Testing package_name
```

3.4.5 Using URPMI

Documentation for urpmi is available in the following man pages :

- urpmi
- urpmi.addmedia
- urpmi.removemedia

- urpmi.update
- urpme
- urpmf
- urpmq
- urpmi.cfg
- urpmi.files

Chapter 4

XOS Grid Configuration Tool, Local GUI Installer and Tools

This chapter describes the available installation methods for an XtreemOS platform, from the installation of the single system to the whole platform configuration. Different tools are used for these tasks, either addressing the configuration of a single node (§4.4 and §4.5) or of a platform of interoperating nodes (§4.2).

Note on 64bit systems. Please note that as of XtreemOS 2nd Public Release, the Grid configuration tool is not installed by default on 64bit architectures; it can be however installed with urpmi from the network, or placing the package on a removable device.

INRIA Check that this is correct, provide instructions in section 4.2

4.1 System Installation

The installation of the XtreemOS 2nd Public Release is very similar to a classical Mandriva 2009.0 installation. You can refer to the Mandriva documentation for reference about the first steps of the installation.

The main difference is the installation type selection step, where you need to select the type of XtreemOS installation you want (i.e. core, resource, client nodes and so on, see §2.1, §**ADM**:6.5.1).

You can either

- install and configure each node separately, following the graphical installer steps all the way through the detailed XtreemOS configuration

parameters; see §4.4 and §4.5 for the GUI and command line installation options.

- install the nodes without configuring them, and use the Grid Configuration Tool described in §4.2 to manage the XtreemOS platform level services.

4.2 Grid Configuration

The `xosautoconfig` allows to configure and automatically install a whole XtreemOS grid. The basic idea is to first define the roles of some nodes as well as some basic configuration files common to all nodes in the grid in some folder, to replicate this folder and run the `xosautoconfig` tool on each node of the grid. The `xosautoconfig` tool extracts configuration informations from 4 configuration files and one configuration folder. By default, `xosautoconfig` searches for configuration files and folder in
`/etc/xos/xosautoconfig`
A different path can be provided through argument `--datapath`.

4.2.1 Configuration files

The roles of the nodes in a grid are defined using 4 configuration files: `nodeTypes`, `services`, `globalDefs` and `localDefs`. The first 3 files should be identical on all nodes of the same Grid.

nodeTypes: File `nodeTypes` (alternative path: use `--nodetypes`) allows to define node types and to specify which node of the grid are of which type. Each line of the `nodeTypes` file lists nodes belonging to a node type. Example:

```
head-node: xos-core.xtreemos.eu
resource-node: xos-node1.xtreemos.eu xos-node2.xtreemos.eu
```

This example defines two node types, `head-node` and `resource-node`. Type `default-type` is associated to all nodes by default.

services: File `services` (alternative path: use `--services`) specifies which nodes of the XtreemOS grid runs which services, depending on the node's type or the node's name. Here is the example of a small grid with two types of nodes: a `head-node` running all core services and a resource node type.

```
head-node: JobDirectory JobMng RCAServer ResAllocator ReservationManager
head-node: ExecMng RCAClient ResAllocator ResourceMonitor SRDSMng ResMng
head-node: VOLife xvoms cdaserver VOPS
head-node: xtreemfs-dir xtreemfs-mrc xtreemfs-osd
head-node: amsd nsspam openssh xtreemos-openssh ntp xtreemfs-client

resource-node: ExecMng RCAClient ResAllocator ResourceMonitor
resource-node: SRDSMng ResMng
resource-node: amsd nsspam openssh xtreemos-openssh ntp xtreemfs-client

all-nodes: CronDaemon DaemonGlobal XMLExtractor
```

Each line can contain node types, node names and service names. A service is configured on some node if the service name appears on the same line as the node type or the node name.

globalDefs: File `globalDefs` (alternative path: use `--globaldefs`) defines configuration values common to all nodes of the same grid. This file is "sourced" by the configuration script. The following variables are currrently handled by the tool:

variable	
`PROXY`	configure the proxy in `/etc/wgetrc` and/or in `/etc/profile.d/[proxy.csh|proxy.sh`. Default: `noProxy`
`NTP`	configures the NTP server
`GLOBALVOPSIP`	IP address of VOPS
`SCALARISBOOTIP`	IP address of scalaris bootstrap node
`OWBOOTSTRAPIP`	IP address of OW bootstrap node
`RSSBOOTSTRAPIP`	IP address of RSS bootstrap node
`DIXIROOTHOS`	hostname of DIXI root
`DIXIROOTIP`	IP address of DIXI root
`DIRHOSTIP`	IP address of XtreemFS dir
`MRCHOSTIP`	IP address of XtreemFS mrc
`OSDHOSTIP`	IP address of XtreemFS osd
`USESSL`	`true` or `false`

Undefined variables are not configured in the system.

localDefs: In order to configure a node, the `xosautoconfig` tool needs to define some data about the node. In the general case, the tool can guess these data automatially. It is possible to provides the following values through file `localDefs` (alternative path: use `--localaldefs`) or through command-line arguments.

variable	argument	
MYHOSTNAME	--myhostname	node host name
MYIP	--myip	node IP
MYINTERFACE	--myinterface	corresponding ethernet interface
MYDISK	--mydisk	monitored filesystem
MYNODETYPE	--mynodetype	node type
XOSDADDRESSEXTERNALADDRESS		xosd external address. Default: node IP
XOSDADDRESSHOST		xosd address. Default: node IP
ADDRESSHOST		???
SETMEDIA	--setmedia	true or false: initialise repository media when true. Reset to false after each configuration

All variables not defined explicitely in localDefs or in the command line are guessed by xosautoconfig. The tool first guesses the IP address and the interface of the node (IP address and interface corresponding the default route), then the hostname of the node from DNS, /etc/hosts, or conf/etc/hosts in this order. Once the hostname is configured, the node type and the list of services are extracted from the configuration files.

4.2.2 Configuration folder

The configuration folder is located by default in /etc/xos/xosautoconfig/conf. This path can be redefined using argument --datapath of the tool. This directory reproduces part of the system root tree /. The files located in this directory are first copied in their home location in the configured node and then updated using the information from the xosautoconfig configuration files. Files absent from the configuration folder are ignored. The most interesting files handled by the configuration tool are:

- /root/.ssh/authorized_keys2: the user SSH public key allows user to log in the node as root using ssh.
- /root/.xos/config-xos: defines the location of the user certificate.
- /root/.xos/truststore/certs/user.crt: user certificate.
- /root/.xos/truststore/private/user.key: user key.
- /etc/hosts: defines host names and their IP addresses for the Grid. This file is usefull when no common naming service can provide this mapping for the Grid.
- /etc/xos/truststore/certs/: files xtreemos.crt, cda.crt, vops.crt and rcaserver.crt are handled by the configuration toll. The destination directory is re-hashed (c_rehash) after any modification.

- `/etc/xos/truststore/private/`: files `rcaserver.key`, `cda.key` and `vops.key` are installed only if the node runs the corresponding service.

For a complete list of files and folders managed by `xosautoconfig`, check the `xosautoconfig` documentation.

4.2.3 Node Configuration

Once the root folder of `xosautoconfig` has been configured, it must be replicated on all nodes of the grid: `tar` the folder and then copy and `untar` it on the nodes. If necessary, update some values of the `localDefs` file. Then, run `xosautoconfig`. Before leaving, `xosautoconfig` requests a resource certificate for the node if none is already installed. In order to generate this certificate, it is necessary to run command `confirmResource` on the node running service `RCAServer` (this node should be configured first) and then execute command `finishConfig` on the node. These two steps are not necessary on the node running `RCAServer`. After these steps, the node should be configured, the requested services running and the boot scripts configured to restart the requested services after each boot.

4.2.4 Modifying Node Configuration

It is possible to reconfigure an already configured node: just modify the configuration files and re-run the configuration tool. The tool will stop no more necessary services and configure new ones. Re-running the configuration tool allows also to change the IP address of the node.

4.2.5 Cloning a Node

To clone a node, just copy the root tree on a new machine, boot the node and run the configuration tool. The configuration tool will automatically set up the new node.

4.2.6 Known Problems

Resource Certificates

At the end of the configuration step, `xosautoconfig` runs `rca_apply`. It may happen that this request fails. If something seems wrong about resource certificates, check the status using `rca_list_pending` and `rca_list_registered`. Rerun `rca_apply` if necessary.

VO Configuration

If a VO unique ID is provided to `xosautoconfig`, through variable `CONFIGUREVO` of file `localDefs` or through argument `--configurevo` of the tool, `xosautoconfig` tries to configure the local policies of the node and to install a VO certificate. These steps may fail. In this case, configure local policies (see section 7.4.4) and request VO certificate manually (see 6.7.7).

Service Dependencies

Using `xosautoconfig`, the XtreemOS nodes are configured independently. The tool does not take care of dependencies between services running on different nodes, for instance between a `xtreemfs-mrc` and a `xtreemfs-dir`, between a `RCAClient` and a `RCAServer`, or between a `DIXI` client and a `DIXI` bootstrap node. Problems may happen during the configuration step as well as after a reboot. The general rule is that the nodes providing core services should be configured/booted first.

Note

Configuration tool `xosautoconfig` is experimental in XtreemOS 2nd Public Release.

4.3 Package management

copy package update setup here

add short doc about urpmi

4.4 Graphical Configuration Tool

The XtreemOS graphical configuration tool, *drakxosconfig* is a tool that allows you to modify the main XtreemOS configuration parameters. It targets single computational nodes, thus it allows you to make custom configuration changes on a per-node basis, but it leaves the responsibility of the overall platform configuration (installing and configuring resources in a mutually compatible way) on the head of the network system administrator. You are expected to know all the relevant technical details.

How to for the GUI based installer;
what installation methods it supports, how to select them, which configurations can be installed (e.g. core, resource, client) and how to achieve them;
known issues, configurations that have to be performed manually (if any)
This Section to be prepared with the help of Nicolas Vigier and Antoine Ginies from Mandriva.
For now at least an explanation of the options of the command-line tool, and a summary of the steps to be followed with the GUI installer.

4.5 Command Line Configuration Tool

The *xosconfig* program is a command line interface to the same configuration library used by the graphical configuration tool. It will however allow you to edit more options than the graphical tool, which will only show the main options.

This command allow you edit the configuration for the different XtreemOS modules. To get a list of the different modules available on your system, run the *xosconfig* command without any parameter :

```
$ xosconfig
usage: xosconfig module [options]

available modules :
- ads
- xtreemfs-mrc
- xosd
- scalaris
- xtreemfs-dir
- srds
- xvoms
- xtreemfs-osd
- rss
- ow
```

Chapter 5

Setting up a single-PC installation of XtreemOS in 10 minutes

whole chapter commented out as it is no longer useful; move useful info to chapter 6, and possibly replace with instructions about using VM images

Chapter 6

Setting up your XtreemOS grid on a Testbed composed of multiple PCs

This chapter explains how to set up a small XtreemOS testbed. The idea is to set up an XtreemOS grid with every needed service in few nodes, starting with the XtreemOS install CD. That is not much of a grid, but it will show how to install and configure all the modules needed in a VO. Adding resources and users afterwards should be quite straightforward. The installation process is divided in three parts: the Core node one, the Resource node one and the Client node one. The process here is described in a very short form: for each node you will find a table summarizing the steps needed, that you can use as a checklist. Each entry in the table gives a few details, and references the section (usually in chapter 7) where full information and a general discussion of the topic is reported.

The installation procedure is described with simple workstations in mind (XtreemOS workstation flavour). Differences with respect to the SSI flavour of XtreemOS are documented in §7.9.

Important Security notice Here we will discuss the set up of the system, during which secure socket connection (SSL) among the nodes has to be disabled. You **will** need SSH, but SSL will remain disabled for a while. At the end of the set up, if your XtreemOS platform is not entirely contained within a secured network, you are strongly advised to also enable SSL. At this point, both SSH and SSL will be able to recognize other XtreemOS nodes and services by their certificates.

Please note that after certificate configuration is complete, to enable SSL you will need to restart **all** platform nodes. It is **not possible** to mix SSL enabled and not enabled nodes within the same platform.

6.1 Core node Checklist

Step	What to do	Notes	See
1st Install	Install from CD		
Repository	Set up repositories for Update		6.5.1
Root Certificate Authority	Configure Root CA		6.6.1
XVOMS	Configure XVOMS		6.6.2
XtreemFS	Configure XtreemFS subsystem		6.6.3
Accounts A	register Admin and user accounts on VOLife		6.6.2
Accounts M	Set-up / check account mapping service is running		
XtreemFS	test XtreemFS		6.6.4
DIXI	configure DIXI		6.6.5
SRDS	configure SRDS and its overlay networks	[a]	6.6.6
VOPS	configure VOPS		6.6.7
RCA	configure RCA		6.6.8
AEM	configure AEM		6.6.9
Enable SSL	Enable SSL in all modules	[b]	6.6.10

Notes:

[a] **Exactly one** of the core nodes shall be configured to act as a boot node of the SRDS overlays: OverlayWeaver, Scalaris, RSS.

[b] You need to execute this step unless you plan to experiment only on a single (virtual) machine, or within a secured local network

6.2 Resource node Checklist

Step	What to do	Notes	See
1st Install	Install from CD		
Repository	Set up repositories for Update		6.5.1
Certificates	Configure Root security certificates	[a]	6.7.1
VOLife	use VOLife to join a VO and get XtreemOS certificates		
Policy	set-up local policy management		6.7.2
SSH-XOS	configure SSH PAM modules to see XtreemOS certificates		??
DIXI	configure DIXI		6.7.5
SRDS	configure SRDS and its overlay networks	[b]	6.6.6
AEM	configure AEM		6.7.7
Resource registration	make resource available to AEM		6.7.7
XtreemFS-client	use XtreemFS volumes from resource nodes		6.7.8
Enable SSL	Enable SSL in all modules	[c]	6.7.9

Notes:

[a] will need to get them from . . . ?

[b] Resource nodes are non-BOOTSTRAP nodes

[c] You need to execute this step unless you plan to experiment only on a single (virtual) machine, or within a secured local network

6.3 Client node Checklist

Step	What to do	Notes	See
1st Install	Install from CD		
Repository	Set up repositories for Update		6.5.1
Certificates	Configure Root security certificates		6.8.1
VOLife	use VOLife to join a VO and get XtreemOS certificates		6.8.2
Policy	set-up local policy management		??
SSH-XOS	configure SSH PAM modules to see XtreemOS certificates		6.8.3
SRDS	configure SRDS and its overlay networks	[a]	6.6.6
DIXI	configure DIXI		6.8.5
XtreemFS-client	mount XtreemFS volumes		6.8.7
run job	run a job with AEM		6.8.8
Enable SSL	Enable SSL in all modules	[b]	6.8.9

Notes:

[a] Client nodes are non-BOOTSTRAP nodes

[b] You need to execute this step unless you plan to experiment only on a single (virtual) machine, or within a secured local network

shall SSH be configured or just used?

move examples to separate section

6.4 Active services on XOS nodes

This section reports the specific XtreemOS services that must be active on any kind of node of the platform.

Service Name	Function	Core	Resource	Client
xtreemfs-dir	XtreemFS	Y	N	N
xtreemfs-mrc	XtreemFS	Y	N	N
xtreemfs-osd	XtreemFS	Y	N	N
mysqld	DBMS backend for X-VOMS	Y	N	N
cdaserver	CDA server for X-VOMS	Y	N	N
tomcat5	VOLife web application for X-VOMS	Y	N	N
xos-amsd	account mapping service	Y	Y	Y
gmond	resource monitoring	Y	Y	N
xosd	XtreemOS daemon	Y	Y	N

Starting/stopping services All services are usually started/restarted in the classical way through the service script, e.g. for gmond

```
(root)# service gmond start
```

XOSd is a special case

```
(root)# service xosd stop
Ctrl+C it once the XOSd is finished.
```

6.4.1 Certificates needed for each type of node

Certificate	Description/**How to create**	Core	Resource	Client
xtreemos.crt	Root of trust **create-rootca** Must be installed on all nodes in this Grid	Y	Y	Y
cda.crt	Service certificate for CDA server **create-csr host org cda** Also needs to be installed on resource nodes to verify XOS-Certificate	Y	Y	Y
vops.crt	Signing certificate for VOPS server **create-csr host org vops**	Y	Y	N
{mrc,dir,osd}.crt	Optional certificates for XtreemFS services **create-csr host org svc**	Y	N	N
xtfs_mnt.crt	Optional client certificate for XtreemFS mount **create-csr host org xtfs_mnt**	N	Y	Y
{resource}.crt	Machine resource certificate Machine Attribute Certificates Obtain by using RCA client commands	Y	Y	N
res{void}ext.crt	Machine VO attribute certificate Obtain by using RCA client commands	N	Y	N
user.crt	User XOS-Certificate Obtained with **get-xos-cert** command	N	N	Y

In the description of using **create-csr** above, **host** is the DNS entry for the host that the service will run on. The parameter **org** can be used to describe your organisation - it can be a quoted string giving a brief description.
For the XtreemFS services, the **svc** parameter is either **mrc**, **dir** or **osd** for the MRC, DIR or OSD services respectively.

After creating a CSR file for a service, this should be sent to the operator of the Root CA for signing - this will result in a service certificate being sent back to you. The filename of this certificate will be of the form **host-type.crt**, where **type** represents the type of the service.

6.5 Common Installation Steps in detail

6.5.1 Setting up the packages repositories and updating packages

After installing from the XtreemOS CD, you should update the packages from the online repositories. This is an important step, to ensure you have the latest versions of the packages with the most up-to-date functionality and security fixes. See section 3.4 about how to setup package repositories and update your system.

Is the following partitioning correct?

Core Node packages

During the installation, select "Core Node". If you have not selected core node during the installation, the packages can be installed with the following command :

```
(root)# urpmi task-xtreemos-coreservices
```

Resource Node packages

During the installation, select "Resource Node". If you have not selected resource node during the installation, the packages can be installed with the following command :

```
(root)# urpmi task-xtreemos-resource
```

Client Node packages

During the installation, select "Client Node". If you have not selected client node during the installation, the packages can be installed with the following command :

```
(root)# urpmi task-xtreemos-client
```

Update the installed packages

After installing the desired configuration from one of the **task-xtreemos-*** packages described above, you should update the system as shown in section 3.4.3.

6.6 Setting up a Core Node

6.6.1 Configuring the Root Certificate Authority

A Virtual Organization needs a Root Certification Authority ('root CA') to provide a public key certificate to all nodes in the VO. This subsubsection describes how to generate the private key for the Root CA and its corresponding public key certificate.

The Root CA is the top level of the trust mechanism in XtreemOS. It is a critical part in the XtreemOS Public Key Infrastructure (PKI). To achieve and maintain the level of trust required by users of an XtreemOS Grid, the Root CA must be operated on only one machine. This host must be a physically-secure core node to avoid compromise of the Root CA private key, which would destroy any trust placed in the Root CA. Some organisations may even choose to run the Root CA on a machine which isn't connected to a network, to eliminate any risk of intrusion.

The Root CA provides a root entity credential (the root CA public key certificate) which is trusted by all participants in an XtreemOS Grid, and a mechanism to create service certificates that identify and authenticate XtreemOS core services.

To install and configure the Root CA, follow the steps in §7.1.1 and §7.1.2.

Once the Root CA has been configured, the Grid Admin can process Certificate Signing Requests for service certificates. This operation is described in §7.1.3.

When the Root CA certificate and the CDA certificate have been generated, these "public certificates" can be placed in a central certificate repository where the Xosautoconfig script can access them.

The XtreemOS root certificate needs to be installed on **all** machines in this XtreemOS Grid. The certificate can be placed in
/etc/xos/truststore/certs/xtreemos.crt on these machines.

6.6.2 Configuring X-VOMS

X-VOMS comprises the services for security and VO management. Both the CDA server and the VOLife web application use the X-VOMS database.

Initialise X-VOMS database

The X-VOMS database is fundamental to the operation of XtreemOS, so we start by configuring this.

```
(root)# /usr/share/xvoms/bin/xvoms_init.sh
```

The xvoms_init.sh command will prompt you for a password for the database 'root' user. It then sets up another database user, 'volifecycle' (password 'xosvo'). Finally, it populates the database with a VO 'exampleVO', administrative user 'admin' (password 'xtreemos-admin'), and the example VO user, 'xtreemos-vouser' (password 'xtreemos').

Install and configure CDA server

The Credential Distribution Authority is implemented in the **cdaserver** package. There is a single instance of a CDA server in an XtreemOS Grid. The CDA server must be on the same core node as the X-VOMS database.

```
(root)# urpmi cdaserver
```

The standalone CDA client program can be used to obtain user VO credentials from the CDA, and is provided by the **cdaclient** package.

The CDA server issues XOS certificates to users. The server needs an service certificate issued by the Root CA to authenticate itself to the corresponding CDA client. This service certificate can be obtained by the procedure described in section 7.1.3. This procedure also produces a private key, which should be placed into `/etc/xos/truststore/private/cda.key`. The service certificate contains the service's public key, and should be placed in `/etc/xos/truststore/certs/cda.crt`.

```
(root)# certdir=/etc/xos/truststore/certs # short-cut to reduce typing
(root)# cp host-cda.crt ${certdir}/cda.crt
(root)# chmod a+r ${certdir}/cda.crt # Anyone can read the public key
(root)# chown cdauser.cdauser ${certdir}/cda.crt
```

The CDA private key is placed by default in `/etc/xos/trustore/private/cda.key`.

```
(root)# keydir=/etc/xos/truststore/private # short-cut to reduce typing
(root)# cp host-cda.key ${keydir}/cda.key
(root)# chmod a+r ${keydir}/cda.key
#
# users 'cdauser' and 'tomcat' need to read the key
# File permissions are reduced, but key is pass-phrase protected
```

```
#
(root)# chown cdauser.cdauser ${keydir}/cda.key
# Key is owned by the 'service user' e.g. 'cdauser' for CDA
```

The following aspects of the CDA server are configurable by setting values in the file `/etc/xos/config/cdaserver/cdaserver.properties`:

- **cdaserver.keyFilename** — private key of CDA server - must be kept secure, readable only by owner.
- **cdaserver.keyPassphrase** — the private key is secured by a passphrase, the longer the better.
- **cdaserver.certFilename** — public key certificate of CDA server.
- **xtreemos.rootCertificate** — public key certificate of root CA.
- **cdaserver.sslAlgorithm** — cipher used by SSL.
- **cdaserver.sslHandshakeCipher** — the cipher used in initial SSL key exchange.
- **cdaserver.signatureAlgorithm** — algorithm used to sign the XOS-certificate returned to user.
- **cdaserver.validityDays** — number of days that certificate is valid for
- **cdaserver.validityHours** — number of hours that certificate is valid for
- **cdaserver.validityMinutes** — number of minutes that certificate is valid for

In general, you should only need to change the property `cdaserver.keyPassphrase` to the passphrase you set when running the `create-csr` command for the CDA.

The validity of a certificate is calculated as (cdaserver.validityDays) days + (cdaserver.validityHours) hours + (cdaserver.validityMinutes) minutes. Any two of these values can be zero. Hence, the lifetime of certificates issued by the CDA server can be set on a fine basis, if required.

Other aspects of the CDA server operation are:

Connection to X-VOMS database - this is set in `hibernate.cfg.xml`

The level of logging, log file location, etc, are defined in log4j.properties. The server writes its log files in `/var/log/cdaserver/cdaserver.log` by default.

Once configured, tyou can start the CDA server with the following command:

```
Root # /sbin/service cdaserver start
```

Test the CDA server and X-VOMS database by getting an XOS certificate

At this point, you are recommended to test that the X-VOMS database and the CDA server have been set up correctly by requesting an XOS certificate with the following command:

```
user$ get-xos-cert host:port exampleVO -g group1 -t
```

The **-t** flag runs the command in 'test' mode, where is supplies pre-configured parameters. You should replace **host** and **port** by the actual hostname of the node that is running the CDA server - **port** is normally **6730**, but can be set in **/etc/xos/config/cdaserver/cdaserver.properties**. The other parameters to the command, **exampleVO** and **group1**, are the names of a VO and a group pre-configured in the database. The pre-configured user account **xtreemos-vouser** is used by this 'test' invocation of the command.

If this command succeeds, it saves the private key in **/.xos/truststore/private/user.key**, protected by the passphrase **xtreemos**. It saves the XOS-certificate in **/.xos/truststore/certs/user.crt**.

You can check that the XOS-Certificate `user.crt` obtained from the CDA server can be verified against the certificate chain:

```
$ cd
$ openssl verify -CApath /etc/xos/truststore/certs \
.xos/truststore/certs/user.crt
.xos/truststore/certs/user.crt: OK
```

Running this step shows that the X-VOMS database is correctly configured and that the CDA server is operating correctly. You should not proceed further until you are able to execute this step correctly.

Configuring VOLife

VOLife is a web application which allows users to register for a Grid account, and for administrators to approve or reject these applications. Users can also create VOs, define their structure (groups and roles), and request to join VOs. There is a single instance of VOLife in an XtreemOS Grid, and it must be on the same core node as the X-VOMS database.

Ensure that VOLife is configured to find the CDA private key and certificate. When generating XOS-Certs, VOLife uses settings in `/etc/xos/config/volife/volife.properties` to locate the CDA private key and certificate, and to supply the pass-phrase protecting the private key. These settings can be copied from the CDA configuration in `/etc/xos/config/cdaserver/cdaserver.properties`.

```
$ cat /etc/xos/config/volife/volife.properties
cdaserver.keyFilename=/etc/xos/truststore/private/cda.key
cdaserver.keyPassphrase=changeme
cdaserver.certFilename=/etc/xos/truststore/certs/cda.crt
```

You will need to put the correct passphrase for the CDA private key in the property `cdaserver.keyPassphrase`.
The file needs to be readable by user 'tomcat' and no-one else.

```
root# chown tomcat.tomcat /etc/xos/config/volife/volife.properties
root# chmod go-r /etc/xos/config/volife/volife.properties
```

You need to start (or restart) Tomcat5 to run VOLife:

```
(root)# service tomcat5 restart
```

Configure the VOlife admin user

The VOLife webapp permits you to configure VOs and Users.

The first step is the registration of the Grid administrator. You should do this **immediately** after starting the VOLife service for the first time, as the 'admin' user has a pre-defined password which needs to be changed, for security reasons.

You can access VOLife with a browser at this address: `http://host:8080/volifecycle`
Where **host** is the name of the host where VOLife is running.

Sign in to the VOLife webapp:

```
Login id:  admin
Password:  xtreemos-admin
```

You **must** change the admin password straight away via the 'Change Password' tab in the VOLife left-hand navigation panel. This is to prevent unauthorised users logging into the VOLife webapp with the privileges of the Grid administrator.

It is now possible for users to register for a normal (non-admin) user account with VOLife. The VO administrator can approve user registration requests by logging-in to VOLife and selecting the 'Manage Users' tab.

Normal users can create their own VOs. You are the VO administrator of the VO you create:
— `Create a VO: VOName - VVVV`.

You can create your user private key:
— `Generate new key pair`:
Click on `Generate` (you need to specify a passphrase to protect the key) and `Download` your private key - it is stored in the file `user.key`.

Now you need a XOS-Cert to authenticate to the chosen VO. Click on Get a XOS-Cert and choose from the list of VO(s) you have created: — `Get a XOS-Cert` for the VO `VVVV`.

You need to enter your private key passphrase. Then you can download the public certificate by clicking on `Download`. You have now the XOS-Cert in the `user.crt` file. The certificate and key must be stored in the user's file system: see section 6.8.2.

6.6.3 Configuring XtreemFS

After having installed XtreemFS, the services can be started without making any changes to the standard configuration; default configuration files are located in `/etc/xos/xtreemfs`. XtreemFS services will then wait for client connections on their default ports and store their databases in the default directories. Note that the default configuration does not set up a secured SSL-based XtreemFS installation.

To set up XtreemFS, three different services have to be run: `xtreemfs-dir`, `xtreemfs-mrc`, `xtreemfs-osd`. It is important to start `xtreemfs-dir` first, so that the other services can register at the Directory Service. As soon as all services have been started, the system is ready for use.

```
(root)# service xtreemfs-dir restart
stoping XtreemFS Directory Service (DIR)...                    [  OK  ]
starting XtreemFS Directory Service (DIR)...                   [  OK  ]
(root)# service xtreemfs-mrc restart
stoping XtreemFS Metadata and Replica Catalog (MRC)...         [  OK  ]
starting XtreemFS Metadata and Replica Catalog (MRC)...        [  OK  ]
(root)# service xtreemfs-osd restart
stoping XtreemFS Object Storage Device (OSD)...                [  OK  ]
starting Object Storage Device (OSD)...                        [  OK  ]
```

6.6.4 Testing XtreemFS

XtreemFS is an object-based file system designed for federated IT infrastructures that are connected by wide-area networks. Package XtreemFS-client is necessary for testing XtreemFS on a core node.

To create your data volume:

```
$ xtfs_mkvol localhost/TestVolume
```

To see your volume :

```
$ xtfs_lsvol localhost
TestVolume  ->  00065BBD8E7C900B51481CF8
```

If necessary :

```
(root)# modprobe fuse
```

Create a folder and mount the volume:

```
$ mkdir $HOME/TestVolume
(root)# xtfs_mount -o direct_io,allow_other \
        localhost/TestVolume $HOME/TestVolume
```

allow_other permits to access the data without being root.

Create a file in your volume:

```
$ touch $HOME/TestVolume/test.txt


$ ls $HOME/TestVolume
test.txt
```

to umount this volume :

```
(root)# umount $HOME/TestVolume
$ ls $HOME/TestVolume
(nothing)
```

Mount it again :

```
(root)# xtfs_mount -o direct_io,allow_other \
        xtreemos1.zib.de/TestVolume $HOME/TestVolume
$ ls $HOME/TestVolume
test.txt

(root)# umount $HOME/TestVolume
$ xtfs_rmvol xtreemos1.zib.de/TestVolume
```

6.6.5 Configuring DIXI

DIXI needs a designated node to be root (or the Core) node. One might simply designate the first node installed in the grid to be the Core node, and then have all the others connect to it. When setting up a resource node, the Core node will be some other known node.

The main configuration file of the DIXI resides under /etc/xos/config and is named **XOSdConfig.conf**.

In the following steps, please replace the example IP 131.254.201.16 with IP of your core machine, to adapt your configuration.

It is possible to generate the XOSd configuration files by running the XOSd server.

```
(root)# service xosd start
(root)# service xosd stop
```

This way auto-generated /etc/xos/conf/XOSdConfig.conf will contain the defaults, including the network address entries, and it would differ from the one listed here. To change the values, we recommend using the **drakxosconfig** tool, the *xosd* tab. After changing the values, click the *Write the conf* to save them to the configuration file. Here you can also stop or start the xosd, and allow it to run at boot-up.

These are the values to be inspected:

- **networkInterface** (leave if as it is if not sure),
- **rootaddress.host** is IP address of the Core node,
- **rootaddress.externalAddress** if the Core node is behind NAT (or set it equal to **rootaddress.host**, if not sure leave it as it is),
- if this node (the Core node) is behind NAT, change **externalAddress** or leave it as it is if not behind NAT (if not sure, just leave it as it is).

```
# emacs /etc/xos/config/XOSdConfig.conf
xosdStagesSubDirectory=xosd_stages
useSSL=false
xosdport=60000
xmlport=55000
trustStore=/etc/xos/truststore/certs/
rootaddress.host=131.254.201.16
rootaddress.port=60000
```

```
rootaddress.externalAddress=131.254.201.16
externalAddress=131.254.201.16
networkInterface=eth0

xosdRootDir=.

trustStoreSSL=/etc/xos/truststore/certs/dixi_ssl/
privateKeyLocation=/etc/xos/truststore/private/reskey.key
certificateLocation=/etc/xos/truststore/certs/rescert.crt

Quit emacs.
```

The DIXI looks in /etc/xos/config/xosd_stages for the .stage files to see which services to run. The following files need to be present and have the line `enabled=true` for the DIXI to work properly:

```
XMLExtractor.stage
CronDaemon.stage
DaemonGlobal.stage
```

For the settings related to the **secure communication** using **SSL**, please refer to 6.6.10.

6.6.6 Configuring SRDS

Configuring SRDS and its dependencies (RSS and ADS_OverlayWeaver and ADS_Scalaris) means to edit some files with proper IP and port numbers. Full explanation is provided in 7.6.

Each node within **XtreemOS** has to run an instance of the **SRDS**, all those instances connecting into overlay networks. Currently, three overlays are created. All the SRDS overlays need one special node called "boot" "recorder" or "root", in order to startup the overlay.

We here assume that you choose **exactly one** Core node to do that, the BOOTSTRAP, and it will be specially configured. All other Core, Resource, Client nodes will have a general configuration.

The following sections give an overview on the main parameters that are configurable for each module inside the SRDS. The proper configuration file is indicated for each component. Please refer to the table in Section 7.6.1 for a complete list of configuration files for the SRDS.

The XOS configuration graphical tool (tab: SRDS and RSS) allows to automatically configure some of the parameters listed below (signed by a star).

Ads

Edit the /etc/xos/config/Ads/srds.properties with the proper values the following parameters

- **srds.configuration.device.disk*:** disk used to compute free space
- **rds.configuration.device.ethernet*:** main network interface

Rss

Open the /etc/xos/config/Rss/config.cfg configuration file and edit the following parameters

- **bootstrap_address*:** the SRDS BOOTSTRAP node for the RSS overlay
- **network_interface:** the network interface used by the RSS daemon; default eth0
- **disk_device:** the disk device monitored by the RSS; optional parameter

The disk_device parameter defines the device used by the RSS to estimate the amount of disk space at the node. If this parameter is not set, the RSS uses the device where the temporary file directory is mounted (typically /tmp).

Overlay Weaver

Configuration in `/etc/xos/config/OW/OWconfig.cfg` should be correct at installation, check the following values.

- **dht.BootstrapHost*:** specifies the address (numerical) of the boostrap host
- **dht.BootstrapPort:** specifies the port of bootstrap node (default 3997)

Scalaris

This configurations must be checked against the scalaris configuration info; check the meaning of localhost in the examples below

Every node in the scalaris overlay must have a FQHN (Fully Qualified Host Name).

BOOTSTRAP node
file /etc/scalaris/scalaris.properties

- scalaris.node : should take *boot@localhost*
- scalaris.cookie : cookies used by scalaris for connection (by default take value *chocolate chip cookie*)

file /etc/xos/config/Ads/srds.properties

- srds.configuration.bootstraping.isScalarisBootNode* = **true**

file /etc/scalaris/scalaris.cfg

- the boot_host, in the row like {`boot_host`,{{`146,48,82,99`},`14195`,`boot`}} should take the numerical public ip of this machine

all non-BOOTSTRAP nodes

file /etc/scalaris/scalaris.properties

- scalaris.node : should take *node@localhost*
- scalaris.cookie : cookies used by scalaris for connection (by default take value *chocolate chip cookie*)

file /etc/xos/config/Ads/srds.properties

- srds.configuration.bootstraping.isScalarisBootNode* = **false**

file /etc/scalaris/scalaris.local.cfg

- the boot_host, in the row like {`boot_host`,{{`146,48,82,99`},`14195`,`boot`}} should take the numerical public ip of the boot machine..

Running and Monitoring SRDS

The DIXI's Xosd will start up SRDS if the `/etc/xos/config/xosd_stages` directory contains a file named `SRDSMng.stage`. See 7.6.1 for its contents.

You can monitor the SRDS behaviour by connecting to the node at port at the srds.configurations.webserverPort as specified in `/etc/xos/config/Ads/srds.properties` (it is port 9000 by default). The SRDS web server relays here in a single web page all configuration information and web interfaces of its overlays, as well as the log messages from the XOSd.

6.6.7 Configuring VOPS

VOPS is a core service, which needs to run on no more than one node in the domain. If your domain is already running VOPS elsewhere, the service should not be set up on the new node, and it should be disabled.

VOPS requires the service certificate and a private key for signing its decisions. Please refer to 7.1.3 for instructions on how to create a service certification request for the `vops` application. This will also create the private key. Once the administrator processes the certification request and sends the service's public certificate, please install the private key and certificate:

```
(root)# cp MyHostName-vops.crt /etc/xos/truststore/certs/vops.crt
(root)# mv MyHostName-vops.key /etc/xos/truststore/private/vops.key
```

The VOPS runs in DIXI, which looks in /etc/xos/config/xosd_stages for the .stage files to see which services to run. The following file needs to be present and have the line `enabled=true` to have VOPS run:

```
VOPS.stage
```

VOPS reads /etc/xos/config/VOPS.conf to get the configuration. If the file does not exist yet, you can have VOPS generate it with defaults by starting and then stopping the DIXI:

```
(root)# service xosd start
(root)# service xosd stop
```

Please check the following values:

- **keyPassword** — the password for accessing the service's private key.

6.6.8 Configuring RCA

RCA Server is a core service, which needs to run on no more than one nodes in the domain. If your domain is already running RCA Server elsewhere, the service should not be set up on the new node, and it should be disabled.

The RCA Server service belongs to the core services. RCA Server requires the service certificate and a private key for issuing resource certificates. Please refer to 7.1.3 for instructions on how to create a service certification request for the `rca` application. This will also create the private key. Once the administrator processes the certification request and sends the service's public certificate, please install the private key and certificate:

```
(root)# cp MyHostName-rca.crt /etc/xos/truststore/certs/rcaserver.crt
(root)# mv MyHostName-rca.key /etc/xos/truststore/private/rcaserver.key
```

The RCA Server runs in DIXI, which looks in /etc/xos/config/xosd_stages for the .stage files to see which services to run. The following file needs to be present and have the line `enabled=true` to have RCA Server run:

`RCAServer.stage`

RCA reads /etc/xos/config/RCAServer.conf to get the configuration. If the file does not exist yet, you can have RCA generate it with defaults by starting and then stopping the DIXI:

```
(root)# service xosd start
(root)# service xosd stop
```

Please check the following values:

- **keyPassword** — the password for accessing the service's private key.
- **certDNCountry** — the country of your organisation. The value will be used as a part of the distinguished name in the issued certificates.
- **certDNLocation** — the location of your organisation. The value will be used as a part of the distinguished name in the issued certificates.
- **certDNOrganisation** — the name of your organisation. The value will be used as a part of the distinguished name in the issued certificates.
- **certDNOrganisationUnit** — the organisation unit. If you have multiple RCA servers within the organisation or domain, you can use this field to distinguish them. The value will be used as a part of the distinguished name in the issued certificates.

At this point you can also update the list of the VOs that will be supported by the RCA. This list will contain those VOs that the resources in this RCA will be able to provide the computation to.

To include the the VO with the ID `VVVV` to the RCA's list, use the following command:

```
(root)# rca_vo a VVVV
```

For example, if we have a VO with the ID `7485601c-43b0-413f-83a5-a968b1835aea`, we can add it to the list:

```
(root)# rca_vo a 7485601c-43b0-413f-83a5-a968b1835aea
```

To then check the RCA's list, use the `l` switch of the command:

```
(root)# rca_vo l
  7485601c-43b0-413f-83a5-a968b1835aea
```

The Core machine certificate

The core node will likely need to communicate with other nodes using SSL. It is most convenient to therefore use the machine certificate and private key for the communication.

Before running the commands for obtaining the certificates, the xosd service must be configured and run with the RCA Client service enabled in the /etc/xos/config/xosd_stages folder:

```
RCAClient.stage
```

The administrator then needs to execute the following sequence of commands:

```
Core node $ service xosd start
Core node $ service gmond restart
```

This ensures that the xosd is running and that the Ganglia monitoring is reset. The monitoring part is optional, but it helps obtaining the information on the node. Then we apply for registration.

```
Core node $ rca_apply
The RCA client needs to initialise the resource's descriptor.
Setting the RCA client's
The local AEM's ResourceMonitor reports:
  [hostIP={Address = [://172.16.117.13:60000(172.16.117.13)]},
  hostUniqueID={worker.xlab.si}, operatingSystemName={Linux},
  processorArchitecture={x86}, CPUCount={1.0}, RAMSize={3.06184192E8},
  cpuLoadLast15Min=0, cpuLoadLast5Min=3, cpuLoadLast1Min=4]

Registered services:
  [eu.xtreemos.system.communication.proxy.ServiceCallProxy,
  eu.xtreemos.system.communication.redirector.ServiceCallRedirector,
  eu.xtreemos.xosd.daemon.Daemon, eu.xtreemos.xosd.resallocator.ResAllocator,
  eu.xtreemos.ads.connection.dixi.SRDSMng, eu.xtreemos.xosd.daemon.Daemon,
  eu.xtreemos.xosd.xmlextractor.XMLExtractor,
  eu.xtreemos.xosd.security.rca.client.RCAClient,
  eu.xtreemos.xosd.resourcemonitor.ResourceMonitor,
  eu.xtreemos.xosd.crondaemon.CronDaemon]

Please select:
 1 Use ResourceMonitor's info
 2 Use info from /etc/xos/config/ResourceDescriptor.xml
```

```
3 Initialise /etc/xos/config/ResourceDescriptor.xml with
ResourceMonitor info
0 Quit
```

The menu, as shown in the example, may appear, prompting the administrator to select the manner in which the information on the node will be obtained. If the information displayed is satisfactory, select `1` and press Enter.

Resume by confirming the node and requesting the certificate.

```
Core node $ rca_confirm
Core node $ rca_request
```

6.6.9 Configuring AEM

Configuration files of the AEM reside under /etc/xos/config.

The DIXI looks in /etc/xos/config/xosd_stages for the .stage files to see which services to run. The following files need to be present and have the line `enabled=true` in order to provide AEM core functionality:

```
JobMng.stage
ResMng.stage
JobDirectory.stage
ReservationManager.stage
ResourceMonitor.state
```

Please note that, on the domain, these services do not have to all run on the same node. Instead, multiple core AEM nodes can exist, each running a subset of the core services.

If you are not sure about the IP to be used with command **rca_confirm**, **rca_list_pending** should list the IP to be used here.

6.6.10 Enabling SSL Secure Connection

This section shall explain what is needed in order to enable SSL connections within an XtreemOS platform. This is a critical issue.

Enabling SSL in DIXI

The communication between services on different nodes uses SSL by default. This can be turned off using the **useSSL** setting in the **XOSdConfig.conf**. However, for security reasons it is imperative that the secure communication is enabled. For using SSL, we need to configure which private key and which certificate to use in the handshakes, as well as which certificates to trust. These settings can be set via **privateKeyLocation**, **certificateLocation** and **trustStoreSSL** settings in **XOSdConfig.conf**, respectively. The certificates valid for the handshakes and accepted by other xosd nodes by default are the ones issued by the **CDA** (user's credentials) and the ones issued by the **RCA** (machine credentials).

The settings in the **XOSdConfig.conf** should be followed accordingly in the XATI and XATICA configuration files, found in the `.xos` subfolder of the user's home directory (`/root/.xos` or `/home/$USER/.xos`). Please refer to 7.2.2 for more information.

DIXI can only use one type of communication at a time. This means that **the nodes with SSL enabled will refuse the inbound plain-text connections**. In most cases this means that DIXI will need to have the SSL enabled in order to obtain the certificates that, in turn, will then be used for subsequent SSL communications. We therefore have two procedures, depending on whether we are starting with the first XtreemOS node, or we are adding a new node to an existing XtreemOS grid.

Installing the first XtreemOS node

When setting up XtreemOS for the first time, we recommend the following steps, performed by the resource administrator on the node using the root account:

- Designate a node that will be the first core node.
- Install and configure both the core-level and the node-level RCA service (6.6.8). Please make sure the node has the proper RCA's certificate and key files in place (7.3.5).
- Optionally, for security, set the node's firewalls to block any incoming traffic.
- Disable SSL by setting **useSSL** to **false** in **XOSdConfig.conf**.
- Disable SSL for the clients by setting **useSSL** to **false** in **/root/.xos/XATIConfig.conf**.
- Start the **xosd** service by calling `service xosd start`.

- Obtain the machine identity certificate by calling, in sequence: `rca_apply`, `rca_confirm`, `rca_request`
- Stop the **xosd** service by calling `service xosd stop`.
- Restore any firewalls to permit normal incoming traffic.
- Re-enable SSL by setting **useSSL** back to **true** in **XOSdConfig.conf**. Set the password to the **privateKeyPassword** value. Other settings should already point to the machine certificate and key by default.
- Re-enable SSL for the command-line administration commands by setting **useSSL** to **true** in **/root/.xos/XATIConfig.conf**. Make sure that **sslPrivateKeyPassword** contains the correct password.
- Obtain the resource administrator's user certificate and install it to the default folders.
- Next time the **xosd** service will start, it will operate using SSL only.

Adding a new node to a pre-existing XtreemOS site

When adding a new node to the network running XtreemOS with SSL, it is possible to use of the resource administrator's user certificate for boot-strapping the SSL for the new node. We recommend the following steps to be performed by the new resource's administrator using the root account:

- Obtain (using **get-xos-cert**) or install the administrator's user certificate and key.
- Open **/root/.xos/XATIConfig.conf** and make sure that the **sslPrivateKeyPassword** value contains the proper value.
- Open **XOSdConfig.conf** for text editing.
- Insert the hash (#) character to the beginning of the lines containing **privateKeyLocation**, **certificateLocation** and **privateKeyPassword** to comment them out.
- Add a new entry for **certificateLocation**, using the administrator's user certificate as the value (e.g., `/root/.xos/truststore/cert/user.crt`).
- Add a new entry for **privateKeyLocation**, using the administrator's private key as the value (e.g., `/root/.xos/truststore/private/user.key`).
- Add a new entry for **privateKeyPassword**, using the password to the administrator's private key as the value.

- Save the changes to **XOSdConfig.conf**.
- Install and configure the node-level RCA service (7.3.4). Please make sure that the RCA server's certificate is installed.
- Start the **xosd** service by calling `service xosd start`.
- Apply for the resource registration using `rca_apply`.
- The site administrator then needs to confirm the resource entry using `rca_confirm`.
- Obtain the node's machine identity certificate by calling `rca_request`.
- Stop the **xosd** service by calling `service xosd stop`.
- Open **XOSdConfig.conf** for text editing.
- Comment out using the hash (#) character the **privateKeyLocation**, **certificateLocation** and **privateKeyPassword** that are set to the administrator's user certificate and private key.
- Remove the hash (#) character from beginning of the lines with the original values for **privateKeyLocation**, **certificateLocation** and **privateKeyPassword**.
- Save the changes to **XOSdConfig.conf**.
- Use the xosd normally.

Enabling SSL in XtreemFS

A detailed description of how to set up an SSL-secured XtreemFS installation is given in Sec. 7.7.2.

6.7 Setting up a Resource node

6.7.1 Configuring the certificates

You need the Root CA certificate on your machine:

```
(root)# cp xtreemos.crt /etc/xos/truststore/certs/
```

You need to use the Core machine public certificate to authenticate SSL access to the services on the core node(s). You can find all the certificates (`host-cda/rca/vops.crt`) in the `/etc/xos/truststore/certs/` folder of the core node(s). Copy them to this resource node.

You have to put the CDA Server, RCA Server and VOPS public certificates in the `/etc/xos/trustore/certs/` on this machine.

```
(root)# cp *.crt /etc/xos/truststore/certs/
```

Now you need to link the certificates with their hash. The OpenSSL command `c_rehash` will create hash files for all the certificates in a directory:

```
c_rehash /etc/xos/truststore/certs
```

6.7.2 Configuring the local policy

Before using the PAM module, the Account Mapping Server (AMS) must be running:

```
(root)# service xos-amsd restart
```

Now test your public certificate with the local policy tool, *xos-policy-admin-chk*:

```
(root)# xos-policy-admin-chk -pem $HOME/.xos/truststore/certs/user.crt
dn = [UUUU], vo = [VVVV], role = [null]
Sucess in PAM checking !
```

If you get the sucess message, the configuration of local policy is ready. The tool also presents the number of user's DN,VO,ROLE. But generally, there are not any policy are defined in the beginning, so you may get the following message usually:

```
(root)# xos-policy-admin-chk -pem $HOME/.xos/truststore/certs/user.crt
dn = [UUUU], vo = [VVVV], role = [null]
Mapper: Unfound the match rule!! check your mapping rule,pls
PAM:fail in mapping connect !
        * a)Please check whether AMS daemon is running correctly *
        * b)Please check whether mapping rules are correct.      *
        *     If not, try:                                      *
        *       xos-policy-admin-am  -vo <vo> --force           *
        *       xos-policy-admin-gm  -vo <vo> --force           *
        * c)Please check whether setting rule is correct.       *
        *     If not, try:                                      *
        *       xos-policy-admin-set -uidmax <num> -uidmin <num> *
        *                            -gidmax <num> -gidmin <num> *
Oops: Permission denied
```

The *Mapper* in AMS outputs the message on missing mapping rules, so following the hint b) to add the mapping rules as local policy:

```
(root)# xos-policy-admin-am  -vo VVVV  --force
(root)# xos-policy-admin-gm  -vo VVVV  --force
```

Then, try again the above checking. If you got the following message from *Mapper*:

```
(root)# xos-policy-admin-chk -pem $HOME/.xos/truststore/certs/user.crt
dn = [UUUU], vo = [VVVV], role = [null]
Mapper:Undefine the uid/gid spaces for mapping!! check your
setting rule,pls
PAM:fail in mapping connect !
...
```

You have to add another policy rule to told the AMS how large the space (span) local uid/gid is mapped. By the tool, *xos-policy-admin-set*, it is eay to do that following the hint c):

```
(root)# xos-policy-admin-set -uidmax 60500 -uidmin 60000 \
-gidmax 60500 -gidmin 60000
```

And then, try again the checking tool and the sucess message will be presented in screen. You can check the added rules by *xos-policy-admin-prt*.

6.7.3 Configuring the pam_xos

You need to configure PAM to use the Root CA certificate:

```
(root)# emacs /etc/xos/nss_pam/pam_xos.conf
VOCACertDir             /etc/xos/truststore/certs
VOCACertFile            xtreemos.crt
Quit emacs.
```

If the resource node allows access users read/write global home volume, the automount option has to be opened after installation. Local administrator may configure the functionality by editing PAM's configuration file.

```
(root)# vim /etc/xos/nss_pam/pam_xos.conf
...
OpenAutoMount yes
...
```

Opening the file and setting the option to "yes" will help user's home volume mount to their local home directories (in /home, named with their DN number).

Retry it with *pam_app_conv* which can open a session,mounting with global XtreemFS home volume.

```
(root)# pam_app_conv -pem $HOME/.xos/truststore/certs/user.crt
```

If all is allright, you enter a new shell :

```
[11:44:28] core_machine /home/c4b32574-cf06-47e7-b960-97e7f6b994a4
/CN=c4b32574-cf06-47e7-b960-97e7f6b994a4 $
...
Ctrl-D.
```

6.7.4 Configuring SSH-XOS server

```
resource root $ emacs /etc/ssh/sshd_config-xos
UsePAM yes
Quit emacs.
```

6.7.5 Configuring DIXI

Here SSL is disabled

The procedure of configuring DIXI for the resource node is identical to the one for setting up a core node. The only difference is that the resource node's xosd will always need a Core or a root xosd to connect to. Please refer to section 6.6.5 for further instructions.

This is an example is valid for a resource node at IP 131.254.201.21 in a grid where the Core DIXI node runs on the node with the IP 131.254.201.16.

```
# emacs /etc/xos/config/XOSdConfig.conf
xosdStagesSubDirectory=xosd_stages
useSSL=false
xosdport=60000
xmlport=55000
trustStore=/etc/xos/truststore/certs/
networkInterface=eth0

rootaddress.host=131.254.201.16
rootaddress.port=60000
rootaddress.externalAddress=131.254.201.16
externalAddress=131.254.201.21

xosdRootDir=.

trustStoreSSL=/etc/xos/truststore/certs/dixi_ssl/
privateKeyLocation=/etc/xos/truststore/private/reskey.key
certificateLocation=/etc/xos/truststore/certs/rescert.crt

Quit emacs.
```

Please ensure that *none of the core services* are enabled on the resource node. This means that the following files in /etc/xos/config/xosd_stages **should be removed** if they exist there:

```
JobMng.stage
ResMng.stage
JobDirectory.stage
ReservationManager.stage
VOPS.stage
RCAServer.stage
```

For the settings related to the **secure communication** using **SSL**, please refer to 6.6.10.

6.7.6 Configuring SRDS, RSS

Resource and Client nodes exploit the same configuration as non-BOOTSTRAP Core nodes, see 6.6.6

6.7.7 Configuring the AEM

In the resource node scenarios, all the node-level services (`ExecMng`, `ResourceMonitor` and `ResAllocator`) need to be running on the node that would offer its resources to the jobs. This means that every resource node has to have its own set of the services.

Most of the AEM's services run in DIXI, which looks in /etc/xos/config/xosd_stages for the .stage files to see which services to run. The following files need to be present and have the line `enabled=true` for the DIXI to work properly:

```
ExecMng.stage
ResourceMonitor.stage
ResAllocator.stage
```

Make the resource available to AEM

Run the resource monitoring, or restart it if it is already running (sometimes this step is necessary after the boot-up or any network adapter changes on the system):

```
(root)# service gmond restart
```

(for killing the XOSd if necessary)

```
(root)# service xosd stop
```

If you are on the Core node then just run xosd. If not the Root XOSd must be running on 131.254.201.16 Core node. Link your Resource XOSd to Core XOSd:

```
(root)# service xosd start
```

1- The Core node needs to collect the details on the Resource node to be registered and to send the details and the node's ID to the RCA server for enlisting the resource to the resources pending for registration:

```
Resource $ rca_apply

Resource $ rca_list_pending
Returned from service call: successMethod
Listing pending resources:
ResouceID = [IP=131.254.201.21:60000]: [hostIP={Address =
[://131.254.201.21/131.254.201.21:60000(/131.254.201.21)]},
hostUniqueID={131.254.201.21}, operatingSystemName={Linux},
processorArchitecture={x86}, CPUCount={1.0}, RAMSize={2.125463552E9}]

Resource $ rca_list_registered
Returned from service call: successMethod
Listing pending resources:
  List empty.
```

2- When the resource does rca_confirm a higher instance on the Core node confirms the registration of the resource. This is probably done by an administrator at the organisation. So, on the Core node:

```
Core node $ rca_confirm 131.254.201.21:60000

Resource $ rca_list_pending
Returned from service call: successMethod
Listing pending resources:
  List empty.

Resource $ rca_list_registered
Returned from service call: successMethod
Listing registered resources:
ResouceID = [IP=131.254.201.21:60000]: [hostIP={Address =
[://131.254.201.21/131.254.201.21:60000(/131.254.201.21)]},
hostUniqueID={131.254.201.21}, operatingSystemName={Linux},
processorArchitecture={x86}, CPUCount={1.0}, RAMSize={2.125463552E9}]
```

It creates or update the RCA database in the file pointed by rcaDBFile in the RCAServerConfig.conf config file of the Core node.

```
Core node $ ls /etc/xos/RCADB.bin
/etc/xos/RCADB.bin
```

3- The Resource node can now create a key pair and request the signature of the RCA server with:

```
Resource $ rca_request

Requesting a new certificate...
Identity certificate:
 ...
```

4- The Core node generates a attributes certificate for the Resource node. You need the VO ID and the Resource node IP:

Before all, you have to open the incoming directory on your host:

```
Resource node $ chmod 777 /etc/xos/truststore/certs/incoming/

Core node $ rca_resource_vo a 7485601c-43b0-413f-83a5-a968b1835aea \
                        131.254.201.21:60000
```

So the Resource is registered to the VO and you get a attribute certificate in:

```
Resource node $ ls /etc/xos/truststore/certs/incoming/
attrcert7485601c-43b0-413f-83a5-a968b1835aeaext.crt

Resource node:
(root)# mv /etc/xos/truststore/certs/incoming/attrcert7485601c...1835aeaext.crt \
  /etc/xos/truststore/certs/
```

This node is now available as a VO 7485601c-43b0-413f-83a5-a968b1835aea resource. We can display the registered resources with the command which lists the available reservation slots on the discovered resources:

```
# xreservation -qf
Address = [://131.254.201.21:60000(131.254.201.21)]: * : *
```

We can also use xconsole_dixi with the blank.jsdl file.

```
$ emacs $HOME/blank.jsdl

<?xml version="1.0" encoding="UTF-8"?>
<JobDefinition xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl">
   <JobDescription>
       <JobIdentification>
            <Description>Blank</Description>
            <JobProject>Blank</JobProject>
         </JobIdentification>
```

```
        <Application>
            <POSIXApplication
                 xmlns:ns1="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix">
                <Executable></Executable>
            </POSIXApplication>
        </Application>
        <Resources>
            <TotalResourceCount> <Exact>1</Exact></TotalResourceCount>
        </Resources>
    </JobDescription>
</JobDefinition>

Quit emacs.

$ xconsole_dixi
XtreemOS Console
$xrs -jsdl $HOME/blank.jsdl
Listing resources matching JSDL query:
  Address = [://131.254.201.21/131.254.201.21:60000(/131.254.201.21)]

Pay attention to not put a space between the prompt and the xrs in the xconsole.
```

It is OK to run a job on this resource.

6.7.8 Configuring XtreemFS client

To allows automounting XtreemFS home volume for grid application on a resource node, provide the following configuration files:

File `/etc/xos/xtreemfs/default_dir` allows the XtreemOS home-volume automounter to locate the XtreemFS directory service.

```
[root@paraxos1 xtreemfs]# cat /etc/xos/xtreemfs/default_dir
dir_service.host = 131.254.201.16
dir_service.port = 32638

ssl.enabled = false

# server credentials for SSL handshakes
ssl.service_creds = /etc/xos/xtreemfs/truststore/certs/osd.p12
ssl.service_creds.pw = passphrase
```

```
ssl.service_creds.container = pkcs12

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

File `/etc/xos/xtreemfs/default_mrc` allows the XtreemOSautomounter to locate the XtreemFS MRC service for creating the home volume the first time a new user logs in the grid.

```
[root@paraxos1 xtreemfs]# cat /etc/xos/xtreemfs/default_mrc
mrc.host = 131.254.201.16
mrc.port = 32636

ssl.enabled = false

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

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

6.7.9 Enabling SSL Secure Connection

This section shall explain what is needed in order to enable SSL connections within an XtreemOS platform. This is a critical issue.

Enabling SSL in DIXI

The principles of enabling the SSL in DIXI on the node level are similar to the ones in the core level, as explained in 6.6.10. Here we assume there is already an existing XtreemOS network running on the site, and we need to connect to it using SSL to perform the initial steps.

When adding a new node to the network running XtreemOS with SSL, it is possible to use of the resource administrator's user certificate for boot-strapping

the SSL for the new node. We recommend the following steps to be performed by the new resource's administrator using the root account:

- Obtain (using **get-xos-cert**) or install the administrator's user certificate and key.
- Open **/root/.xos/XATIConfig.conf** and make sure that the **sslPrivateKeyPassword** value contains the proper value.
- Open **XOSdConfig.conf** for text editing.
- Insert the hash (#) character to the beginning of the lines containing **privateKeyLocation**, **certificateLocation** and **privateKeyPassword** to comment them out.
- Add a new entry for **certificateLocation**, using the administrator's user certificate as the value (e.g., `/root/.xos/truststore/cert/user.crt`).
- Add a new entry for **privateKeyLocation**, using the administrator's private key as the value (e.g., `/root/.xos/truststore/private/user.key`).
- Add a new entry for **privateKeyPassword**, using the password to the administrator's private key as the value.
- Save the changes to **XOSdConfig.conf**.
- Install and configure the node-level RCA service (7.3.4). Please make sure that the RCA server's certificate is installed.
- Start the **xosd** service by calling `service xosd start`.
- Apply for the resource registration using `rca_apply`.
- The site administrator then needs to confirm the resource entry using `rca_confirm`.
- Obtain the node's machine identity certificate by calling `rca_request`.
- Stop the **xosd** service by calling `service xosd stop`.
- Open **XOSdConfig.conf** for text editing.
- Comment out using the hash (#) character the **privateKeyLocation**, **certificateLocation** and **privateKeyPassword** that are set to the administrator's user certificate and private key.
- Remove the hash (#) character from beginning of the lines with the original values for **privateKeyLocation**, **certificateLocation** and **privateKeyPassword**.

- Save the changes to **XOSdConfig.conf**.
- Use the xosd normally.

6.8 Setting up a Client node

6.8.1 Configuration certificates

You need the Root CA certificate on your machine:

```
(root)# cp xtreemos.crt /etc/xos/truststore/certs/
```

You need to use the Core machine public certificate to access the services on these machine. You can find all the certificates {cda,rca,vops}.crt in the directory /etc/xos/truststore/certs/ of the core server machines.

You have to put the public certificates in the directory /etc/xos/trustore/certs/ on this machine:

```
(root)# cp corehost-{cda/rca/vops}.crt /etc/xos/truststore/certs/
```

Now you need to link the certificates with their hash:

```
c_rehash /etc/xos/truststore/certs
```

You need to configure PAM to use the Root CA certificate:

```
(root)# emacs /etc/xos/nss_pam/pam_xos.conf
VOCACertDir             /etc/xos/truststore/certs
VOCACertFile            xtreemos.crt
Quit emacs.
```

6.8.2 Get User XOS-Certificates with VOLife

The VOLife permits you to register and to join a VO. You can access it with a browser at this address: http://server_machine_ip:8080/volifecycle

— Create a Client user:
Client
password
An Xtreemfs Volume is created for the user.

You are not the VO administrator of the VO you want to Join. Click on Join a VO then choose the right VO and click on the JoinVO button for the VO VOName - VVVV

You have to wait that the VOAdmin adds you at the VO.

If it is the first time you have to create your user private key:
— Generate new key pair:

Click on Generate (you need to create a new password) and Download your private key (user.key).

Now you need a XOS-Cert to authenticate to the chosen VO. Click on Get a XOS-Cert and choose the VO you have created:
— Get a XOS-Cert for the VO called VOName with the VO ID VVVV.

You need to enter your private key password. Don't forget to download the public certificate by clicking on Download. You have now the user.crt file.

Create the user .xos folder and the certificate subfolder and copy the user.crt and user.key certificates in the $HOME/.xos/truststore/{private,certs} folder:

```
$ mkdir -p $HOME/.xos/truststore/private/
$ mkdir -p $HOME/.xos/truststore/certs/
$ cp user.key $HOME/.xos/truststore/private/
$ cp user.crt $HOME/.xos/truststore/certs/
```

You can check that the XOS-Certificate can be verified against the certificate chain:

```
$ cd
$ openssl verify -CApath /etc/xos/truststore/certs/ \
.xos/truststore/certs/user.crt
.xos/truststore/certs/user.crt: OK
```

6.8.3 Configuring SSH-XOS client

Try to connect to the VO Core machine.

```
Client root $ emacs /etc/ssh/ssh_config-xos
XosProxyFile    $YOUR_HOME/.xos/truststore/certs/user.crt
Quit emacs.

Core root $ emacs /etc/ssh/sshd_config-xos
UsePAM yes
Quit emacs.
```

(NB You should substitute $YOUR_HOME for the pathname of your home directory.)

Connect to the destination machine:

```
Client $ ssh-xos core_machine
-bash-3.2$ whoami
/CN=c4b32574-cf06-47e7-b960-97e7f6b994a4
Ctrl-D.
```

6.8.4 Configuring SRDS, RSS

Resource and Client nodes exploit the same configuration as non-BOOTSTRAP Core nodes, see 6.6.6

6.8.5 Configuring the DIXI

here SSL is disabled

You have to configure some files in /etc/xos/config/ and .xos/ folders.

In my system the Client IP is 131.254.201.20, the Resource IP is 131.254.201.21 and the Core and root xosd IP is 131.254.201.16. You need to adapt your configuration.

You need to generate your client configuration files. You could run xconsole_dixi or xsub to generate them:

```
$ xconsole_dixi
Ctrl-C
```

Now modify them:

- **networkInterface** (leave if as it is if not sure),
- **userKeyFile** is the path to the user's private key,
- **userCertificateFile** is the path to the user's public certificate,
- **xosdaddress.host** IP address of the XOSd running on the Resource node, i.e., the address of the same node,
- **xosdaddress.externalAddress** if the Resource node is behind NAT (or set it equal to **xosdaddress.host**, if not sure leave it as it is),
- if this node (the Resource node) is behind NAT, change **externalAddress** or leave it as it is if not behind NAT (if not sure, just leave it as it is).

If running **xconsole_dixi**:

```
$ emacs $HOME/.xos/XATIConfig.conf

useSSL=false
certificateLocation=/etc/xos/truststore/certs/xati_dummy.pem
privateKeyLocation=/etc/xos/truststore/private/xati_dummy.pem
trustStoreSSL=/etc/xos/truststore/certs/dixi_ssl/
```

```
cdaCertificatePath=/etc/xos/truststore/certs/corehost-cda.crt
clientKeyPath=$YOUR_HOME/.xos/truststore/private/user.key
userCertificateFile=$YOUR_HOME/.xos/truststore/certs/user.crt
clientCertificatePath=$YOUR_HOME/.xos/truststore/certs/user.crt

cdaaddress.port=65000
xosdaddress.port=60000
address.port=10000

cdaaddress.host=131.254.201.16
address.host=131.254.201.20
xosdaddress.host=131.254.201.16
xosdaddress.externalAddress=131.254.201.16

Quit emacs.
```

If running **xsub**: Here you should modify:

- **useSSL** set it to false,
- **xosdaddress.host** IP address of the XOSd running on the Resource node, i.e., the address of the same node,
- **address.host** IP address of the XATI running on the Resource node (same as **xosdaddress.host**),
- **cdaaddress** IP address of the XOSd running on the Core node with running CDA service (if not sure, leave it as it is or set it to the IP of this node).

```
$ emacs $HOME/.xos/XATICAConfig.conf

useSSL=false
certificateLocation=/etc/xos/truststore/certs/xati_dummy.pem
privateKeyLocation=/etc/xos/truststore/private/xati_dummy.pem
trustStoreSSL=/etc/xos/truststore/certs/dixi_ssl/

xosdaddress.host=131.254.201.21
xosdaddress.port=55000
address.host=131.254.201.21
address.port=10000
cdaaddress.host=131.254.201.16
cdaaddress.port=60000
```

```
Quit emacs.
```

For the settings related to the **secure communication** using **SSL**, please refer to 6.6.10.

6.8.6 Configuring the RCA client

RCA runs in DIXI, which looks in /etc/xos/config/xosd_stages for the .stage files to see which services to run. The following files need to be present and have the line `enabled=true` for the DIXI to work properly:

```
RCAClient.stage
```

Please make sure that the RCA server's public certificate is installed in its place.

```
(root)# cd /etc/xos/truststore/certs/
(root)# ls rcaserver.crt
rcaserver.crt
```

6.8.7 Mount your XtreemFS Volume

XtreemFS permits you to save your data and binaries.

To see your own volume :

```
$ xtfs_lsvol http://xfs_core_ip:32636/
user-c4b32574-cf06-47e7-b960-97e7f6b994a4 -> 00065BBD8E7C900B51481CF8
```

Do you recognize you ID number ? It is user-c4b32574-cf06-47e7-b960-97e7f6b994a4. You need it to mount your XFS volume. Create a folder and mount the volume:

Is this syntax correct?

```
$ mkdir $HOME/MyVolume
(root)# xtfs_mount -o dirservice=http://xfs_core_ip, \
       volume_url=http://xfs_core_ip:32636/user-c4b32...994a4, \
       direct_io,allow_other $HOME/MyVolume
```

allow_other permits to access the data without being root.

Add a job file in your volume:

```
No root $ emacs $HOME/MyVolume/cal.jsdl

<?xml version="1.0" encoding="UTF-8"?>
<JobDefinition xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl">
 <JobDescription>
  <JobIdentification>
   <Description>Execution of cal</Description>
   <JobProject>Test</JobProject>
  </JobIdentification>
   <Application>
    <POSIXApplication xmlns:ns1="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix">
     <Executable>/usr/bin/cal</Executable>
    </POSIXApplication>
   </Application>
 </JobDescription>
</JobDefinition>

Quit emacs.
```

To umount this volume:

```
(root)# umount $HOME/MyVolume
$ ls $HOME/MyVolume
(nothing)
```

Should we keep this explanation as the user needs to be root on his node?

Mount it again:

```
(root)# xtfs_mount -o \
       volume_url=http://xfs_core_ip:32636/user-c4b32...994a4, \
       direct_io,allow_other $HOME/MyVolume

$ ls $HOME/MyVolume
cal.jsdl
```

We will use it for running the job.

6.8.8 Run a job with the AEM

do we need this ?

(for killing the XOSd if necessary)

```
(root)# service xosd stop
Ctrl+C it once the XOSd is finished.
```

XOSd must be running on 131.254.201.16 Core node and 131.254.201.21 Resource node. Link your Client XOSd to Core XOSd:

```
(root)# service xosd start
```

Check there is a registered resource :

```
$ rca_list_registered
```

To run the job :

```
$ xsub -f $HOME/MyVolume/cal.jsdl
```

this test is not correct: why should the user go to rhe resource console?
In one resource XOSd console you have :

how does the user get access to his home volume from outside a job? ssh-xos? interactive bash job?

```
September 2008
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
```

It's over. Now you can run more jobs described in the previous section or umount your XtreemFS volume:

```
(root)# umount $HOME/MyVolume
```

6.8.9 Enabling SSL Secure Connection

This section shall explain what is needed in order to enable SSL connections within an XtreemOS platform. This is a critical issue.

Enabling SSL in DIXI

The clients connect to the DIXI services using XATI or XATICA interfaces. By default, they are already configured to use the user's certificate and private key for initiating the SSL communication. To enable such communication for the clients, open **XATIConfig.conf** and **XATICAConfig.conf** in **/home/$USER/.xos/** for text editing, and set **useSSL** to **true**. Also ensure that the **sslPrivateKeyPassword** value is correct.

6.9 Special case configurations

Here we should provide more information about special cases that are useful to debug/develop the platform, but are NOT the default behaviour.

Chapter 7

Installing and configuring XtreemOS

7.1 Installing and Configuring the XtreemOS Root Certification Authority (Root CA)

The Root CA is the top level of the trust mechanism in XtreemOS. It is a critical part in the XtreemOS Public Key Infrastructure (PKI). To achieve and maintain the level of trust required by users of an XtreemOS Grid, the Root CA must be operated on only one machine. This host must be a physically-secure core node to avoid compromise of the Root CA private key, which would destroy any trust placed on the Root CA. Some organisations may choose to run the Root CA on a machine which isn't connected to a network, to eliminate any risk of intrusion.

The Root CA comprises root entity credentials which are trusted by all participants in an XtreemOS Grid, and a mechanism to create service certificates that identify other XtreemOS core services.

The package **rootca-config** contains the scripts and configuration files for creating a Root CA, and for creating service certificates from Certificate Signing Requests.

7.1.1 Installing and Configuring the Root Certificate Authority

On the machine which will be running the Root CA, install the rootca-config package :

```
(root)# urpmi rootca-config
```

This places configuration files in **/etc/xos/config/openssl**.

You may now choose to configure some of the details which will appear in the Root CA public key certificate (the 'root certificate').

The Root CA certificate is configured by properties in the file `/etc/xos/config/openssl/create-rootca-creds.conf`.
You can modify the section **[root_ca_distinguished_name]** to change the certificate fields **commonName**, **organizationName** and **organizationalUnitName** as required.
The **[req]** section contains the property **default_days** to set the duration of the certificate's validity, and **default_bits** to set the size of the Root CA private key. The root CA public certificate is valid for 730 days (two years) by default, and its private key is the recommended 2048 bits. You will probably not need to change these two values.

7.1.2 Creating the Root CA and the root credentials

Decide on a directory to hold the files related to the Root CA, for example, **/opt/xtreemosca**.

To create the Root CA directory, the private key and public certificate, run the **create-rootca** command as shown in figure 7.1.

You will be prompted for a passphrase - this protects the private key, and is required when using the Root CA to create service certificates from Certificate Signing Requests (CSRs). This passphrase must be kept secret to prevent use of the private key by anyone other than the operator of the Root CA.

The private key is created in the sub-directory **private** under the Root CA (in this case, `/opt/xtreemosca/private/xtreemos.key`). The public key certificate of the Root CA is the XtreemOS 'root certificate'. It is created in the sub-directory `public` of the Root CA directory (in this example, **/opt/xtreemosca/public/xtreemos.crt**).

The XtreemOS root certificate needs to be installed on all machines in this XtreemOS Grid. The certificate can be placed in `/etc/xos/truststore/certs/xtreemos.crt` on these machines. On Core Nodes, this allows the use of the XtreemOS Root Certificate by XtreemOS Grid Services in sending a certificate chain for SSL verification. On Client Nodes, the

```
Root # create-rootca /opt/xtreemosca
Generating a 2048 bit RSA private key
....+++
.........................................+++
writing new private key to '/opt/xtreemosca/private/xtreemos.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
Root CA Private key written to /opt/xtreemosca/private/xtreemos.key -
keep this private key secure
Root CA Public key certificate written to
/opt/xtreemosca/public/xtreemos.crt (valid until
Aug 12 15:06:42 2011 GMT).
This is the XtreemOS root certificate to be installed on all
machines in this Grid
It can be published on the VOLife home page for this Grid
Root CA public key certificate copied to
/etc/xos/truststore/certs/xtreemos.crt.
```

Figure 7.1: Creating the XtreemOS Root CA.

XtreemOS Root Certificate is use by SSL clients to verify the certificate chain received from a Grid Service.

The Root CA is now ready for its operational role. This consists of processing Certificate Signing Requests (CSRs) from administrators of core nodes, and creating service certificates for XtreemOS Grid Services (for applications such as CDA, RCA and VOPS servers, and XtreemFS client and servers). This is described in the following section.

7.1.3 Operating the Root Certificate Authority

This section is for the grid administrator, that is, the system administrator of the machine running the XtreemOS Root Certificate Authority.

The main operating mode of the XtreemOS Root CA is to take Certificate Signing Requests (CSR files) for applications and convert them to service certificates. The CSR file can be sent to the administrator of the root CA in an email message or by other means. The operator of the root CA should save the CSR file and examine it to check its validity, and contact the originator of the request if necessary (to verify its source).

The Root CA is now ready for its operational role. This consists of processing Certificate Signing Requests (CSRs) from administrators of core nodes (for applications such as CDA, RCA and VOPS servers, and XtreemFS client and servers).

In all the examples that follow, the hostname `'host'` should be replaced by either the Fully-Qualified Domain Name for the host (its DNS entry), or the IP address of the host (the latter should be seen only as a temporary measure).

NB In the following description, creating the CSRs would normally be done on the machine(s) which will run the services; this need not be the same machine that runs the Root CA.

First, we need to create a request for a certificate (a CSR file), so we install the `create-csr` package:

```
(root)# urpmi create-csr
```

The **create-csr** command creates a private key and a certificate signing request (CSR) file for an application (either a service or the XtreemFS mount client).

```
$ create-csr host "My Organisation" cda
Generating a 1024 bit RSA private key
..................++++++
..............++++++
writing new private key to 'host-cda.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
$ ls
host-cda.csr
host-cda.key
```

In this example, the private key for the CDA server is on hostname **host** written to the file **host-cda.key** and the certificate signing request is written to the file **host-cda.csr**.
The string **My Organisation** can be replaced with a suitable description of your organisation. The parameter **cda** indicates we are creating a certificate request for the CDA server.

The prompt 'Enter PEM pass phrase:' requests a passphrase to protect the private key. This passphrase should be as long as feasible (e.g. 20 charac-

ters or so). You will need to be enter this passphrase when configuring services/clients that use this private key. In the case of the CDA server, you will need to set the value of the property `cdaserver.keyPassphrase` in file `/etc/xos/config/cdaserver/cdaserver.properties` to the passphrase you have entered. These settings are described in section 7.3.2.

Now you need to run the **process-csr** command to sign the request in the file **host-cda.csr**. The first argument to this command is the location of the XtreemOS Root CA, the second is the file containing the certificate signing request.

```
root# process-csr /opt/xtreemosca host-cda.csr
Using configuration from /etc/xos/config/openssl/process-csr.conf
Enter pass phrase for /opt/xtreemosca/private/xtreemos.key:********
...

$ ls
host-cda.crt
host-cda.csr
host-cda.key
```

You can view the newly-created service certificate with the following OpenSSL command:

```
$ openssl x509 -text -in host-cda.crt -noout
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number:
            9c:11:53:54:5e:11:e0:83
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: CN=XtreemOS CA, O=XtreemOS Project, \
                OU=XtreemOS Project Root Certification Authority
        Validity
            Not Before: Sep 17 08:30:59 2009 GMT
            Not After : Sep 17 08:30:59 2010 GMT
        Subject: CN=host/cda, O=My Organization, \
                 OU=cda
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:ce:d9:fe:50:51:f7:c4:f4:bf:49:69:4b:1a:44:
                    ...
                    0c:66:dc:3f:13:63:7e:d8:eb
                Exponent: 65537 (0x10001)

        X509v3 extensions:
            Netscape Comment:
                XtreemOS Certificate
            X509v3 Subject Key Identifier:
                E5:DE:70:FA:59:56:B6:A7:64:7A:61:FA:B4:BE:D9:F9:B4:51:D1:3D
            X509v3 Authority Key Identifier:
                DirName:/CN=XtreemOS CA/O=XtreemOS Project\
                /OU=XtreemOS Project Root Certification Authority
                serial:E4:12:B5:BC:33:29:6B:3B
            X509v3 Basic Constraints:
                CA:TRUE
            X509v3 Subject Alternative Name:
                DNS:host
            X509v3 Key Usage:
                Digital Signature, Key Encipherment, Certificate Sign

    Signature Algorithm: sha1WithRSAEncryption
        2f:fc:8c:9c:6f:4d:97:27:1d:f2:0d:0e:11:a4:50:0b:c2:1a:
        ...
```

It is possible to create other Service certificates (for RCA, VOPS, etc), following the procedure above, changing the last argument passed to `create-csr` to `rca` or `vops`, and operating the Root CA to create certificates from the CSR files. In case of creating Service certificates it is important to configure services to use the appropriate private key. Also see section 7 for details regarding setup of the specific services [1].

You have to put the CDA Server, RCA Server and VOPS Service certificates in the directory `/etc/xos/trustore/certs/` on this machine. For the CDA application, the procedure is:

```
(root)# certdir=/etc/xos/truststore/certs # short-cut to reduce typing
(root)# cp host-cda.crt ${certdir}/cda.crt
(root)# chmod a+r ${certdir}/cda.crt # Anyone can read the public key
(root)# chown cdauser.cdauser ${certdir}/cda.crt
```

The corresponding private key must be placed in `/etc/xos/trustore/private`. For the CDA application, the procedure is:

```
(root)# keydir=/etc/xos/truststore/private # short-cut to reduce typing
(root)# cp host-cda.key ${keydir}/cda.key
(root)# chmod a+r ${keydir}/cda.key
#
# users 'cdauser' and 'tomcat' need to read the key
# File permissions are reduced, but key is pass-phrase protected
#
(root)# chown cdauser.cdauser ${keydir}/cda.key
# Key is owned by the 'service user' e.g. 'cdauser' for CDA
```

Protecting the private key with such a passphrase allows us to relax the filesystem security on the key, as it needs to be readable by both the CDA user account **cdauser** and the Tomcat web server account **tomcat** for the VOLife web application.

Now you need to 'rehash' the certificates - this involves creating a link to each of the certificates in with their OpenSSL hash. The OpenSSL command **c_rehash** will create hash files for all the certificates in a directory:

```
(root)# c_rehash /etc/xos/truststore/certs # or c_rehash ${certdir}
```

[1] Default VOPS certificate comes with predefined password "xtreemos", see section 7.3.6 for setting key password for VOPS.

A hash file is created for each of the certificates in the *truststore* directory. **/etc/xos/truststore/certs**. The 'stem' of the filename is a hash of the certificate contents, the suffix is either '.0' or '.1' in case of a hashing collision. You need to run the **c_rehash** command whenever certificates are added to the directory.

The service certificate is created in the file `host-cda.crt` in the current working directory. A copy of the service certificate is also stored in the 'certs' subdirectory of the root CA. In the example above, this would be **/opt/XtreemOS-CA/certs/**<**NN**>**.pem**, where `<NN>` is the index number of this certificate.. A record of the newly issued service certificate is made in **/opt/XtreemOS-CA/index.txt**, including the certificate's expiry date, revocation status, serial number and CN.

7.2 Installing and Configuring DIXI

DIXI (DIstributed Xtreemos Infrastructure) is a framework and a hosting environment for running many of the XtreemOS services. Therefore it is essential to first install DIXI before attempting to use XtreemOS.

It is not necessery to install the DIXI packages on their own, because they are the service packages' dependencies, and will install automatically. All DIXI services will use the following packages:

- **dixi-main**. The package contains the core classes needed to run DIXI, XATI and the services.
- **dixi-services**. The classes implementing the main services, needed by other services.

To obtain the command-line functionality of the services running in DIXI, additional packages need to be installed. These packages contain scripts and programs as well as runtime libraries that represent the XATI (client part of DIXI. Install XATI from the following packages:

- **dixi-xati**. XATI, the collection of API and client-side access points for running clients. It also contains sample client programs and scripts for running them.
- **dixi-cxati**. Additional libraries and programs written and compiled in C.

As a part of the package there is a Linux service named **xosd** used for running the DIXI. The service runs an instance of the XtreemOS DIXI daemon which hosts services, configured in the configuration file (**XOSdConfig.conf**

by default), provides the means for the services to communicate, and bridges services on different nodes.

The configuration files for **xosd** and all the services run by the daemon are placed in **/etc/xos/config/** by default.

User can use **/etc/init.d/xosd** script for running, stopping and restarting **xosd**. Alternatively, the command **service** can be used. Commands *start*, *stop* and *restart* are supported at this time:

```
root# service {start|stop|restart}
```

By default, the logging information is placed into **/var/log/xosd/xosd.log**. This can be changed by modifying the **/usr/share/dixi/log4j.properties** file.

The configuration file (**XOSdConfig.conf** by default) contains the following settings:

- **xosdRootDir** — the string containing the path to the DIXI daemon root. This path is used by some of the services to locate files needed for their proper operation.
- **useSSL** — indicates whether the communication should use SSL for security. Default value: *false*.
- **trustStoreSSL** — indicates the path to the folder containing the signer certificates trusted in the SSL handshakes.
- **trustStore** — indicates the path to the folder containing the signer certificates trusted in the AEM or other services.
- **certificateLocation** — the path to the public certificate used for the SSL handshakes. The certificate needs to be able to handle both server and client connections.
- **privateKeyLocation** — the path to the private key used for the SSL handshakes and for encrypting the communication.
- **networkInterface** — an optional option that tells the DIXI daemon which network interface to use for the inbound traffic, e.g., `eth0` or `eth1`. If **externalAddress**, **rootaddress.externalAddress** and **rootaddress.host** are left out from configuration (are commented out), **networkInterface** is used to set up IP proprely assuming XOSd root is running on this node.
- **xosdport** — the port the DIXI daemon will use for lisen to the inbound traffic.

- **externalAddress** — optional parameter. It defines the IP address or the host name of the gateway xosd that nodes from other subnets can use to connect to this node. If this node is running behind a firewall, then the router or firewall that responds on the external address IP needs to forward the inbound traffic arriving to port **xosdport** to this node. If the local subnet has multiple DIXI nodes, then one of the nodes will act as proxy, and the the firewall needs to forward the traffic on the port number as configured in that node's **XOSdConfig.conf**'s **xosdport** value. If the parameter is omitted, the same address is used as that of the xosd's host.

- **xmlport** — the port used for the inbound XML connections from C applications using XATICA C library.

- **rootaddress.host** — the IP address or the host name of the **root DIXI daemon**. Can be the node's own address for a stand-alone or a root DIXI daemon. This address needs to be the one the root DIXI daemon's host's local address, as seen from the peers in the host's subnet. If the host is running on a different subnet than this node, the rootaddress.externalAddress has to be set to the address exposed to the current node.

- **rootaddress.port** — an optional parameter defining the port number the **root DIXI daemon** is listening to for requests. Default value is 60000.

- **rootaddress.externalAddress** — the IP address or the host name of the node visible from external network nodes. The node with this IP needs to have the port **rootaddress.port** forwarded to one of the nodes that on the internal network run the DIXI daemon.

- **xosdStagesSubDirectory** — the directory to contain **.stage** files which define what services need to run in the xosd. If the directory is relative, the xosd will assume it is a subdirectory of the configuration path (please refer to the `-C` command-line option in **??**). By default, the **xosd_stage** value is used, meaning that the files are expected to be in **/etc/xos/config/xosd_stages/**.

The xosd will look for the files with names ending with **.stage** in the designated directory. The files may be empty, or contain certain options. Depending on each of the file's name and contents, one service will be run per file. The **.stage** files can contain the following options:

- **service** — the name of the service to be loaded. If the option is omitted, the file's name without the **.stage** suffix will be used.

- **numThreads** — the number of threads to start the service in. The default value is 1.

- **enabled** — can take values **true** (default value) or **false**. This option lets the user to disable loading of a certain service even if the respective **.stage** file is present.

Figure 7.2 shows an example **XOSdConfig.conf** which configures the local DIXI daemon to connect to a *root daemon* running at the address myroot.xlab.si. The communication will be SSL-encrypted, using the SSL credentials of the local node. The daemon will start up and serve the services defined by the **.stage** files in **/etc/xos/config/xosd_stages**.

```
# This is a sample configuration.
# The lines starting with the hash # are ignored.

# Look for the .stage files here
xosdStagesSubDirectory=/etc/xos/config/xosd_stages

# Security and SSL-related settings
trustStore=/etc/xos/truststore/certs/
useSSL=true
trustStoreSSL=/etc/xos/truststore/certs/
certificateLocation=/etc/xos/truststore/certs/resource.crt
privateKeyLocation=/etc/xos/truststore/private/resource.key

# Node's parameters and root addresses
xosdRootDir=.
networkInterface=eth0
xosdport=60000
xmlport=55000
externalAddress=testnode2.xlab.si
rootaddress.host=myroot.xlab.si
rootaddress.externalAddress=gateway.xlab.si
rootaddress.port=60000
```

Figure 7.2: A sample DIXI daemon configuration file.

When the xosd starts, the hosted services become available at a common port of the selected or configured network interface. The Java services and clients would use the port 60000 to send their service messages, and the C clients would use the port 55000 to exchange the XML-encoded service messages.

The services use the **communication address** structure when addressing their service messages. The communication address is composed of the target host address, the port number, and, optionally, the subnet's gatway (external address). For example, `192.168.0.30:60000(194.249.173.87)` represents a ser-

vice running on an xosd that uses host address 192.168.0.30, accepts the incoming messages at port 60000, and is running on a subnet which has a gatway xosd running on host with address 194.249.173.87.

The **.stage** files in the **/etc/xos/config/xosd_stages** directory define which services will load. On a system installed using the `urpmi` packages provided by the XtreemOS consortium, this directory will already contain all the needed files to start the services. What follows is a reference for the advanced users who wish to have a higher level of control over what services run on the system.

For example, the directory with stages may contain the following:

```
$ ls -l /etc/xos/config/xosd_stages
-rw-r--r-- 1 [...]  133 [...] DaemonGlobal.stage
-rw-r--r-- 1 [...]    0 [...] eu.xtreemos.xosd.security.
rca.server.service.RCAServerHandler.stage
-rw-r--r-- 1 [...]  158 [...] RCAClientHandler.stage
```

This listing suggests the xosd will load three stages. The RCAServerHandler's stage file is empty, therefore the **eu.xtreemos.xosd.security.rca.server.service.RCAServerHandle** will be used as the class name for the service to load. DameonGlobal.stage contains the following:

```
$ cat /etc/xos/config/xosd_stages/DaemonGlobal.stage
service=eu.xtreemos.xosd.daemon.DaemonGlobal
enabled=true
numThreads=1
```

This means that the **eu.xtreemos.xosd.daemon.DaemonGlobal** will be the class name used when instantiating the service. Finally,

```
$ cat /etc/xos/config/xosd_stages/RCAClientHandler.stage
service=eu.xtreemos.xosd.security.rca.client.service.RCAClientHandler
enabled=false
numThreads=5
```

the RCAClientHandler would not be loaded, because the content of the file tells the xosd that the service is disabled.

7.2.1 Connecting DIXI daemons from multiple nodes

In the first release, the DIXI daemons need to be connected explicitely to a central daemon. This is obviously an unscalable solution, therefore in the future releases the interconnections will occur automatically via ADS. Currently, however, there should be a designated root node for the other nodes to connect to. In the test phase, this can be 194.249.173.88 at port 60000, or another designated node within your LAN.

To connect the xosd to another xosd, use either the **-s** command-line directive, or the **rootaddress.host** and **rootaddress.port** settings in the **XOSdConfig.conf**.

7.2.2 Secure communication (SSL)

In the `/etc/xos/config/XOSdConfig.conf`, the following options are relevant for the SSL communication in DIXI:

- **useSSL** — to enable the secure commuincation, set the value to `true`. Otherwise, set it to `false`.
- **trustStoreSSL** — set the value to be the path to the trusted CA certificates. The default `/etc/xos/truststore/certs/` should work.
- **certificateLocation** — set the value to the full path of the certificate to be used for the SSL. By default, this can be the resource's machine certificate `/etc/xos/truststore/certs/resource.crt`.
- **privateKeyLocation** — set the value to the full path of the private key that will be used for encrypting the communication. By default, this can be the resource's machine private key `/etc/xos/truststore/private/resource.key`.
- **privateKeyPassword** — specify the password for reading the private key (`12345678` by default).

After the modifications are saved, restart the service:

```
(root)# service xosd restart
```

The client configuration needed by the root user commands needs to be modified accordingly. Open the `/root/.xos/XOSdConfig.conf` and modify the following options:

- **useSSL** — set the value to `true` to enable the secure communication, or to `false` otherwise.
- **trustStoreSSL** — set the value to be the path to the trusted CA certificates. The default `/etc/xos/truststore/certs/` should work.
- **certificateLocation** — set the value to the full path of the certificate to be used for the SSL. We recommend the use of the adminstrator's user certificate `/root/.xos/truststore/certs/user.crt`.

- **privateKeyLocation** — set the value to the full path of the private key that will be used for encrypting the communication. We recommend the use of the adminstrator's user private key `/root/.xos/truststore/private/user.key`.
- **sslPrivateKeyPassword** — specify the password for reading the private key. If the **privateKeyLocation** is the same as the **userKeyFile** (i.e., the user's private key is used), then the value is ignored and the typed-in password will be used.

Similar procedure should be followed for all the other users' accounts: modifying the `/home/USERNAME/.xos/XATIConfig.conf` according to the instructions above. In the case of non-administrator users, they should use their own user certificate and private key stored in their `/home/$USERID/.xos/truststore` folder).

For the commands and clients, written in C, open `/home/USERNAME/.xos/XATICAConfig.conf` and modify the following options:

- **useSSL** — set the value to `true`.
- **certificateLocation** — set the value to the full path of the certificate to be used for the SSL.
- **privateKeyLocation** — set the value to the full path of the private key that will be used for encrypting the communication.

7.2.3 Traversing NAT

The DIXI framework supports connecting and communicating with nodes that are on another subnet and behind the firewall. However, this only works if the target node's subnet contains a node that runs xosd, is visible from external networks by port-forwarding, and the target node's firewall allows inbound and outbound traffic to the visible node on port designated for the xosd communication (60000 by default). For proper operation, every xosd needs to have **XOSdConfig.conf** configured so that **externalAddress** contains the externally visible address that maps into the local LAN address space and which runs its own xosd.

7.2.4 Running xosd as root

In principle, an ordinary user can run xosd. However, many of the services are VO-critical, they run on a secure node or access local resources. In these cases, the **xosd** needs to be run with as root. For running xosd use **/etc/init.d/xosd** script with command *start* or *restart*.

7.2.5 Stopping xosd

The **xosd** can be stopped by sending the break (Ctrl+C) signal to the process, or using **xosdkill** XATI script. Prefered way is to use **/etc/init.d/xosd** script with command *stop* or just executing following command:

```
[root@localhost ~]# service xosd stop
```

7.2.6 DIXI logging

Property file **/usr/share/dixi/log4j.properties** assigns path to target file for logging facility used in DIXI. By default it points to **/var/log/xosd/xosd.log** but someone can easily change property file to point to different location. There is an issue when running XATI (described below) in user mode without superuser rights. Since XATI uses the same logging facility and the same property file, there can be an error message saying that user does not have rights to append to target file. By changing the line in **log4j.properties**

```
...
log4j.appender.A3.File=/var/log/xosd/xosd.log
...
```

this permission issue can be easily resolved.

7.2.7 XATI

The client programs use XATI to access the functionality of the services. XATI needs a running DIXI daemon to connect to. This can be a daemon running on any node accessible from the client node. However, it is preferable to run an instance on the same node that runs the XATI program. One possible issue of connecting to a remote daemon is that the daemon can open multiple ports towards the client, while two daemons use a single channel to communicate.

The XATI in all client programs use **XATIConfig.conf** file to read the settings from. By default, the configuration file is located in the **/home/username/.xos/** folder. The settings are as follows:

- **address.port** — the port number used by XATI to communicate with xosd.
- **networkInterface** — an optional option that tells the XATI which network interface to use for the inbound traffic, e.g., `eth0` or `eth1`.

- **xosdaddress.host** — the address of the DIXI daemon (xosd) this XATI program is connecting to.
- **xosdaddress.port** — the port number used by the DIXI daemon (xosd) this XATI program is connecting to.
- **xosdaddress.externalAddress** — the externally visible address of the xosd this XATI program is connecting to.
- **userCertificateFile** — the path and filename of the certificate issued by the CDA to the user. Usually, the user certificates are placed in **/home/username/.xos/trusts**
- **useSSL** — indicates whether the communication should use SSL for security. Default value: *false*.
- **trustStoreSSL** — indicates the path to the folder containing the signer certificates trusted in the SSL handshakes.
- **certificateLocation** — the path to the public certificate used for the SSL handshakes. The certificate needs to be able to handle both server and client connections.
- **privateKeyLocation** — the path to the private key used for the SSL handshakes and for encrypting the communication.
- **userKeyFile** — this entry points to the path of the user private key file
- **address.host** — the address of the XATI daemon

Figure 7.3 shows a sample configuration file for client programs using XATI.

7.2.8 Logging in XATI

In the client applications it is best that the debugging information does not show in the user's console. However, for occasional diagnostic reasons it is useful to be able to track what the libraries did. By default, the log file is placed into **/home/user/.xos/xos-xati.log** file.

The logging behaviour of the client commands and applications can be altered by editing the logging configuration file that is by default located in **/usr/share/dixi/log4j_xati.properties**.

```
# This is a sample XATIConfig.conf.

# Use this port for sending XATI requests
address.port=10000

# xosd details
xosdaddress.externalAddress=192.168.0.178
xosdaddress.host=192.168.0.178
xosdaddress.port=60000

# SSL
useSSL=false
trustStoreSSL=/etc/xos/truststore/certs/dixi_ssl/
certificateLocation=/etc/xos/truststore/certs/xosd_dummy.pem
privateKeyLocation=/etc/xos/truststore/private/xosd_dummy.pem

# Security and credentials
userCertificateFile=/home/matej/.xos/truststore/certs/user.crt
userKeyFile=/home/matej/.xos/truststore/private/user.key

address.host=192.168.0.178
```

Figure 7.3: A sample XATI configuration file.

7.3 Virtual Organization Management

7.3.1 Configuring X-VOMS

X-VOMS (XtreemOS Virtual Organization Management Service) is an advanced Virtual Organisation (VO) management service for supporting secure and flexible collaborations and resource sharing among people, projects and organisations. It is written in Java and back by a (Hibernate-based) X-VOMS database schema. Like other VO management software packages, X-VOMS provides a set of APIs for managing identity, attributes, and VO membership of users and resources.

X-VOMS can be used as a backend of different presentation frontends: a web application (allowing the access via a web browser), and a OS daemon service (allowing the access via a OS command line console, or directly from a user application). In *the current release*, X-VOMS is not a standalone service. It attaches to the VOLife web frontend to provide (part-of) its VO management capabilities to end users. In the future releases, the daemon frontend of X-VOMS will be offered so that applications can directly utilize X-VOMS functionalities.

X-VOMS manages, but does not distribute, credentials. It can be used with a Certification Authority (CA), such as the Credential Distribution Authority (CDA) service developed by the XtreemOS project, or a third-party attribute authority, to disseminate credentials.

X-VOMS also supports home volume creation for users of XtreemFS, a Grid file system being developed in the XtreemOS project.

This instruction assumes you know how to use MySQL (e.g. how to add a user in MySQL). For user management in MySQL, please read:

`http://dev.mysql.com/doc/refman/5.0/en/adding-users.html`

Your MySQL server will have a root password set when you the database preparation script. You can reset the MySQL root password at any time by following the procedure outlined at the following URL:

`http://bit.ly/resetMySqlPassword`

Software prerequisites

The current X-VOMS implementation relies on the following software:

- Hibernate 3.0[2]
- MySQL version 5.x.x[3]
- C3P0 Connection pooling library [4]

Install MyQSL:

```
# urpmi mysql
```

Major files and their location

The steps needed to create the X-VOMS database and load it with data are encapsulated in the script xvoms_prepare_database.sh.

Configure the X-VOMS database:

```
Root # /usr/share/xvoms/bin/xvoms_init.sh
```

[2]http://www.hibernate.org/
[3]http://www.mysql.com
[4]http://www.mchange.com/projects/c3p0/index.html

Running this script will create a database schema, populate it with some examples entries, and prompt for a database root password. This is sufficient to allow the following steps 'Installing the CDA' (7.3.2), 'Installing VOLife' (7.3.3) etc to be performed.

The VOLife administrator account is created in the above step. The user 'admin', password 'xtreemos-admin' is used to approve user applications to use this XtreemOS Grid.

The following files described below are merely described for reference purposes.

The major configuration files are located at `/etc/xos/config/xvoms`, with Log4J logging defined in `/usr/share/xvoms/log4j.properties`.

The X-VOMS library (xvoms-version.jar) is located at: `/usr/share/java/`.

- `/etc/xos/config/xvoms/hibernate.cfg.xml` a Hibernate configuration file for setting Hibernate connection properties. The most notable settings are:

```
<property name="connection.url">jdbc:mysql://localhost/xvoms
</property>
<property name="connection.username">volifecycle</property>
<property name="connection.password">xosvo</property>
```

- `/usr/share/xvoms/log4j.properties` a log4j configuration file for setting hibernate logging properties. The most notable settings are:

```
log4j.logger.org.hibernate=fatal
log4j.logger.org.hibernate.SQL=fatal
```

These are the default settings for an initial installation of XtreemOS, with XtreemFS services running on a single core node. In generak, the MRC server for XtreemFS is more likely to be located on a different server.

7.3.2 Configuring and Running a Credential Distribution Authority (CDA) Server

This sub-section is for a Grid administrator running a CDA server.

The Credential Distribution Authority is implemented in the **cdaserver** package. There is a single instance of a CDA server in an XtreemOS Grid.

The standalone CDA client program can be used to obtain user VO credentials from the CDA, and is provided by the **cdaclient** package.

The CDA server issues XOS certificates to users. The server needs an service certificate issued by the Root CA to authenticate itself to the corresponding CDA client. This service certificate can be obtained by the procedure described in section 7.1.3. This procedure also produces a private key, which should be placed into `/etc/xos/truststore/private/cda.key`. The service certificate contains the service's public key, and should be placed in `/etc/xos/truststore/certs/cda.crt`.

```
(root)# certdir=/etc/xos/truststore/certs # short-cut to reduce typing
(root)# cp host-cda.crt ${certdir}/cda.crt
(root)# chmod a+r ${certdir}/cda.crt # Anyone can read the public key
(root)# chown cdauser.cdauser ${certdir}/cda.crt
```

The CDA private key must be placed in `/etc/xos/trustore/private/cda.key`.

```
(root)# keydir=/etc/xos/truststore/private # short-cut to reduce typing
(root)# cp host-cda.key ${keydir}/cda.key
(root)# chmod a+r ${keydir}/cda.key
#
# users 'cdauser' and 'tomcat' need to read the key
# File permissions are reduced, but key is pass-phrase protected
#
(root)# chown cdauser.cdauser ${keydir}/cda.key
# Key is owned by the 'service user' e.g. 'cdauser' for CDA
```

Now you need to link the certificates with their hash. The OpenSSL command `c_rehash` will create hash files for all the certificates in a directory:

```
(root)# c_rehash /etc/xos/truststore/certs # or c_rehash ${certdir}
```

A hash file is created for each of the certificates in the directory. The 'stem' of the filename is a hash of the certififcate contents, the suffix is either '.0' or '.1' in case of a hashing collision. The `c_rehash` command needs to be run on the directory whenever certificates are added to it.
As root, install the CDA server via:

```
Root # urpmi cdaserver
```

The following aspects of the CDA server are configurable by setting values in the file `/etc/xos/config/cdaserver/cdaserver.properties`:

- **cdaserver.keyFilename** — private key of CDA server - must be kept secure, readable only by owner.
- **cdaserver.keyPassphrase** — the private key is secured by a passphrase, the longer the better.
- **cdaserver.certFilename** — public key certificate of CDA server.
- **xtreemos.rootCertificate** — public key certificate of root CA.
- **cdaserver.sslAlgorithm** — cipher used by SSL.
- **cdaserver.sslHandshakeCipher** — the cipher used in initial SSL key exchange.
- **cdaserver.signatureAlgorithm** — algorithm used to sign the XOS-certificate returned to user.
- **cdaserver.validityDays** — number of days that certificate is valid for
- **cdaserver.validityHours** — number of hours that certificate is valid for
- **cdaserver.validityMinutes** — number of minutes that certificate is valid for

In general, you should only need to change the property `cdaserver.keyPassphrase` to the passphrase you set when running the `create-csr` command for the CDA.

The validity of a certificate is calculated as (cdaserver.validityDays) days + (cdaserver.validityHours) hours + (cdaserver.validityMinutes) minutes. Any two of these values can be zero. Hence, the lifetime of certificates issued by the CDA server can be set on a fine basis, if required.

Other aspects of the CDA server operation are:

Connection to X-VOMS database - this is set in `hibernate.cfg.xml`

The level of logging, log file location, etc, are defined in log4j.properties. The server writes its log files in `/var/log/cdaserver/cdaserver.log` by default.

Once configured, the server is started by issuing the following command:

```
Root # /sbin/service cdaserver start
```

7.3.3 Installing VOLife

Virtual Organization Lifecycle Management(VOLife) is a web-based tool for accessing various VO-related services in XtreemOS. Currently VOLife, supports the manipulation of the X-VOMS database and the generation of private keys and XOS-Certs for users. Integration with runtime security services such as VOPS and RCA is still under development.

VOLife consists of two parts: backend and frontend. The backend is a light java wrapper around current security libraries. The frontend is a web application to be deployed into Tomcat. The backend provides a command-line utility which has almost the same functionality as the web front-end. But its main purpose is to test the integrity of data and the recommended way to use VOLife is via the web frontend.

Prerequisites

The prerequisites of VOLife include:

- Tomcat 4.x or higher
- JRE 1.6 or higher

The web frontend of VOLife is written in JSP and is deployed into Tomcat.

The running of VOLife also depends on the installation of XVOMS database. Please see the command in 7.3.1.

Installation

To install VOLife, use the following command:

```
# urpmi volife
```

General configuration

By default, VOLife is installed as a web application in the `webapps/` directory of Tomcat home directory (i.e. $CATALINA_HOME).

When generating XOS-Certs, VOLife uses settings in `/etc/xos/config/volife/volife.properties` to locate the CDA private key and certificate, and to supply the pass-phrase protecting the private key. These

settings can be copied from the CDA configuration in
/etc/xos/config/cdaserver/cdaserver.properties.

```
$ cat /etc/xos/config/volife/volife.properties
cdaserver.keyFilename=/etc/xos/truststore/private/cda.key
cdaserver.keyPassphrase=changeme
cdaserver.certFilename=/etc/xos/truststore/certs/cda.crt
```

The key pass-phrase should be changed to the actual pass-phrase protecting the private key.

The VOLife webapp puts the generated user private key and XOS-Cert files in the certs/ directory. The path of the directory certs/) is relative to the directory from which Tomcat is launched. That means:

- For Tomcat, there should exist certs/ directory under Tomcat webapps/volifecycle directory, and the user ID that Tomcat runs under (generally named tomcat) should have the write permission on the certs/ directory.
- For the command line utility volife_run.sh, there should exist the certs/ directory mentioned above.

To make sure your installation of VOLife works normally, check the directories as follows:

```
# cd /usr/share/tomcat5
# ls -l
```

There should have some lines indicating the certs/ directory exists and has the right permissions:

```
drwxrwxr-x 2 tomcat tomcat 4096 2008-07-10 13:09 certs/
```

You should ensure that the MySQL database and Tomcat web server have been started before accessing the VOLife. The following lines are to check the services' status:

- Check if the MySQL service is running:
  - /etc/init.d/mysqld status
- Check if the Tomcat service is running:
  - /etc/init.d/tomcat5 status

Consolidating Security of VOLife Server

By default, an encrypted connection is not opened in tomcat. This could form a security weakness where users might be tricked into presenting their username/password to a hijacked VOLife server.

Currently, the VOLife server is built on Tomcat in XtreemOS, so we can provide an SSL connection between web browser and VOlife webapp by authenticating the server.

(1) Prepare a local certificate keystore. The keystore file is used in Tomcat to store certificate. In fresh tomcat, the keystore file can be created by following command:

```
$JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA
  -keystore /path/to/my/keystore
```

You can specify its access pasword in this step. The defaut keystore file is located in user's home directory, named `.keystore` and given `changit` as password.

(2) Create a Certificate Signing Request (CSR) for VOLife server. The certificate request should be signed by XtreemOS root certificate. By following command:

```
$JAVA_HOME/bin/keytool -certreq -keyalg RSA -alias tomcat
-file volife_server_certreq.csr
-keystore /path/to/my/keystore
```

If a custom keystore file has been specified in (1), it should be presented explicitly here.

(3) Send the CSR file to the operator of the XtreemOS root Certification Authority. Currently, the request file for the server has to be signed manually (Sending to administrator of XtreemOS root certificate by email).

(4) Import the server certificate and root certificate into local keystore. After signed certificate for VOLife server returned, it can be imported into Tomcat keystore file. Also, XtreemOS root certificate need to be import for authentication of chain certificate. The following instructions can help import both certifcates:

```
$JAVA_HOME/bin/keytool -import -alias root
  -keystore /path/to/my/keystore
  -trustcacerts -file /path/to/xtreemos_root_cert
```

```
$JAVA_HOME/bin/keytool -import -alias tomcat
  -keystore /path/to/my/keystore
  -trustcacerts -file /path/to/volife_server_cert
```

If above commands throw a error message:
`keytool error: java.security.cert.CertificateParsingException: invalid DER-encoded certificate data`
the certificates have to be convert to DER format first and then reimport in local keystore, because JAVA keytool only use DER format to keystore. The command:

```
openssl x509 -in volife_server_cert -inform PEM
-out volife_server_cert.der -outform DER
```

help to convert the PEM format to DER format.

(5) Edit the Tomcat configuration file to open secure socket. Tomcat need to be edited its `<Connector>` in the `$CATALINA_HOME/conf/server.xml` file, where `$CATALINA_HOME` represents the directory into which you installed Tomcat. Find and comment the non-SSL section:

```
<!-- Define a non-SSL HTTP/1.1 Connector on port 8080 -->
<Connector port="8080" maxHttpHeaderSize="8192"
       maxThreads="150" minSpareThreads="25"
       maxSpareThreads="75" enableLookups="false"
       redirectPort="8443" acceptCount="100"
       connectionTimeout="20000"
       disableUploadTimeout="true" />
```

and then, open the SSL-support section and edit the parameters as follows:

```
<!-- Define a SSL HTTP/1.1 Connector on port 8443 -->
<Connector protocol="HTTP/1.1"
       port="8443" maxHttpHeaderSize="8192"
       maxThreads="150" minSpareThreads="25"
       maxSpareThreads="75" enableLookups="false"
       disableUploadTimeout="true" acceptCount="100"
       scheme="https" secure="true" SSLEnabled="true"
       keystorePass="keystorepassword"
       keystoreFile="/path/to/my/keystore"
       debug="1" >
       <Factory clientAuth="false" protocol="TLS" />
</Connector>
```

- Restart Tomcat and test the webapp in browser. This final step is to test whether the SSL socket is in function. Restart your Tomcat service and try:

  `https://localhost:8443`

  If this works, it should prompt you to choose whether you accept certificate.

  I need to provide instructions in the 'Using VOLife' section on how users can import the XtreemOS root certifcate into their browser, to allow the SSL connection to the VOLife webapp to be trusted. Ian 24/10/08. Still need to do this. Ian 02/06/09

FAQ

(1) Since the IcedTea JVM are not completely same as Sun JVM, the difference may cause the malfunction of SSL socket. The symptoms is like this:

```
...
Using CATALINA_HOME:   /usr/share/tomcat5
Using CATALINA_TMPDIR: /usr/share/tomcat5/temp
Using JRE_HOME:
- Error initializing endpoint
java.io.IOException: Invalid keystore format
        at sun.security.provider.JavaKeyStore.
         engineLoad(JavaKeyStore.java:650)
        at sun.security.provider.JavaKeyStore$JKS.
         engineLoad(JavaKeyStore.java:55)
        at java.security.KeyStore.load(KeyStore.java:1201)
        at org.apache.tomcat.util.net.jsse.JSSESocketFactory.
         getStore(JSSESocketFactory.java:287)
        at org.apache.tomcat.util.net.jsse.JSSESocketFactory.
         getTrustStore(JSSESocketFactory.java:261)
...
```

One of the solution is replacing the IcedTea JVM with Sun JVM.

To replace the IcedTea JVM with Sun JVM, you have to first download and install the Sun's JDK (recommended JDK1.6). And then `JAVA_HOME` and `JRE_HOME` have to be speficied to Sun's JVM.

- Edit `.bashrc` to add the new path of `JAVA_HOME` and `JRE_HOME` as global directory. Also, add the `$JAVA_HOME/bin` to global path.

```
...
export JAVA_HOME=/path/to/jdk1.6.0_06/
export JRE_HOME=/path/to/jdk1.6.0_06
PATH=$JAVA_HOME/bin:$PATH
...
```

- Edit `/usr/bin/tomcat5` and add the following path at file header.

```
...
# OS specific support.  $var _must_ be set to either true or false.
JAVA_HOME=/path/to/jdk1.6.0_06
JRE_HOME=/path/to/jdk1.6.0_06

cygwin=false
...
```

Finally, restart your Tomcat and try again.

(2) When SSL socket is enabled in VOLife under SUN JDK environment, another problem occurred due to the incompatibility of some IcedTea jar packages. A symptom is VOLife can not download user's certificate. And, in Tomcat log file, you can find the following exception message:

```
java.io.IOException: problem creating RSA private key:
java.io.IOException:
exception using cipher: java.lang.SecurityException:
JCE cannot authenticate the provider BC
...
```

This results from the IcedTea's encryption libraries are not compatible with Sun's. To solve the trouble, administrator has to replace the following BouncyCastle jar package (which has been compiled with the IcedTea JDK): `/usr/share/java/bcprov-1` with the BouncyCastle jar package, compiled with the Sun JDK: `bcprov-jdk16-139.jar` . This can be downloaded downloaded from:http://www.bouncycastle.org/.

7.3.4 RCA

The Resource Certification Authority services run as DIXI services. Before installing it, please first install DIXI (Section 7.2).

RCA comes in two packages:

- **dixi-vom-rca-node** — This package contains the node level service which should run on each node capable of executing jobs.

- **dixi-vom-rca-server** — This package contains the core-side service which usually runs on one node within a physical organisation.

To install the necessary software, simply use `urpmi` with the name of the package. In order to actually run either RCA client or the RCA server, the DIXI daemon's configuration file (**XOSdConfig.conf** by default, please refer to Section 7.2 for more details) needs to have its handler enabled. The configuration files used by the RCA are placed into **/etc/xos/config/** by default.

For the normal operation of the RCA client, the node running the RCA client's service also needs to run the AEM's Resource Monitor.

Enabling services in DIXI daemon's configuration.

To have one or both services start with the local DIXI daemon, place an empty file with proper file name into `/etc/xos/config/xosd_stages`, named after the class that starts the stage. The names are as follows:

- **RCA server**: eu.xtreemos.xosd.security.rca.server.service.RCAServerHandler.stage
- **RCA client**: eu.xtreemos.xosd.security.rca.client.service.RCAClientHandler.stage

For more information on the service start-up options, please refer to Section 7.2.

Configuring core-level RCA service.

The RCA server service creates and uses the **RCAServerConfig.conf** to obtain the configuration:

- **certDNLocation** — the location of the organisation covered by the RCA server. The value is a part of the distinguished name (DN) of a certified resource.
- **certDNCountry** — the country of the organisation covered by the RCA server. The value is a part of the distinguished name (DN) of a certified resource.
- **certDNOrganisation** — the name of the organisation covered by the RCA server. The value is a part of the distinguished name (DN) of a certified resource.
- **certDNOrganisationUnit** — the name of the organisation unit covered by the RCA server. The value is a part of the distinguished name (DN) of a certified resource.

- **daysCertValidity** — The number of days the certificate will be valid, starting from the day of certification and expiring this number of days later.
- **privateKey** — The path to the server's certificate authority's private key.
- **certificateFileName** — The path to the server's certificate authority's public key/certificate.
- **cdaPassword** — The server's certificate authority's public key's passphrase.
- **keyPassword** — The server's certificate authority's private key's passphrase.
- **rcaDBFile** — The path to the file containing the RCA DB.
- **attributeType** — the type of the attribute certificates. Use V2 for attribute certificates, or V3 for certificates with attributes stored in extensions. The default value is V3, and it is a recommended value for compatiblity with openssl libraries.

The RCA server requires a private key and a certificate signed by a certification authority that is trusted by the nodes in the XtreemOS. The RCA server will use the certificate signed by a commonly trusted root authority to sign the machine certificate requests. The steps for creating the certificate for the RCA are similar to those described in Section 7.3.2 for the CDA server. The location of the private key and the certificate are defined by **privateKey** and **certificateFileName** of the **RCAServerConfig.conf**, respectively.

Configuring node-level RCA service.

The RCA client service creates and uses the **RCAClientConfig.conf** to obtain the configuration:

- **cdaCertificateFileName** — the path to the RCA server's certificate authority's public key/certificate.
- **resPrivateKeyFileName** — the path to the resource's private key.
- **resIdentityCertFileName** — the path to the resource's identity certificate (public key).
- **resAttributeCertFileName** — the path to the resource's attribute certificate (attribute certificate).
- **resAttributeCertExtFileName** — the path to the resource's attribute certificate (attributes stored in an extension).

- **resVOAttributeCertIncoming** — the path to the folder that will store the attribute certificates pushed from the RCA Server.

7.3.5 Preparing Core Services - CDA, RCA, and VOPS servers, XtreemFS servers and XtreemFS mount client

Generic instructions for preparing core services

The XtreemOS core services require configuring with an service certificate before they can be started. In addition, services using the DIXI message bus require configuring of the DIXI and XATI subsystems (see sections 7.2).

Most service certificates are used to authenticate core services to client programs. For mounting XtreemFS filesystems, one mode of use is to use the service certificate for the `xtfs_mount` client to authenticate the host running the client to the XtreemFS server. Alternatively, the XtreemFS mount client can use an XOS-certificate if the client is being run on behalf of a single user.

Prerequisite for installing any core service application.

The following conditions apply:
Before installing any server, the XtreemOS Root Certificate Authority must be active in your XtreemOS Grid. See Section 7.1 for details.
The `create-csr` package must be installed; it contains the **create-csr** command and an OpenSSL configuration file to create a certificate signing request (CSR) file for a service. This CSR needs to be sent (e.g., by email) to the operator of the Root CA to obtain the service certificate.

The steps involved are shown below.

The **create-csr** command creates a Certificate Signing Request (CSR) file. Install this command via

```
Root # urpmi create-csr
```

The arguments to the **create-csr** command are:

- the host name — this is encoded in the subjectAltName extension field of the certificate, and as part of the Subject CN field.
- the name or description of your organisation.
- the name of the application. This is incorporated into the Subject CN field as <fqhn>/<application>. E.g. for a CDA server at host.org.domain, the Subject field would include `CN=host.org.domain/cda`.
  Legitimate values for the application argument are:

- `cda` The Credential Distribution Authority server
- `vops` The VO Policy Service server
- `mrc` The XtreemFS Metadata and Replica Catalogue Server
- `dir` The XtreemFS Directory Service
- `osd` The XtreemFS Object Storage Device server
- `xtfs_mount` The XtreemFS mount client

An example, creating a request for a service certificate, where `host.org.domain` is replaced with either the Fully-Qualified Domain Name for the host, or its IP address. The last argument to this command identifies the type of service/client that this certificate will be used by.

```
create-csr host.org.domain "My Organization" cda
```

Figure 7.4: Creating a request for a CDA service certificate.

This command produces a private key for the application in `host.org.domain-cda.key`, and a CSR in `host.org.domain-cda.csr`. Send this CSR file to the administrator of the Root CA in your organization to get an service certificate (e.g. `host.org.domain-cda.crt`) in return. Install this service certificate in `/etc/xos/truststore/cer` and the private key in `/etc/xos/truststore/private/cda.key`.

The passphrase protecting the key can be specified in the properties/configuration file of the server it is to be used with. In this case, you must ensure that the file containing the passphrase is only readable by the owner of the service itself, e.g. for the CDA server, the properties file should only be readable by 'cdauser'.

Connecting the CDA server to the X-VOMS database

The CDA server uses the Hibernate ORM library to retrieve VO attributes from the X-VOMS. Hibernate uses a JDBC connection that is specified by the parameters in the Hibernate configuration file, `hibernate.cfg.xml`. The settings that may need to be changed here are `connection.username` and `connection.password`.

7.3.6 VOPS

VOPS is a **core-level service** which, due to usage of the DIXI framework, runs as a service using DIXI communication stages. Please refer to Section 7.2 for details. VOPS has to be started in a way like other XOS daemons are: using

xosd script provided in a bundle containing VOPS package. First, administrator has to set up **XOSdConfig.conf** and **VOPSConfig.conf** appropriately. **ResMng.conf** (on server, where ResMng service is running) has to be configured appropriately to use VOPS, see also figure 7.6. VOPS is a server primarily intended serving requests and forwarding answers from/to resource discovery services and therefore it needs private key and public certificate to be able to digitally sign its decisions before forwarding them to services. Services querying VOPS should have access to VOPS public certificate to be able to check authenticity of its answers. To obtain VOPS server key/certificate please refer to section 7.3.2 where steps for obtaining server certificate is described.

To be able to run VOPS server using DIXI framework, place an empty file with proper file name into `/etc/xos/config/xosd_stages`, named after the class that starts the stage:

```
eu.xtreemos.xosd.security.vops.service.VOPSHandler.stage
```

For more information on the service start-up options, please refer to Section 7.2.

If VOPSConfig.conf does not exist yet, you can run xosd and stop it. This way VOPSConfig.conf is automatically generated under /etc/xos/config, where you can edit it manually (see figure 7.5).

```
enableAccessControl=true

VOAdminRoles.size=15
VOAdminRoles.0=role_get_VOAttributes4

ResourceAdminRoles.size=15
ResourceAdminRoles.0=res_role_get_VOAttributes

serviceKey=/etc/xos/truststore/private/vopsserver.pem
policyStorage=/usr/share/dixi/VOPS/files/policy/testStorage
keyPassword=xtreemos
```

Figure 7.5: A sample VOPS configuration file.

- **globalVOPS.port** and **globalVOPS.host** legacy settings that are not used, so they can be safely comment out or ignored.
- **enableAccessControl** enables or disables access control: if enabled, extension (role) from user certificate is checked whether it is one from roles listed under **VOAdminRoles** or **ResourceAdminRoles**.

- **VOAdminRoles.size** is the size of array defining VO administrator roles.
- **VOAdminRoles** are roles of users which are permitted to manipulate with XACML policies. These roles must be same as roles specified in certificates (VO administrator roles).
- **ResourceAdminRoles.size** is the size of array defining resource administrator roles.
- **ResourceAdminRoles** are roles of users which are permitted to manipulate with XACML policies. These roles must be same as roles specified in certificates (resource administrator roles).
- **serviceKey** is VOPS's private key used to sign responses.
- entry **policyStorage** points to storage (XML files) which contains user policies and resource policies defining access control to users over these resources.

```
#Properties File for the client application
#Thu Jun 26 13:08:14 CEST 2008
VOPSPubCert=/etc/xos/truststore/certs/vopsserver.pem
testVOPS=true
```

Figure 7.6: A sample ResMng configuration file.

While resource discovery services have to check authenticity of the VOPS's answers, the node running **ResMng** service has to include next lines in its configuration files. List of entries under **ResMng** configuration file:

- **VOPSPubCert** is path to public certificate of the vops server.
- **testVOPS** enables or disables calls to VOPS service.

It is important that if VOPS is to enforce policies over user queries, RCA client must run on resource node which is considered in query. VOPS needs to access RCA client service to obtain resource certificates from which attributes are considered in the query.

Packages:

- **dixi-vom-vops** —The VOPS service provides means to store and manage VO-level policies, to obtain the policy filters and the policy decisions on the VO level.

7.4 Application Execution Management

The Application Execution Management (AEM) is a set of services which ensure that the user can submit requests for jobs, schedules the authorised jobs' execution, and oversees the jobs' lifetime.

The AEM services mostly host in a DIXI framework. In order to install DIXI, please refer to Section 7.2. The following sections explain which services are needed for AEM's proper operation, how to start them and how to configure them.

7.4.1 Core-level AEM services

AEM has a set of services which run in a core mode. This means there should be only one instance of these services running within a domain. A replication of these services is planned for future releases.

The services and the software related to the core-level AEM can be found in the following package:

- **dixi-aem-server**.

The package contains the following services:

- **Job Manager** receives the job requests, schedules the time and nodes for the job execution, and provides the manipulation of the job runtime and information.
- **Job Directory** provides the storage for job information. The service is the front-end to the SRDS, which does the actual storage and retrieval of the service information.
- **Resource Manager** provides the functionality of querying the nodes' local information, and the resource selection. The service is the front-end to the SRDS, which does the actual storage and retrieval of the resource selection.
- **Reservation Manager** is the service which handles the storage and maintenance of the reservations of resources.

Enabling services

Installing the package will enable all the AEM's core services on the same node. Since it is not necessary for all of them to run on the same node, but there currently needs to be just one instance of each service per grid, an administrator may decide to selectively enable or disable the services on the nodes.

The DIXI daemon service checks for presence of stage files in the `/etc/xos/config/xosd_stages` directory. When the package installs, all the required files are already installed and thus the services are enabled by default. The Table 7.2 ennumerates the services, their respective default stage files, and the names of the service handler classes. To disable a service, delete or move elsewhere the service's stage file or open the file with the text editor and modify the relevant line to `enabled=false`. Setting this line back bo `enabled=true` reenables the service. Please note that the xosd service needs to be restarted in order for the changes to take place.

Service name & Default stage file name	**Handler class**
Job Manager JobMng.stage	eu.xtreemos.xosd.jobmng.service.JobMngHandler
Job Directory JobDirectory.stage	eu.xtreemos.xosd.jobDirectory.service.JobDirectoryHandler
Resource Manager ResMng.stage	eu.xtreemos.xosd.resmng.service.ResMngHandler
Reservation Manager ReservationManager.stage	eu.xtreemos.xosd.reservationmanager.service. ReservationManagerHandler

Table 7.1: Enabling the AEM core services using the .stage files.

While some of the services require the SRDS's functionality, it is not necessary that the SRDSMng service is to be installed on the core node(s).

For more information on the service start-up options, please refer to Section 7.2.

Configuring Job Manager

JobMng service uses **JobMng.conf** to read and store its configuration. This configuration file was used in early development stage and at this point it is ignored by the JobMng service. This file consists of the following settings:

- **trustStore** — the path to the public certificates of the trusted signer authorities (e.g., the **cda.crt** of the local organisation's CDA server).

- **scheduling** — selection of the type of the scheduling used. The following values are permitted:
  - 0 — random node selection,
  - 1 — select the less used node,
  - 2 — round-robin selection.

Configuring Resource Manager

ResMng service uses **ResMng.conf** to read and store its configuration. This file consists of the following settings:

- **VOPSPubCert** — the path and the filename of the VOPS's public certificate used for checking the VOPS response's data signatures.
- **testVOPS** — the value should be set to true. For testing purposes, to bypass the policy checking in VOPS, set this to false.
- **useADS** — set whether the ADS should be used as the back-end for the resource selection. If set to false, a central resource selection method will be used (i.e., all within Resource Manager).

7.4.2 Node-level AEM services

The node-level services of AEM represent each node's "physical" services, that check the system's state, execute and work with job's processes, monitor the jobs and keep a reservation time-table. Please note that, for a node to be a fully functional AEM worker node which can receive the jobs in execution, *all* the services from the package need to be installed, configured and running.

The services and the software related to the node-level AEM can be found in the following package:

- **dixi-aem-node**.

This installs the following services:

- **Execution Manager** — forks processes and executes the jobs as requested in the job description.
- **Local resource allocator service** — maintains and provides the functionality to manipulate the local reservation time-table.

- **Resource Monitor** — polls the information on the local node from Ganglia by request.

To run the node-level AEM services which are responsible for providing the resource information of the local node, and executes jobs. They require that the local node's kernel is compiled with connectors enabled, and that the Ganglia monitoring is installed and running on the node. These requirements are reflected in the package's dependencies, which means that the installer will do the needed work.

Further, in order for the node to be selected, and the processes to be successfully run, the node-level AEM requires:

- the XtreemOS PAM module support to be installed and configured,
- the RCA client service (Section 7.3) to be used for obtaining resource's certificates and VO certificates for the node,
- the SRDS Manager service for publishing the node's details and for including it into the overlay (Section 7.6).

Enabling the services

The table 7.2 summarizes the services used by the node-level AEM. Please refer to Section 7.2 for further details.

Service name & Default stage file name	**Handler class**
Execution Manager ExecMng.stage	eu.xtreemos.xosd.execMng.service.ExecMngHandler
Allocator service ResAllocator.stage	eu.xtreemos.xosd.resallocator.service.ResAllocatorHandler
Resource Monitor ResourceMonitor.stage	eu.xtreemos.xosd.resourcemonitor.service.ResourceMonitorHandler

Table 7.2: Enabling the AEM node services using the .stage files.

Enabling kernel connectors

The kernel connectors are already enabled in the kernel installed for the XtreemOS Linux. For custom kernels, however, please refer to Appendix **??** for further instructions.

Enabling services in DIXI daemon's configuration.

To have one or both services start with the local DIXI daemon, place an empty file with proper file name into `/etc/xos/config/xosd_stages`, named after the class that starts the stage. The names are as follows:

- **Resource Monitor**:
  eu.xtreemos.xosd.resourcemonitor.service.ResourceMonitorHandler.stage
- **Execution Manager**:
  eu.xtreemos.xosd.execMng.service.ExecMngHandler.stage

For more information on the service start-up options, please refer to Section 7.2.

Configuring resource monitor.

The ResourceMonitor service uses **ResourceMonitorConfig.conf** to read and store its configuration. The file contains the following settings:

- **monitorType** — the type of external monitor used. The default value *ganglia* sets the Resource Monitor to use Ganglia monitoring system. Alternative value, *xmonitor*, is not explained here.
- **gangliaPort** — the port number that the Ganglia monitoring listens at for the requests. Default port number for Ganglia monitoring is 8649.
- **cpuVals.size** — *xmonitor*-related setting.
- **memVals.size** — *xmonitor*-related setting.
- **cpuVals.0** — *xmonitor*-related setting.
- **memVals.1** — *xmonitor*-related setting.
- **memVals.0** — *xmonitor*-related setting.
- **xMonitorPath** — *xmonitor*-related setting.
- **xMonMemProbe** — *xmonitor*-related setting.
- **xMonCPUProbe** — *xmonitor*-related setting.
- **xMonValName** — *xmonitor*-related setting.

7.4.3 AEM clients

The clients to the AEM services use XATI to access to the services' functionality. please refer to Section 7.2 for more details on XATI.

7.4.4 Job execution preparation

In order for the jobs submitted to the node to be able to successfully run, the PAM extensions and the Account Mapping Service need to be properly configured first [1]. The following checklist covers the necessary steps, performed as a root user.

1. Ensure xos_amsd is running.
   - The `ps -A | grep xos_amsd` command displays whether the daemon is currently running. If it is not, then
   - If this is before the first time running xos_amsd, it can be started by calling `xos_amsd -init`.
   - Otherwise it can be started using `/etc/init.d/xos-amsd start`.
2. Check the PAM extension configuration.
   - Create a folder to contain the trusted CA certificates for the PAM extensions, e.g., `/etc/xos/truststore/certs/pam` and populate it with the root CA certificate and the certificate(s) that sign trusted user certificates. The latter need to be named as their hash, and have to have the .0 extension.
     - `mkdir -p /etc/xos/truststore/certs/pam`
     - `ln -s /etc/xos/truststore/certs/xtreemos.crt \ /etc/xos/truststore/certs/pam/xtreemos.crt`
     - `export CDA_CERT=/etc/xos/truststore/certs/cda.crt`
     - `ln -s $CDA_CERT /etc/xos/truststore/certs/pam/‘openssl x509 -noout -hash -in $CDA_CERT‘.0`
   - In a text editor, open `/etc/xos/nss_pam/pam_xos.conf`,
   - Set `VOCACertDir` to `/etc/xos/truststore/certs/pam/`
   - Set `VOCACertFile` to `xtreemos.crt`
3. Create the mappings for the VO.
   - To learn the VO's globally unique identifier, you can refer to the VOLife (log in, then navigate to Home / Join a VO, the value is the GVID column). You may find it easier examine the XOS-Certificate of a user that belongs to the VO:

  - Assuming the certificate is named `user.crt`, then issue `view-xos-cert user.crt`
  - In the output, the VO's global unique ID is printed after the label "GlobalPrimaryVOName:".
- first create a mapping for the user by calling `xos-policy-admin-am -dn \* -vo $VO_UNIQUE_KEY -locnam \* -drvname root -drvparam 2510`
- then create a mapping for the group by calling `xos-policy-admin-gm -grp \* -vo $VO_UNIQUE_KEY -locgrp \*`

4. Check the ability to execute jobs on the node. This involves using a user XOS-Certificate which includes the proper VO information:
   - `pam_app_conv -pem user.crt`
   - The test succeeds if it shows no "Ooops..." error and puts the prompt in the environment's inner shell.
5. Check or create PAM AEM configuration (/etc/pam.d/aem). Its content should be:

```
#%PAM-1.0
auth sufficient         /lib/security/pam_xos.so
account sufficient      /lib/security/pam_xos.so
session sufficient      /lib/security/pam_xos.so
```

7.5 Resource Selection Service

The RSS (Resource selection service) is a Java service providing a dedicate overlay network (exploiting the Cyclon communication layer) to efficiently locate computing resources based on their static attributes (CPU, memory amount and so on).

RSS interacts with SRDS, the two modules running inside the same JVM. There is one instance of each per computing resource. Beside that, the RSS network needs one additional process to manage nodes who want to join the RSS overlay. This process has to be active on exactly one node (called the bootstrap node) and needs to be known to all nodes in the system.

7.5.1 Install RSS

The RSS module is installed with `urpmi` using the following command:

```
urpmi xtreemrss
```

7.5.2 Configure RSS

The configuration file for RSS is located in `/etc/xos/config/Rss/config.conf`

- **network_interface**: public interface used by the RSS (e.g. eth1)
- **bootstrap_address**: the address of the RSS bootstrap node
- **local_port**: the port number where the local RSS implementation will listen to requests from other nodes
- **bootstrap_port**: the port number used by the RSS bootstrap node

It is possible to configure the log output. The RSS can log its events to the standard output (console) and/or a log file. Logging can be enabled and disabled by commenting out the corresponding lines in the config file.

The config file can optionally define the device used by the RSS to estimate the amount of disk space. By default, the RSS uses the device where the temporary file directory is mounted (typically `/tmp` in Linux).

The config file also defines node attributes used by the RSS. By default, the following five attributes are enabled: CPU architecture, number of CPU cores, amount of RAM, amount of disk space, and operating system. Due to constraints in the current code, the RSS attributes cannot be changed in the config file.

Finally, the config file specifies a number of lower-level RSS parameters. These include the properties of the Cyclon and Vicinity P2P overlays (cache size, gossip period, etc.), TCP connection timeout, randomised delay when bootstrapping nodes (to prevent recorder overflow), and a few parameters for code testing by developers. These parameters do not need to be changed, and the default settings should work well in most environments.

7.6 Scalable Resource Discovery System

The **SRDS** (Scalable Resource Discovery Service) provides several types of directory services to other modules of **XtreemOS**. Different interfaces and set of functionalities are defined for each client.

Current release provides to the **AEM** module both the Resource Location Service and the Job Directory Service (**JDS**). The AEM interface returns a list of computing resources matching a specific resource query (in **JSDL** syntax), and the JDS interface allows to manage job information in a decentralized manner. Further, the SRDS provides an experimental coordinate storing service

for Vivaldi coordinates, with future application to informed machine search in geographic area networks.

Each node within **XtreemOS** has to run an instance of the **SRDS**, all those instances connecting into overlay networks.

Currently, three overlays are created, and they all share the important property that one special node is required to belong to the overlay, which supports all other nodes to initially join the overlay itself. This special node is called "boot" "recorder" or "root" node in the overlay, it is generally required that exactly just one such a node exists in the overlay. The SRDS controlled overlays can in principle have different special nodes, but it is advised for now that the special nodes in the overlay correspond to a single core node.

Sub modules and packages

1. Dynamically changing information is stored into one or more Distributed Hash Tables (**DHT**) by the SRDS; this version of SRDS exploits **Overlay Weaver** and **Scalaris**, the first one is an Open Source Java project the second one is an internal package of Xtreemos. Scalaris is based on Erlang and maintained by Zib. SRDS thus depends on the **Overlay Weaver** and **Scalaris** package. The first DHT is launched in the same JVM that runs SRDS, it is configured by SRDS. The second one is configured by WP that maintains it. It's launched by a script as external process.

2. Static information about computing resources is retrieved through the **RSS** module that is the third overlay (see Section 7.5). The RSS package is also an independent module, but in XtreemOS it is called directly by the **SRDS**, running within the same JVM. Thus **SRDS** and **RSS** depend on each one another.

7.6.1 Install SRDS

If not already pulled by dependencies, the SRDS can be installed via `urpmi` with the following command:

```
urpmi srds
```

This command should install the other packages needed by SRDS: **Overlay Weaver**, **Scalaris** and **Rss**

Disabling overlays In principle, the functionality of disable overlays is provided by the SRDS. However, this can not be done in the 2.0 release of XtreemOS. To disable an overlay the SRDS must be launched by command line and

passing a proper parameter. Within XtreemOS the launch of the SRDS is harcoded into the XOSd, thus making the parameter passing not possible without changes into the code.

Note that disabling an overlay may causes some services to stop working. For instance, the Vivaldi coordinate service depends on the Scalaris module, while the resource selection service strongly depends on the RSS.

Configure SRDS

Configuring SRDS and its dependencies (RSS and ADS_OverlayWeaver and ADS_Scalaris) means to set up in a few configuration files all the proper IP addresses and port numbers. In particular, the files defining the configuration are:

Filename	Description
/etc/xos/config/Ads/srds.properties	*ADS/SRDS general properties and common DHT properties*
/etc/xos/config/Rss/config.cfg	*RSS related properties*
/etc/xos/config/OW/OWconfig.cfg	*Overlay Weaver related properties*
/etc/scalaris/scalaris.properties	*Information about Scalaris instance*
/etc/scalaris/scalaris.cfg	*for the Scalaris boot node*
/etc/scalaris/scalaris.local.cfg	*for other nodes*
/usr/bin/scalarisctl	*script to launch/control nodes*

As explained, each DHT/overlay must have a boostrap node that allows new nodes to join the overlay. In general for each DHT in the VO-frame there must exist a bootstrap node. The bootstrap nodes may be different for each DHT, but for ease of maintenance we prefer to use the same bootstrap node for each DHT.

In Rss the Boot node will start the Recorder daemon.
In Overlay Weaver the boostrap node is any node used for joining (each node has a port for listen incoming join messages).
In Scalaris the are 2 kind of nodes, boot and regular ones. Any overlay must have at least one boot node.

After configuring the overlays (see below), check that the SRDS services are enabled within the DIXI daemon (xosd). This is normally taken care by the XtreemOS configuration tools. The Xosd will start up SRDS if the `/etc/xos/config/xosd_stages` directory contains a file named `SRDSMng.stage`. The file contents should be:

```
service=eu.xtreemos.ads.connection.dixi.service.SRDSMngHandler
enabled=true
```

Note that any changes to the configuration files will be applied next time the Xosd service is started.

rest of this section to be rearranged

ADS Configuration

- File `/etc/xos/config/Ads/srds.properties`
  This part describes the mapping between JSDL dynamic attributes and the real names used by ADS. To get informations from the system (/proc) ADS uses a module called AemInformationProvider, the tags listed below describe which particular JSDL tags we want to filter.

  srds.configurations.dyntags.attributesnumber : indicates how many attributes we want filter, default value is 5

  srds.configurations.dyntags.1 : indicates the relationship between the JSDL tag IndividualPhysicalMemory and the name used by ADS for the attribute that indicates the dynamic RAM Memory available, it takes the mandatory value *IndividualPhysicalMemory;RamAvailable*

  srds.configurations.dyntags.2 : indicates the relationship between the JSDL tag IndividualVirtualMemory and the name used by ADS for the attribute that indicates the dynamic SWAP Memory available, it takes the mandatory value *IndividualVirtualMemory;SwapAvailable*

  srds.configurations.dyntags.3 : indicates the relationship between the JSDL tag DiskSpace and the name used by ADS for the attribute that indicates the dynamic Disk space available, it takes the mandatory value *DiskSpace;DiskAvailable*

  srds.configurations.dyntags.4 : indicates the relationship between the JSDL tag IndividualNetworkBandwith and the name used by ADS for the attribute that indicates the dynamic Bandwith available, it takes the mandatory value *IndividualNetworkBandwith;BandAvailable*

  srds.configurations.dyntags.5 : indicates the relationship between the JSDL tag IdlePercentage and the name used by ADS for the attribute that indicates the dynamic Idle CPU available, it takes the mandatory value *IdlePercentage;IdlePercentage*

  srds.configurations.dyntags.6 : indicates the time from last restart of the system, it takes the mandatory value *Uptime;Uptime.* However this attributes should be commented out, because is not really tested at the moment.

  This part describes general configurations for ADS.

srds.configuration.files.glueSchema : specifies the absolute path of the schema file used to validate GLUE. It should take */usr/share/dixi/XMLExtractor/Schemas/GLUESchema12R2.xsd*

srds.configuration.device.disk : identifies the disk used for retrieve dynamic informations, it takes as default value */tmp*

srds.configuration.device.ethernet : identifies the ethernet interface used to retrieve dynamic info about connection (should be the same specified into */etc/xos/config/Rss/config.cfg*) and it takes as default value *eth0*

srds.configurations.configPath : identifies the xos configuration path, must take */etc/xos/config/*

srds.configurations.webserverPort : identifies the port of the SRDS web server, by default it is 9000

RSS Configuration

- File `/etc/xos/config/Rss/config.cfg`

network_interface : the network interface to use (likely *eth0*, use *lo* for localhost testing)

disk_device : the disk device to monitor for free space; currently, physical disks (e.g. */dev/sda2*) and logical volumes (e.g. */tmp*, where */tmp* was the mountpoint of a volume)

bootstrap_address : the bootstrap node for the RSS overlay (the IP address of the chosen as Core Node to act as a bootstrap; when doing localhost testing, use 127.0.0.1)

DHTs Common Configuration

- File `/etc/xos/config/Ads/srds.properties`
The parameters listed below can be changed for both Scalaris and Overaly Weaver. The default setup is fine for most of the cases.

srds.configurations.bootstraping.dht1.type : identifies index for disambiguate various DHTs

srds.configurations.bootstraping.dht1.name : description or name

srds.configurations.bootstraping.dht1.keylengthinbit : length of the key used inside the DHT (Usually 128 or 160 bits)

srds.configurations.bootstraping.dht1.replicationDegreeMax : defines how much each resource should be replicated.

srds.configurations.bootstraping.dht1.isTransactional : Enable the transactional properties. The default is true for scalaris, and false for Overaly Weaver (OW does not support transactional operations yet).

Overlay Weaver Configuration

- File `/etc/xos/config/Ads/srds.properties`

  srds.configurations.scripts.ow.type : identifies the DHT type, set the value to 2

  srds.configurations.scripts.ow.command.start : no longer used in XtreemOS

  srds.configurations.scripts.ow.configurationFile : identifies the configuration file read in by Overlay weaver, set it to *OW/OWconfig.cfg*

  srds.configurations.scripts.ow.webserverPort : where to provide the OW monitoring web server. It should take the same port of dht.xmlrpcPort, which by default is 3998

- File `/etc/xos/config/OW/OWconfig.cfg`

  dht.RoutingAlgorithm : specifies routing algorithm (Possible values are 'Chord', 'Kademlia', 'Koorde', 'LinearWalker', 'Tapestry' or 'Pastry'). We use **Chord**.

  dht.TransportProtocol : specifies transport protocol (TCP or UDP) (We use TCP as default)

  dht.WorkingDir : the working directory of OW (by default '.')

  dht.UPnPNAT : Enable the NAT traversal with device (router) that support UPnP protocol. (It is 'false' by default)

  dht.BootstrapPort : specifies the port on which any node listen for the incoming connections (we use the default value 3997)

  dht.xmlrpcPort : specifies the port of the xml rpc server. (no longer used, the default value is 3998)

  dht.BootstrapHost : specifies the numerical address of the boostrap host; if equal to the local address, this node will act as a bootstrap node.

Note on Bootstrapping in Overlay Weaver If the bootstrap node fails, a problem arises in the OW bootstrapping strategy as is it now. When the bootstrap node returns back, it try to connect to itself thus creating another overlay that is detached from the main one. A workaround for this issues goes as following. Select **two** bootstrap node, for instance the node A and B. In the configuration

of node A, the bootstrap node should be B, while B has as bootstrap the node A in turn. With this configuration, whenever a bootstrap node returns back, it should find the other node up and so it is able to join the correct overlay. For the other nodes (i.e. not bootstrap) either A or B is a good option. Note that since A and B may fail at the same time, this is not meant to be a definitive solution. A more robust bootstrap mechanism will be present in a future version of XtreemOS.

Scalaris Configuration

The following configurations must be thoroughly checked against scalaris configurations.

The Scalaris DHT is configured in different ways depending if a node is bootstrap (BOOT) or not (REGULAR). Both in BOOT and REGULAR nodes make sure that in the file **/usr/bin/scalarisctl** the value for **SCALARISDIR** parameter to be */etc/scalaris*.

The paragraphs below describe the rest of configuration procedures for BOOT and REGULAR nodes.

WARNING: Every node in the Scalaris overlay must be configured with a Fully Qualified Host Name (FQHN, this means that the file /etc/hosts must contain the machine name AND the relative domain, i.e. machinename.somedomain.org).

Scalaris BOOT node (the boot node must be a **Core** node)

- File `/etc/scalaris/scalaris.properties`

  scalaris.node : should take *boot@localhost*

  scalaris.cookie : cookies used by Scalaris for connection (by default take value *chocolate chip cookie*)

  scalaris.client.name : specifies the client name (take value *java_client* by default)

  scalaris.client.appendUUID : specifies whether to append an UUID to client names or not (by default take value *true*)

- File `/etc/xos/config/Ads/srds.properties`

  srds.configuration.bootstraping.isScalarisBootNode : identifies if the node is a REGULAR or BOOT node**true**

  srds.configurations.scripts.scalaris.type : identifies the DHT type in this case the value is 1

srds.configurations.scripts.scalaris.command.start : */usr/bin/scalarisctl*

srds.configurations.scripts.scalaris.command.stop : read but not used in current version.

srds.configurations.scripts.scalaris.configurationFile : */etc/scalaris/scalaris.properties*

srds.configurations.scripts.scalaris.webserverPort : identifies the port of the web server of Scalaris, by default 9001

- File `/etc/scalaris/scalaris.cfg`

  the row {`boot_host`,{{`146,48,82,99`},`14195`,`boot`}} should take the numerical public IP of the machine.

Scalaris REGULAR node (a regular node can be a **Resource** or a **Core** node of Xtreemos)

- File `/etc/scalaris/scalaris.properties`

  scalaris.node : should take *node@localhost*

  scalaris.cookie : cookies used by Scalaris for connection (by default take value *chocolate chip cookie*)

  scalaris.client.name : specifies the client name (take value *java_client* by default)

  scalaris.client.appendUUID : specifies whether to append an UUID to client names or not (by default take value *true*)

- File `/etc/xos/config/Ads/srds.properties`

  srds.configuration.bootstraping.isScalarisBootNode : **false**

  srds.configuration.bootstraping.scalarisConfFile : */etc/scalaris/scalaris.properties*

  srds.configurations.scripts.scalaris.type : 1

  srds.configurations.scripts.scalaris.command.start : */usr/bin/scalarisctl*

  srds.configurations.scripts.scalaris.command.stop : currently not used

- File `/etc/scalaris/scalaris.local.cfg`

  – the row {`boot_host`,{{`146,48,82,99`},`14195`,`boot`}} should take the numerical public ip of the boot machine..

Running and Monitoring SRDS

The SRDS is launched by the xosd service. The bootstrap process of the overlays is managed by SRDS, no external processes are currently needed. Scalaris is an exception with this respect, it is launched separately and the SRDS connects to the DHT.

Once the SRDS is set up and running, you can monitor the controlled Overlay network through a Web interface. It shows a simple SRDS web console that reports basic information about the SRDS.

To try it by yourself please type target_hostname:9000 in your browser and you should be prompted with something similar to figure 7.7.

Currently there are 3 sections available. The **Overlay Weaver monitor** section gives details about the node OW status, as the numbers of keys stored, which algorithm and which lookup style OW is using. There are also information about topology (finger table, successor list).

The **Xosd Logger** frame allows to see the running log of XOSd on the machine and filter out message lines based on their contents, to ease activities of real-time machine check and system debugging.

The **Scalaris monitor** give access to a nuber of datail about the state of DHT like the number of nodes and the web-server log.

Also, both DHT web interfaces allows the user to submits basic operation like put and get

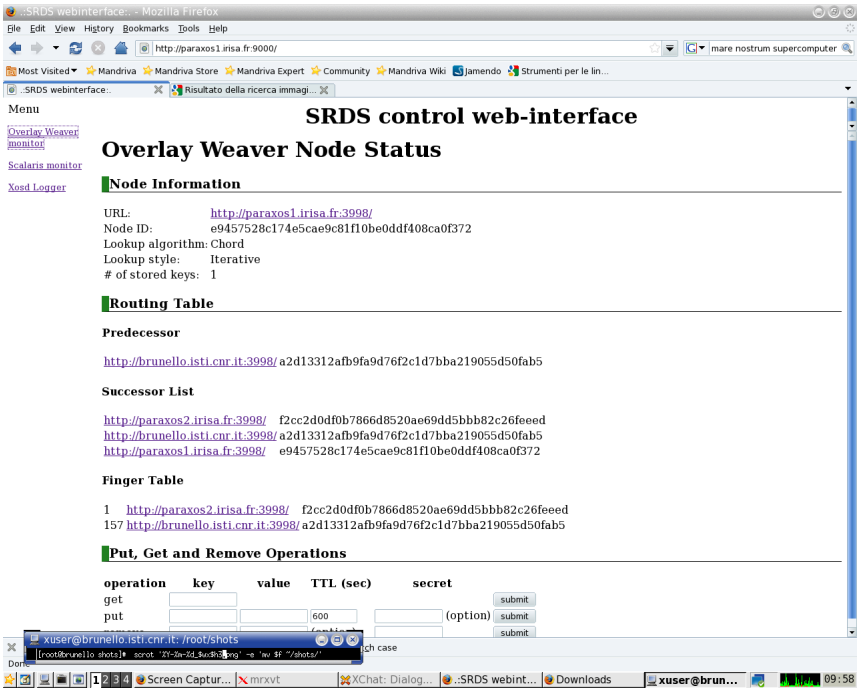

Figure 7.7: SRDS Web Interface: Overlay Weaver status

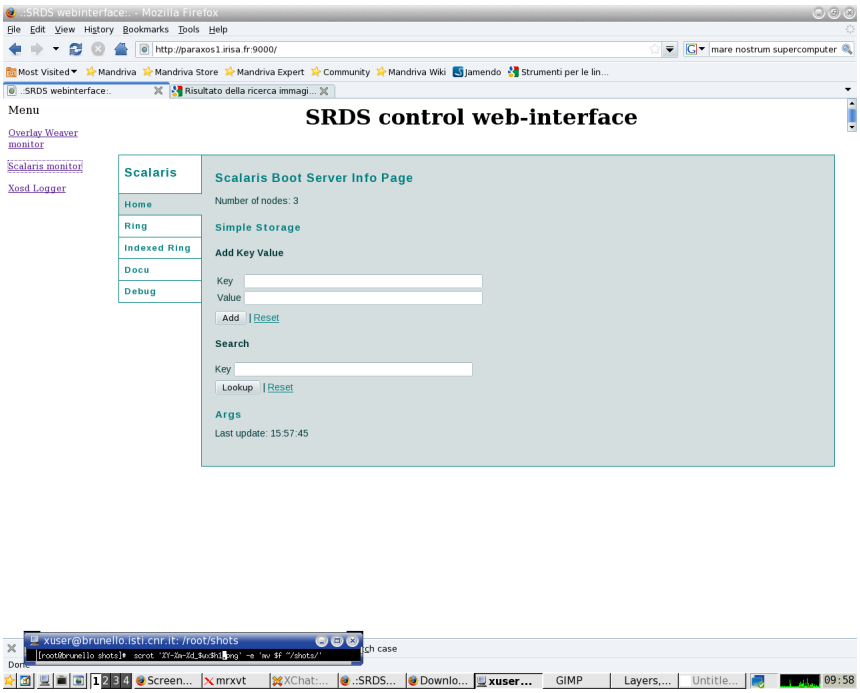

Figure 7.8: SRDS Web Interface: Scalaris main page

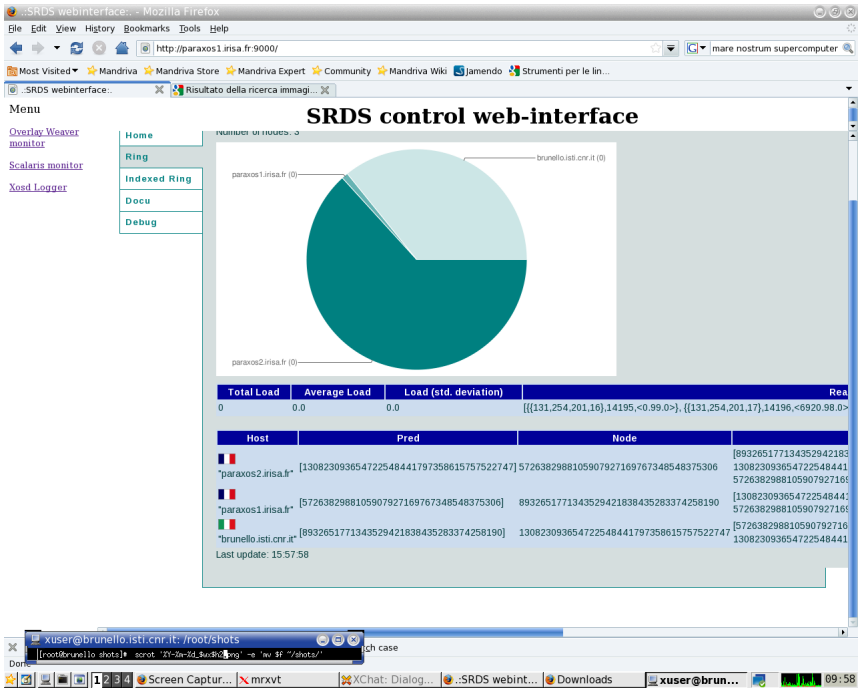

Figure 7.9: SRDS Web Interface Scalaris: ring status and nodes list

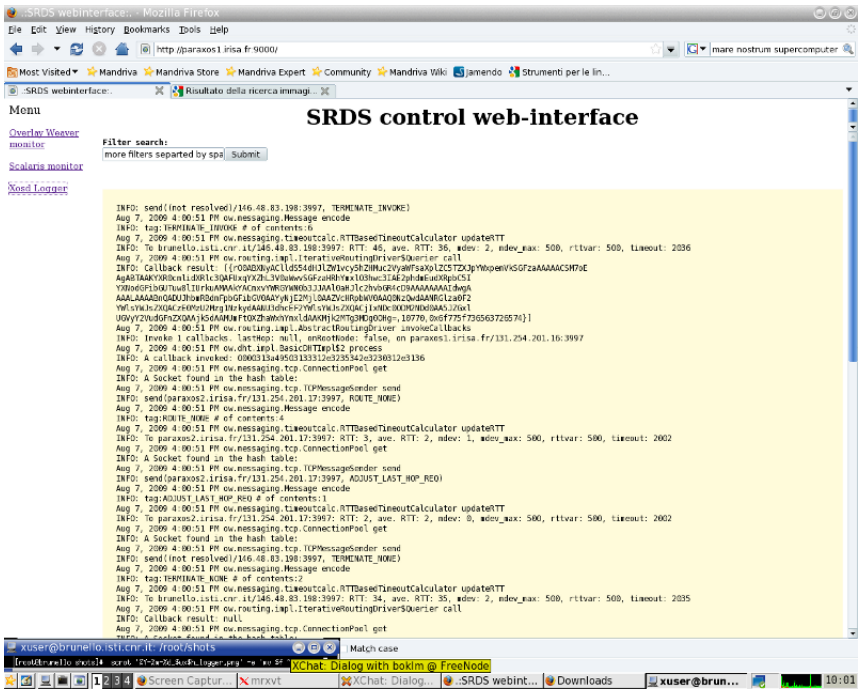

Figure 7.10: SRDS Web Interface: XOSD logger

7.7 XtreemFS

XtreemFS is the distributed file system of XtreemOS. It comprises three server modules:

Metadata and Replica Catalog (MRC)	MRCs are responsible for the file system metadata.
Object Storage Device (OSD)	OSDs store the content of files.
Directory Service (DIR)	The Directory Service acts as a global repository for information about OSDs, MRCs and file system volumes of an XtreemFS installation.

The file system is accessed by means of a client module:

Client/Access Layer (AL)	The Access Layer allows user processes to work with XtreemFS, via a POSIX interface. It is implemented as a user space module based on FUSE.

7.7.1 XtreemFS Installation

There are two different installation sources for XtreemFS: *RPM packages* and *source tarballs*.

Separate RPM packages exist for the server and the client components. To install the XtreemFS server components, execute

```
$> urpmi xtreemfs-server
```

This will install an `init.d` script to set up the server components.

For the client components, execute

```
$> urpmi xtreemfs-client
```

This will install the XtreemFS client to `/usr/bin/xtreemfs`

If you prefer using the source tarball, the first step is to build the components. The following third-party modules are required:

- Java Development Kit 1.6
- Apache Ant 1.6.5
- python 2.4
- gmake 3.81

- gcc-c++ 4.3
- FUSE 2.6
- openssl-dev 0.9.8

To build the server components, make sure that `JAVA_HOME` and `ANT_HOME` are set. `JAVA_HOME` has to point to a JDK 1.6 installation, and `ANT_HOME` has to point to an Ant 1.6.5 installation.

Go to the top level directory and execute:

```
$> make
```

Once finished, binaries and shell scripts can be used to run XtreemFS.

7.7.2 XtreemFS Security Preparations

In order to provide for security in an XtreemFS installation, services need to be equipped with X.509 certificates. Certificates are used to establish a mutual trust relationship between XtreemFS services, as well as between the XtreemFS client and XtreemFS services. Thus, in an XtreemOS environment, certificates have to be created for the services as a first step. This is done by creating a *Certificate Signing Request (CSR)* for the Root CA by means of the `create-csr` command. For further details, see Sec. 7.3.5.

Signed certificates and keys generated by the Root CA are stored locally in PEM format. Since XtreemFS services are currently not capable of processing PEM format, 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/certs/xtreemos.crt
```

This will create a new Java Keystore `xosrootca.jks` with the XtreemOS CA certificate 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 details on XtreemFS security setup, please see The XtreemFS Installation and User Guide [2], Sections 2.2.2, 3.2.2, 3.2.7 and Appendix A.

7.7.3 XtreemFS Setup and Configuration

An XtreemFS installation requires at least one Directory Service, OSD and MRC to be running. In order to render XtreemFS accessible to user processes, a volume needs to be mounted via the Access Layer.

Default Setup.

If installed from the RPM packages, default configuration files have already been created for the services. They are located in `/etc/xos/xtreemfs`. Three such configuration files exist for the different XtreemFS services: `dirconfig.properties` to configure a Directory Service, `mrcconfig.properties` to configure an MRC, and `osdconfig.properties` to configure an OSD. These are self-documenting Java properties files. If root access rights are granted, configuration parameters can be modified with an arbitrary text editor.

The following predefined ports are used for the services:

Directory Service	32638
MRC	32636
OSD	32640

Important: To guarantee that OSDs can be contacted by clients, it is necessary to ensure that OSDs register with their correct communication endpoints. XtreemFS will try to determine such endpoints automatically, but this sometimes leads to wrong results. In order to ensure that OSDs are configured correctly, **it is necessary to manually set the** `listen.address` **property**.

```
# optional address for network device, "any" if not specified
# listen.address = mydomain
```

`mydomain` has to refer to either to an IP address, or to a fully-qualified distinguished name that can be resolved to an IP address. Defining the property ensures that `mydomain` will be used as the endpoint at which the OSD is reachable.

The default configuration for the MRC and OSD assumes that the Directory Service runs on the local machine. The endpoint of the Directory Service to register at can be changed by modifying the following properties:

```
# Directory Service endpoint
dir_service.host = localhost
dir_service.port = 32638
```

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 (see Sec. 7.7.2). Make sure that all paths and pass phrases (`xtreemfs` in this example) are correct.

In order to start and stop the services, three different shell scripts exist in `/etc/init.d`. All services can be started on the local machine by executing the

following commands as root:

```
# /etc/init.d/xtreemfs-dir start
# /etc/init.d/xtreemfs-mrc start
# /etc/init.d/xtreemfs-osd start
```

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.

The services can be stopped by executing the following commands as root:

```
# /etc/init.d/xtreemfs-dir stop
# /etc/init.d/xtreemfs-mrc stop
# /etc/init.d/xtreemfs-osd stop
```

7.7.4 Upgrading XtreemFS

Each new version, uses an upgraded protocol version and a different on-disk database format for the directory service (DIR) and metadata server (MRC). You can safely delete the DIR database, all servers will re-register on start up. The DIR database is ususally stored in /var/lib/xtreemfs/dir/. The MRC database must be dumped before installing the new release!

- Use

  ```
  xtfs_mrcdbtool -mrc oncrpc://mrchost:32636 dump /path/to/dumpfile
  ```

  to save the database to an XML file.
- Check that the file exists on the MRC host and is not empty!
- Shut down the MRC

  ```
  /etc/init.d/xtreemfs-mrc stop
  ```

  (and the DIR and OSD if they are still running)
- Remove the MRC database /var/lib/xtreemfs/mrc/
- Install the new 1.x release of XtreemFS
- Sart the MRC /etc/init.d/xtreemfs-mrc start
- Use

  ```
  xtfs_mrcdbtool -mrc oncrpc://mrchost:32636 restore /path/to/dumpfile
  ```

  to restore the database from the XML file.

7.8 XOSAGA

XOSAGA is available in XtreemOS as several rpm packages:

libsaga-devel contains everything to develop SAGA applications.

libsaga contains all libraries to run SAGA applications.

saga contains some example SAGA programs and environment settings.

xosaga contains XtreemOS-specific additions to SAGA.

Installing XOSAGA can be done using urpmi:

```
$> urpmi xosaga
```

You will be given a choice between the 'libsaga' and the 'libsaga-devel' package. Choose the 'libsaga' package if you only want to *run* SAGA applications (e.g. on an XtreemOS node). Choose the 'libsaga-devel' package if you also want to develop XOSAGA applications on your machine.

7.9 LinuxSSI

A LinuxSSI cluster can act as a resource in XtreemOS. Please, do not use it as a server node. Beware that all your cluster nodes must be on the same physical network. There must be no router between them.

The following steps have to be done on each cluster's node:

1. Install XtreemOS as described in section 3. Do not forget to select LinuxSSI.
2. Configure the boot-loader to have different *node id* for each node of your cluster. Node id are numerical numbers between 1 and 254. To set the *node id*, you need to add on the kernel command line `node_id=X`, with *X* the *node id*. This configuration can be done during the installation process or after by modifying file `/boot/grub/menu.lst`.

   You must also check that the LinuxSSI kernel is the default kernel to boot.
3. Once all nodes have been installed, check the DNS configuration or edit `/etc/hosts` to be able to contact each cluster's node from any cluster's node.
4. You need to create file `/etc/xos/linuxssi/nodes` and to fill it with the hostname of each cluster's node with one node per line. The first node will act as the *head* node.

```
root# emacs /etc/xos/linuxssi/nodes
clusternode1
clusternode2
clusternode3
...

Quit emacs.
```

5. Configure ssh so that you can connect from head node to other nodes as root.

The following steps have to be done **only on the *head* node**:

1. Install package kanif

```
root# urpmi kanif
```

2. Install package nfs-utils

```
root# urpmi nfs-utils
```

It is now time to have all your cluster node booted on LinuxSSI kernel.

Connect as root to the head node and start LinuxSSI:

```
root# start_linuxssi
```

This command will

- configure some NFS exports (`/etc/xos`, `/home`, `/root`, `/tmp`, `/usr/local`, `/var`),
- start LinuxSSI (`krgadm cluster start`),
- start the AEM-LinuxSSI listener,
- switch the head node to runlevel 4,
- make other LinuxSSI cluster's nodes run `start_linuxssi_as_worker`. `start_linuxssi_as_w` will
  - switch the node to runlevel 3,
  - mount the NFS sharing,

- start the AEM-LinuxSSI-dispatcher.

You can check that LinuxSSI is running by invoking `cat /proc/cpuinfo` that must show information about the CPU of all your cluster nodes.

Then, you can run the XtreemOS resource configuration on the *head* node as if it is a single node, as explained in section 6.

Once you have finished, and *xosd* is running, run:

```
root# start_linuxssi_scheduler.sh
root# xosd_linuxssi_capabilities
```

The first command will start the legacy LinuxSSI scheduler. You can configure an personalized one if you prefer (see Section **??**). The second command will ensure that Xosd has and will give good LinuxSSI capabilities for jobs to be distributed and migrated on the cluster.

Bibliography

[1] XtreemOS consortium. Prototype of the basic version of linux-xos. Deliverable D2.1.4, November 2007.

[2] Björn Kolbeck, Jan Stender, and Felix Hupfeld. The XtreemFS Installation and User Guide. http://www.xtreemfs.org/xtfs-guide.pdf.

Chapter 8

Public Resources on the Net

This appendix lists XtreemOS resources available on the Internet, for the common user, the System administrator and the developer.

8.1 Resources for Users

Table 8.1: User Resources

Description	URL / contact info
Official WWW	http://www.xtreeemos.eu
XtreemOS Blog	https://www.xtreemos.org/blog
XtreemOS Mobile Guide	ftp://ftp.free.fr/mirrors/ftp.mandriva.com/MandrivaLinux/devel/xtreemos/2008.0/mobile/xtreemosmd-guide.pdf
IRC channel for user support	irc.freenode.netchannel #xtreemos

8.2 Code and Repositories

we should list here : the main Mandriva and Red Flag repositories for packages, Install cds, and the repositories where the Virtual Box /VMWare images are found;

8.3 Resources for Developers

important deliverables (or parts of them), references to some tech papers, tech tutorials?

Table 8.2: Public Repositories

Description	URL / contact info
Mirror for CD Images	ftp://ftp.free.fr/mirrors/ftp.mandriva.com/MandrivaLinux/devel/iso/xtreemos/
XtreemOS Mobile distribution	ftp://ftp.free.fr/mirrors/ftp.mandriva.com/MandrivaLinux/devel/xtreemos/2008.0/mobile/
TikiWiki site	https://xtreemos.wiki.irisa.fr/
XtreemOS Development site	https://gforge.inria.fr/projects/xtreemos/
Public SVN	http://gforge.inria.fr/plugins/scmsvn/viewcvs.php/?root=xtreemos
Architecture documents	http://www.xtreemos.eu/publications/project-deliverables/

Table 8.3: Developer Resources

Description	URL / contact info
IRC channel for user support	irc.freenode.netchannel #xtreemos-dev
TikiWiki site	https://xtreemos.wiki.irisa.fr/
Developer mailing list	http://lists.gforge.inria.fr/cgi-bin/mailman/listinfo/xtreemos-developers
XtreemOS Development site	https://gforge.inria.fr/projects/xtreemos/
Public SVN	http://gforge.inria.fr/plugins/scmsvn/viewcvs.php/?root=xtreemos
Architecture documents	http://www.xtreemos.eu/publications/project-deliverables/

Chapter 9

Glossary

report acronyms and terms for the end-user

Index

This index right now is a tool to help editing; we shall consider producing real indexes, especially in the admin guide. This note appears on the wrong page