IP-FP6-015964

AEOLUS

Algorithmic Principles for Building Efficient Overlay Computers

Deliverable D6.4.2

Demo software for application scenario: Preliminary release

Responsible Partner: Radiolabs (I)
Report Preparation Date: September 2008

Contract Start Date: 01/09/05 Duration: 48 months
Project Co-ordinator: University of Patras (EL)

Table of contents

Deliverable D6.4.2 concerns the preliminary version of the demo application AEOFORGE whose specification and design is provided in Deliverable D6.4.1. A demonstration will take place during the review meeting.

1 Introduction

AEOFORGE is a prototype of a „peer-to-peer sourceforge“: it allows users to share, control and manage software development.
AEOFORGE provides its users with a simple command line interface to access a distributed file storage where software projects are hosted to be further retrieved or edited.
The protoype demo software is released as a zip file named

aeoforge-runtime.zip

which is available along with a slide presentation and instructions for running it at http://aeolus.ceid.upatras.gr/files/AEOLUS-DEMO-CD.rar. The file is an extended AEOLUS shell including the AEOFORGE functionalities; each component of the AEOFORGE architecture described in D6.4.1 is an instance of the AEOLUS shell.
Two additional jar files are included to patch JRE as described below.
The demo will show how to upload a project tarball file, download it, edit a project's file and store it back in the forge avoiding concurrent modifications.

2 AEOFORGE demo installation

To install the software to run the demo we must start a shell for each of the component of the AEOFORGE architecture:

- the Secure Timestamping Authority (STA)
  - used by the AEOFORGE service to bind, to each file of a project stored in the forge, a secure timestamp to check for concurrent modifications.
- the file storage coordinator
- the storage offering peer
- the AEOFORGE service peer
- the AEOFORGE client

We can achieve this unzipping *aeoforge-runtime.zip* file in a different folder for each shell/peer we need (i.e. 5). We can start all the shells on the same machine for demo purposes.

2.1 Java Environment setup

Prerequisites:

- JDK 1.6 installed
- JAVA_HOME environment variable defined

2.2 STA setup

STA setup needs Java Virtual Machine patching:

- edit **$JAVA_HOME/jre/lib/java.security** file adding the following line:

security.provider.#=org.bouncycastle.jce.provider.BouncyCastleProvider

- # value depends on the actual size of the security provider list
- replace in **$JAVA_HOME/jre/lib/security** the files *local_policy.jar* and *US_export_policy.jar* with the provided corresponding files

The JVM is now patched.

To start the STA services execute the following steps:

- start a JXTA shell (the STA peer) executing the **run** script (run.bat or run.sh depending on the machine OS) located in **shell-2.5** folder of the peer
- generate keys and certificates for the authority issuing the shell command

JXTA> stakeytool -cn sta -keystore staks.p12 -storepass security -alias sta -aliaspass security -keylength 1024

- file **staks.p12** is generated
- start the public registry issuing the shell command

JXTA> star -keystore staks.p12 -storepass security

- start the authority issuing the shell command

JXTA> sta -n sta1 -keystore staks.p12 -storepass security -alias sta1 -aliaspass security

2.3 Coordinator setup

Follow these steps to start the file storage coordinator:

- start a JXTA shell (the coordinator peer) executing the **run** script (run.bat or run.sh depending on the machine OS) located in **shell-2.5** folder of the peer
- issue the shell command

JXTA> fs.coordinator

2.4 Storage peer setup

The storage peer is started via the shell command

JXTA>fs.my_offer <storage units offered> <storage duration>

So, to setup the storage peer offering 100k storage units for 2 years we must

- start a JXTA shell (the storage peer) executing the **run** script (run.bat or run.sh depending on the machine OS) located in **shell-2.5** folder of the peer
- issue the shell command

JXTA> fs.my_offer 100000 63072000000

2.5 AEOFORGE service setup

To start the AEOFORGE service (the server side component of AEOFORGE) we must

- start a JXTA shell (the AEOFORGE service peer) executing the **run** script (run.bat or run.sh depending on the machine OS) located in **shell-2.5** folder of the peer
- issue the shell command

JXTA> aeoforge.service

The service shell logs all the activities of the AEOFORGE service.

2.6 AEOFORGE client setup

To start using the AEOFORGE we must start a dedicated JXTA shell in the same way we started all the other shells. The AEOFORGE commands are issued from within this shell. See D6.4.1 for detailed description of AEOFORGE APIs.

3 How to run AEOFORGE demo

3.1 Upload a tarball file

A project is uploaded on the forge starting from a zip file located in the user's filesystem. The project (i.e. the zip file) is uploaded issuing the shell command

JXTA> aeoforge.upload <filename> <project name>

The file is uploaded and the file is unzipped: the files contained in the tarball are stored in the storage peer preserving the project's folder structure. The following command shows the actual folder structure:

JXTA> aeoforge.describe <project name>

3.2 Download a project

To download a zip file containing all the project's files on our filesystem we type:

JXTA> aeoforge.download <project name>

After the we download and unzip the tarball, we can start editing the project's file.

3.3 File editing

First of all we must check out the file we wishe to edit:

JXTA> aeoforge.co <project name> <filepath>

After having edited the file we store it back in the forge:

JXTA> aeoforge.ci <project name> <filepath>

To add a new file to the project we issue the command:

JXTA> aeoforge.add <project name> <filepath>