



IST-2001-32133

GridLab - A Grid Application Toolkit and Testbed

## Implementation and Test Plan for the AS

---

Author(s):	Marcin Adamski, Michal Chmielewski, Sergiusz Fonrobert, Jarek Nabrzyski, Tomasz Nowocień, Tomasz Ostwald
Document Filename:	
Work package:	WP6 Security
Partner(s):	PSNC, MPG, MU, ISUFI
Lead Partner:	Poznan Supercomputing and Networking Center
Config ID:	GridLab-6-D6.3-0004-0.9
Document classification:	INTERNAL

---

**Abstract:** In this document, a description of implementation and tests of the Authorization Service developed as a part of WP6 of the GridLab project is presented. The document contains a general overview of the first prototype, development plan, description of the next tests and a general plan of integration with other Workpackages



## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Purpose of Document . . . . .	2
1.2	Structure of Document . . . . .	2
1.3	Status of Document . . . . .	2
<b>2</b>	<b>Implementation and Test Plan</b>	<b>3</b>
<b>3</b>	<b>The first prototype</b>	<b>4</b>
3.1	Included functionality and possibilities . . . . .	4
3.2	Technical specification . . . . .	5
3.3	Components . . . . .	5
3.4	The data structure . . . . .	7
<b>4</b>	<b>The integration with other workpackages</b>	<b>9</b>
<b>5</b>	<b>The extensions of the AS</b>	<b>10</b>
5.1	Storing security policy into the SQL database . . . . .	10
5.2	Extending scenarios functionality of the AS . . . . .	10
5.3	Extending method of generating security policy . . . . .	10
5.4	GUI as_admin_client . . . . .	10
5.5	Possible integrations with other security solutions . . . . .	10
<b>6</b>	<b>The overview of future tests</b>	<b>12</b>
<b>7</b>	<b>Summary</b>	<b>13</b>

## **1 Introduction**

### **1.1 Purpose of Document**

This document contains an implementation and tests plan of the Authorization Service and a description of the initial version of the first AS prototype, which will be developed in the Security Workpackage (WP6) of the GridLab project.

### **1.2 Structure of Document**

The document is divided into 7 general sections. In section 1, a general document structure and its goals are presented. Section 2 describes the implementation and test plan of the Authorization Service. Section 3 is dedicated to a description of the first prototype. In section 4, a possibility and proposition of the next integration with other workpackages is presented. Section 5 discusses the extension of the AS after the first prototype. In section 6 there is a description of the next prototype tests. The summary and final notes are given in section 7.

### **1.3 Status of Document**

This document is a working draft and refers to the current state of the project. As a working draft, this specification may be updated, replaced, or made obsolete at any time. It is distributed for discussion purposes only, and should not be used as a reference.

## 2 Implementation and Test Plan

The general implementation and test plan for the Authorization Service is presented below.

No.	Item	End time
1	The initial implementation of the AS contains: <code>as_server</code> main parts of core, <code>as_admin_client</code> , <code>as_client</code> and sample usage ( <code>as_enabled server, client</code> ).	January 2003
2	Preparing and testing the first prototype of the AS. Integration with GridLab Resource Management System (GRMS) (WP9). Integration with Portal (WP4). Testing the cooperation between <code>as_server</code> and GRMS.	March 2003
3	Integration with others workpackages. Testing the cooperation between <code>as_server</code> and other services.	August 2003
4	Documented first release of the Authorization Service and APIs (D6.4).	December 2003
5	The extended version of the AS service contains: - extending method of generating security policy, - adding scenarios functionality to <code>as_server</code> , - possible integrations with other security solutions, - GUI <code>as_admin_client</code> , - storing security policy into the SQL database.	October 2004
6	Last tests and improvements. Documented second release of the Authorization Service and APIs (D6.5).	December 2004

The terms presented in the table above can change in the future. It is possible that some implementation points will be changed, too. Some tests are described more accurately in chapter 6 and a sample plan of integration is described in chapter 4. It is not possible to make an accurate prediction of some parts of future tests and implementation, so these points will be extended in the future.

### 3 The first prototype

General requirements for Security Workpackage 6 in the GridLab project have been presented in the first deliverable, *WP6 Security: Initial Requirements* [1]. In the following deliverable *Technical Specification for Authorization Service* [2] the architecture and building of the AS is described. On the basis of these two deliverables, the first prototype of the Authorization Service will be developed.

#### 3.1 Included functionality and possibilities

With this prototype it will be possible to:

- create a structure for security policy based on services requirements;
- add, delete or edit security policy stored on the AS server by a special admin client;
- generate the authorization decision for user or service request (in this mode the AS is in the Simple Super-Scheduler Scenario [2]);
- generate full security policy for a user or a service (in this mode the AS is in the CAS scenario [2]) and is compatible with the CAS (Community Authorization Service); the generated policy can be used with the CAS enabled components (like gsift) [3].

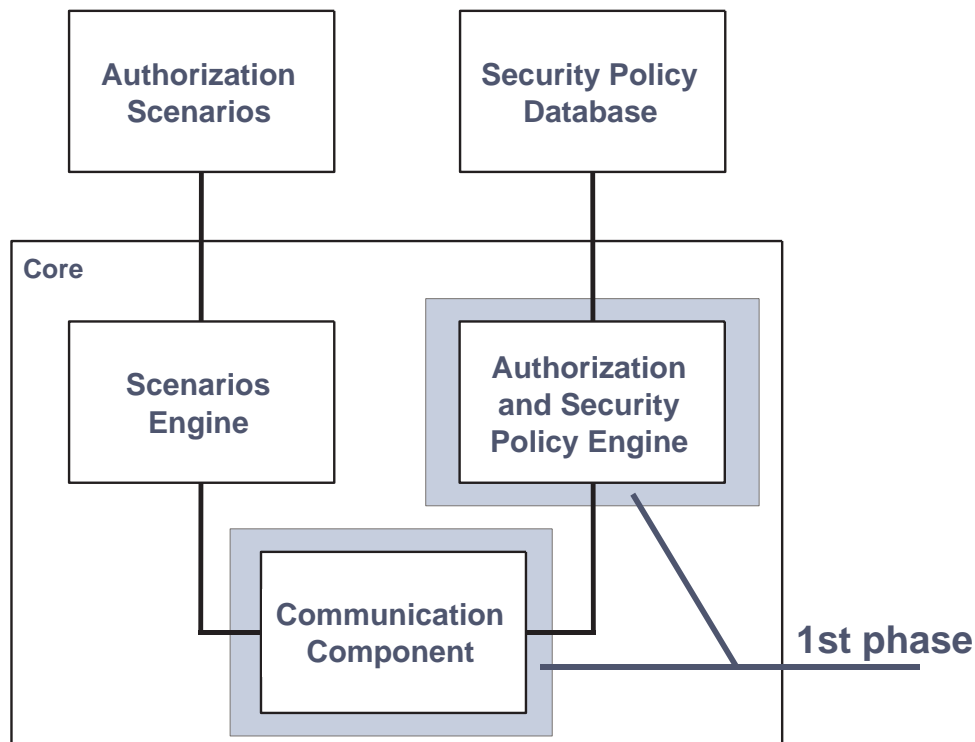


Figure 1: The core of prototype

### 3.2 Technical specification

The technical specification presented in *the Technical Specification for Authorization Service* [2] fully tallies with this specification. In figure 1, a specification of the core component of the AS service is presented. There are two gray boxes which display modules developed in the first place. All other modules will be developed in the second half of 2003.

It is significant that it is impossible to develop only the core component and then integrate this component with a security solution and other components. There should be some software working for the first prototype and this situation defines the order of developing components.

### 3.3 Components

From the technical point of view the first prototype of the AS will be composed of seven components:

- **as\_server** - this is a server; this component is responsible for storing security policy, generating security policy upon request and making the authorization decision,
- **as\_client\_admin** - this is a command line tool for adding new objects, subjects and attributes to security policy stored on **as\_server**,
- **as\_client** - this is a short program which obtains full security policy for user from **as\_server** and stores this policy into the CAS proxy file (this program is similar to **cas-proxy-init** [3]),
- **as\_enabled\_tcp\_server** - this is a sample server application which can use security policy that is provided by a client application (see **as\_enabled\_tcp\_client**),
- **as\_enabled\_tcp\_client** - this is a sample client application which can use a cas proxy file to access **as\_enabled\_tcp\_server**,
- **cas\_proxy\_viewer** - this is tool for printing on the screen security policy which is included into a standard CAS proxy file or the AS proxy file,
- **as\_enabled\_module** - this is a library which can be included into service for communication with the AS and getting full or part of security policy or authorization decision based on service request.

All these components are written in C language. Communication between components will be done on the basis of GSI (Grid Security Infrastructure) protocol [5]. A general structure and method of communication between prototype modules is presented in figure 2. The **as\_server** and **as\_enabled\_tcp\_server** are implemented as the standard Globus TCP servers.

The heart of this prototype will be **as\_server** where security policy will be stored and the authorization decision can be generated. For adding (and other editing operation) new rules to security policy database, a command line tool **as\_client\_admin** can be used. In one of the scenarios the AS will be compatible with the CAS and there will be a special program **as\_client** to get full policy from the AS server and store this policy to the CAS proxy file. Two components **as\_enabled\_tcp\_server** and **as\_enabled\_tcp\_client** will be created only for testing compatibility with the CAS and can properly work only in the CAS scenario [4]. For

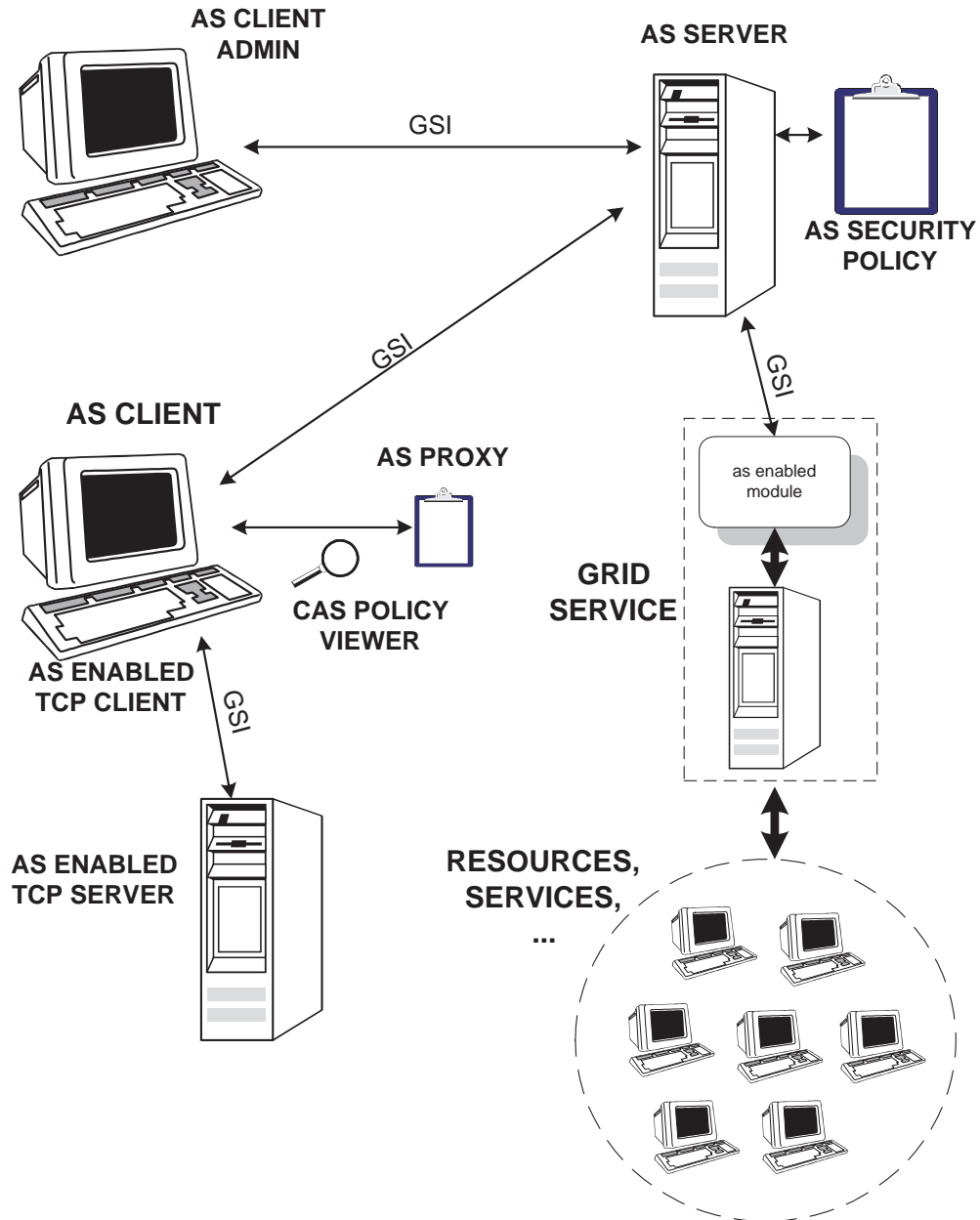


Figure 2: Communication between modules of prototype

displaying security policy information, which is included into the CAS proxy [3] or the AS proxy, the `cas_proxy_viewer` program can be used.

Generally, for cooperation between the `as_server` and other services there is a need to develop a special module (`as_enabled_module`), which can be included into the external service, probably as a library, for communication and other operation with `as_server`. This module will work in scenarios in which the authorization decision will be needed from the AS. With this module it will be possible to get the authorization decision for grid service by calling the API function.

### 3.4 The data structure

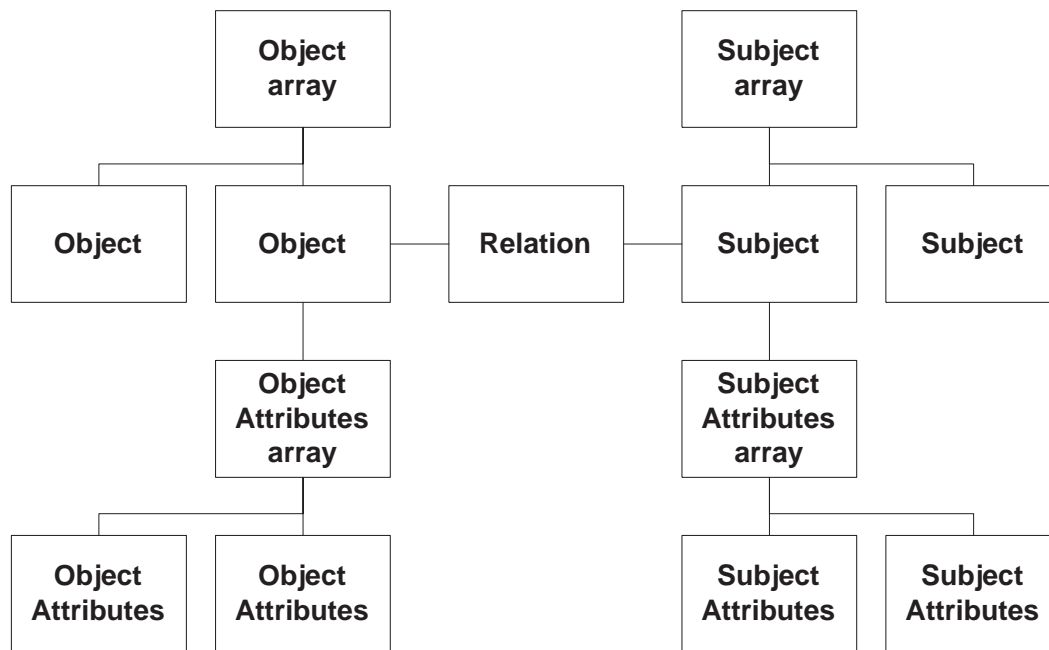


Figure 3: Proposition of data structure

In figure 3 the first proposal of the internal data structure of the `as_server` is presented. The structure is defined in order to cover a lot of possible appearances of data, which will be stored in the security policy database. For example, the CAS security policy, which is sent to the user, is defined below:

```

User: /O=Grid/O=Globus/OU=man.poznan.pl/CN=Marcin Adamski
notbefore: 1043743104
notafter: 1043829504
{
OBJECT\_NAME\_TYPE=wildcard
OBJECT\_NAME=ftp://adam.man.poznan.pl/home/adamski/Bak
SERVICE\_TYPE=file
SERVICE\_ACTION=read
}
{
OBJECT\_NAME\_TYPE=wildcard
    
```

```

OBJECT\_NAME=ftp://adam.man.poznan.pl/home/adamski
SERVICE\_TYPE=file
SERVICE\_ACTION=read
}

```

For the proper generating policy, this CAS policy structure can be written to the AS data structure as it is presented in figure 4. The structure presented in figure 3 will be expanded in the future to cover more solutions (depending on GridLab services, which would like to use `as_server`; the cooperation with some of these services is described below).

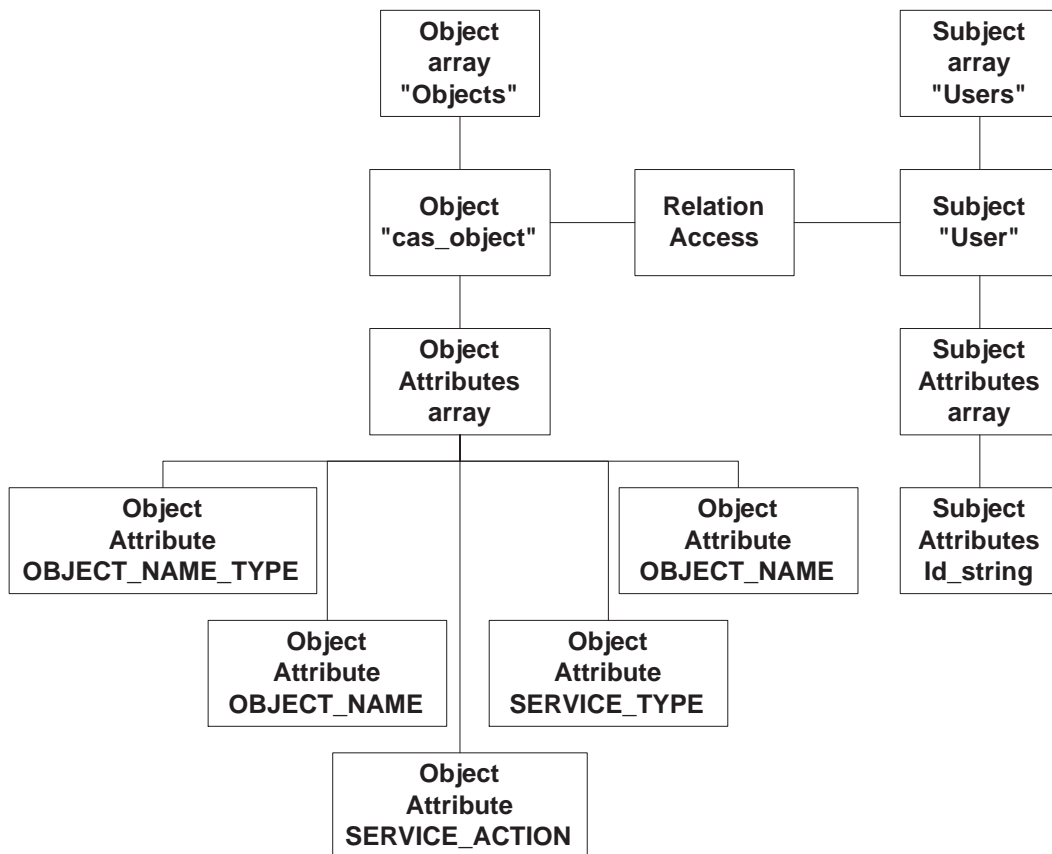


Figure 4: Proposed data structure for the CAS scenario

## 4 The integration with other workpackages

Each integration between service and the AS will probably be similar. In this section a plan of integration which can be used in most cases is proposed. In the near future, the following integrations will be carried out:

- integration with GridLab Resource Management System (GRMS)(WP9),
- integration with Portal (WP4).

At first, the AS will be integrated with the GridLab Resource Management System (GRMS) (WP9). It will be mostly experimental integration and the biggest number of errors and problems with functionality may be found during this stage. However, it is needed to find general solutions for problems that are likely to occur during next integrations. From these causes it will be the integration with a big support from WP6. The next integration might be done with the Portal (WP4).

There is a proposed plan of such integration.

1. At first, the `as_server` data structure for service has to be defined. It means that the `as_server` has to know what kind of authorization information the service would like to store on the AS and what kind of response the service waits for. At this point, it might need some minor changes into the AS inside.
2. The `as_enabled_module` has to be tested if the API functions can return information, which is needed by the service and, if it is possible, to integrate `as_enabled_module` with the grid service (on programming language level, library level and other settings). It is possible that at this point the functionality of `as_enabled_module` will be extended and corrected.
3. The data structure has to be put into `as_server`. A special configuration file will be created, which will contain a data structure definition and it will be possible to load this file to `as_server`.
4. The `as_enabled_module` has to be integrated with the external service. Support and help from WP-6 team for external service team will be needed.
5. Testing functionality. At first some real data have to be put to the security policy database. Next, after testing, some changes of functionality and some corrections might be needed.

## 5 The extensions of the AS

In this section, possible extensions of the AS are presented. They are sorted by their importance. The most important is subsection 5.1 and the least - subsection 5.5. All these extensions will make the AS more useful than it is given by the base functionality.

### 5.1 Storing security policy into the SQL database

In the first prototype of the AS it will be possible to store the security policy in the text local files only, in the structure which will be easy to service. After the first prototype it will be possible to store the security policy database locally on the `as_server` but in the database. This solution improves the performance of the AS. It is planned to use the MySQL database for storing data.

### 5.2 Extending scenarios functionality of the AS

One of the main reasons for creating the AS was a need to have the system of authorization which can work with the varied security scenarios. In the first prototype it will be possible to work with the two scenarios: the cas compatible scenario (in which full policy is generated for a user) and the simple super-scheduler scenario (in which a service gets the authorization decision from the AS for a user). After the first prototype, the extending of the scenarios engine will be provided to work with other security scenarios (More Complex Super-Scheduler Scenario, User to User Asynchronous Collaboration Scenario, ...). The scenarios more accurate are in *the Technical Specification for Authorization Service* [2] described.

### 5.3 Extending method of generating security policy

The methods of generating the security policy will be as easy as possible and not fully optimized in the first version. After getting sufficient experience about performance and other aspects of generating the security policy, it will be advisable to make a review of the codes and make some corrections and improvements.

### 5.4 GUI `as_admin_client`

For the first prototype it will be possible to use the command line tool `as_admin_client` for managing the `as_server` only. Next, it is planned to develop the GUI admin client which can help user to manage `as_server`. This client is likely to be written in Java. Another solution can be a basis of the web technology (the `www` client).

### 5.5 Possible integrations with other security solutions

The main security solution which is supported in the AS is GSI (the Grid Security Infrastructure). From the beginning the AS will have the modular structure and this solution makes it possible to divide specific functionalities into modules. All connections (the functions calling) to GSI will be put into one module and there will be a specific set of API functions which will communicate with this module. This situation enables to use other security solutions without the modification of the internal structure of the AS, only by exchanging one module. The new module will have



to cover all functions calls to this new security solution. It is planned to make support for Microsoft or SUN security technologies, if there is enough time to develop, but obviously the AS will be ready for support the future security technologies.

## 6 The overview of future tests

In this section a general overview of future tests, which will be performed on the prototype of the Authorization Service is provided.

### 1. The internal tests:

- individual tests of components described above ( `as_server`, `as_client` , `as_admin_client` ,...). There are not a lot of such tests because of the fact that the main actions take place between the cooperation of modules;
- testing the cooperation between `as_server` and `as_admin_client`:
  - opening the connection;
  - adding new objects and subjects to policy;
  - other editing operations on security policy;
  - testing the integrity of policy at the `as_server`;
  - closing the connection;
  - testing many simultaneous connections of the `as_admin_client` to one `as_server`;
- testing the cooperation between `as_server` and `as_client`:
  - opening the connection;
  - generating security policy;
  - generating a logical part of security policy;
  - making the authorization decision;
  - closing the connection;
- testing the cooperation between `as_server` and `as_enabled_module`:
  - opening the connection;
  - getting the authorization decision;
  - closing the connection;

### 2. The external tests:

- cooperation between external services (GRMS, Portal and others);
- performance tests;
- integrity of data tests.

Two main types of tests are defined: internal, which are focused only on the AS service, and external in which there is a need to cooperate with others workpackageges. When the functionality of the AS is extended in the future, more complex tests will have to be performed, which will cover other security scenario aspects. Of course, the simple tests presented above have to be completed before any more complex tests.

## 7 Summary

The main functionality of the first AS prototype will be completed at the end of February. At the beginning of March the AS will be firstly integrated with the GridLab Resource Management System (GRMS) (WP9). It is planned that by August the AS will have been integrated with the Portal (WP4) and other selected services. After the first prototype, the next stage of developing the AS will be started. At this stage the functionality of the AS will be extended with a possibility of cooperation with various security scenarios, storing security policy in standard SQL database and the possibility of cooperation with other security solutions (similar to GSI).

The tests will match the implementation, so all the internal tests of the first prototype will be started in March 2003. The external tests will start in April 2003. Next there will be time for complex tests of cooperation of the AS with external services. Probably, more functionality for testing will be ready at the end of the year 2003, so next tests will start after this time and will cover testing new functionality and some hard performance tests.

## References

- [1] M. Adamski, M. Chmielewski, S. Fonrobert, A. Gowdiak, B. Lewandowski, J. Nabrzyski, T. Ostwald, and J. Pukacki. *WP6 Security: Initial Requirements*, April 2002.  
[http://www.gridlab.org/Internal/Drafts/wp6\\_requirements\\_draft.pdf](http://www.gridlab.org/Internal/Drafts/wp6_requirements_draft.pdf).
- [2] M. Adamski, M. Chmielewski, S. Fonrobert, A. Gowdiak, B. Lewandowski, J. Nabrzyski, T. Ostwald, and J. Pukacki. *Technical specification for Authorization Service*, August 2002.  
<http://www.gridlab.org/Resources/Deliverables/D6.2b.pdf>.
- [3] The Globus Project. *CAS User and Administrative Guide*, August 2002.  
[http://www.globus.org/security/cas/alpha-r2/cas-guide-v0\\_2.pdf](http://www.globus.org/security/cas/alpha-r2/cas-guide-v0_2.pdf).
- [4] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke. *A Community Authorization Service for Group Collaboration*, 2002.  
[http://www.globus.org/security/CAS/CAS\\_2002\\_Revised.pdf](http://www.globus.org/security/CAS/CAS_2002_Revised.pdf).
- [5] J. Wray. *Generic Security Service API Version 2 : C-bindings*. Iris Associates, January 2000. <ftp://ftp.isi.edu/in-notes/rfc2744.txt>.