|  |  |
| --- | --- |
| Last Revised: | 6.18.2008 |
| Produced By: | NCICB Development Team |
| Version: |  |

Revision History

When you make a change to a document, you must add an entry to this Revision History table and you must manually type the Last Revised Date on the front cover.

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Revised by |
| 06/12/2008 | 0.1 | Draft to outline grid security options | Chris Piepenbring |
| 06/18/2008 | 0.2 | Added reference to caCORE white paper and removed recommendations pending decision by caArray team | Chris Piepenbring |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

1.1. Purpose 4

1.2. Scope 4

1.3. Definitions, Acronyms, and Abbreviations 4

1.4. References 4

1.5. Overview 4

2. Grid Security Architecture 5

2.1. Candidate 1 – Plugin based update to CSM 5

2.1.1. Description 5

2.1.2. Pros 6

2.1.3. Cons 6

2.1.4. Risks 7

2.1.5. Notes 7

2.2. Candidate 2 – Implement Grid security only on the Grid API 7

2.2.1. Description 7

2.2.2. Pros 9

2.2.3. Cons 9

2.2.4. Risks 9

2.2.5. Notes 9

2.3. Candidate 3 – No changes to Grid API 10

2.3.1. Description 10

2.3.2. Pros 10

2.3.3. Cons 10

2.3.4. Risks 10

2.3.5. Notes 10

3. Recommendations 10

# Introduction

## Purpose

The Candidate Architecture Document (CAD) documents the trade-offs and risks associated with proposed architectural solutions to implementing grid security in the 2.1 release of the system. The current grid service for 2.0.2 does not implement grid security and, as such, returns only public data to the user or grid client. In addition, there is a desire to move all logins away from using the local CSM/UPT provisioned model to a global caGrid provisioned model. This will enable consistent user authentication across multiple instances as well as across all caBIG applications.

This document will propose architectural alternatives to implement the grid security model along with their strengths and weaknesses, and document the recommended solutions.

## Scope

This Candidate Architecture Document applies to the grid security design and implementation within 2.1 and beyond. The overall architecture of the project is sound and is not being reevaluated in the scope of this document.

## Definitions, Acronyms, and Abbreviations

None

## References

caArray Software Architecture Document

caArray Continuous Integration Environment: <http://6116-caarray-int.nci.nih.gov:9191>

Rational Unified Process, version 2003.06.15

## Overview

The Candidate Architecture Document is divided into sections. Each section highlights a specific area of the system architecture and documents the issues that have been discovered and led to the reevaluation. Within the section, several architectural alternatives are proposed, then trade-offs and risks are documented, along with any important notes about the alternative.

At the end of the document, recommendations are made for the best solution to implement going forward. It is important to note that the best alternative is not always the best technological solution, as schedule and cost are balanced in the recommendations as well.

# Grid Security Architecture

## Candidate 1 – Plugin based update to CSM

### Description

The caGrid and CSM teams are preparing a white paper on a solution to migrating applications from a CSM-based authentication scheme to one based on caGrid. This solution will require no (or minimal) code changes within the application and will also handle migration of existing CSM (local) userids to caGrid (global) userids. The current plan is for the caCORE team to develop and deliver this solution to application teams over the coming months. The following diagram shows the planned architecture within the grid, CSM, and the application.

In this model, all users will have a grid identity. This grid identity is based on being authenticated at one of two (current) sources: the NIH-wide LDAP service or the global caGrid Dorian instance. There may be other trusted identity providers added in the future, but at this time, all users must either have a valid NIH userid or be provisioned on the single caGrid Dorian instance.

For a user to log in to any instance, they will first require one of these two IDs. They can then be provisioned on that caArray instance, which in this model involves assigning the grid userid to the correct groups and granting permissions as necessary. This process will be identical (?) to the method that is currently used to provision users with valid LDAP credentials.

For users that already have existing accounts provisioned through CSM/UPT on a caArray instance, the user will be asked for a valid grid credential (userid) upon their first login to the system after upgrade. If the used does not have a valid grid credential, the system will provision them one on the global caGrid Dorian automatically. This will be done through the CSM plugin and will not require application code or administrator intervention, except possibly designing a display page to request the user’s grid credentials. Once the user’s grid credentials have been entered or created, the user’s userid in ’s CSM database tables will be replaced with their grid userid. From that point forward, the user must log in with their grid credentials (see next paragraph).

Users who have grid credentials will log in using 3 pieces of information:

* Which identity provider they are registered with (NIH directory or Dorian)
* Their username at that identity provider
* Their password at that identity provider

The application will use a LoginModule from CSM to authenticate the user at the appropriate identity provider and then use a to-be-determined CSM module to register the user on the grid, which obtains the user’s grid credentials and their grid userid. From this point on, the same authorization code is used in with no changes, just using the grid userid instead of the locally provisioned userid.

Access to the grid service will be secured such that users with no grid credentials (anonymous grid query through the caGrid portal, for example) or users with valid grid credentials that do not have a corresponding account on the instance will see only public data. Users with valid grid credentials that have been provisioned on the instance will have the same permissions enforced as they would if they were directly logged in to the instance.

More details on the technical implementation and interaction between services are provided in a white paper developed by the caCORE team entitled *Migrating Existing Web Applications from CSM Based Authentication to caGrid Based Authentication*.

### Pros

* Zero (or minimal) development necessary by the team
* Guaranteed to be consistent with the implementation in other caBIG applications
* Code base is maintained and upgraded by caCORE and caGrid teams

### Cons

* caCORE team development schedule for this solution precludes using it in the 2.1 release – 2 months to Alpha has been claimed, but no schedule exists
* Requires all users to exist globally. They will have to be provisioned at the caGrid Dorian if they do not have a NIH userid.
* Possible “double provisioning” of users: once at caGrid Dorian to establish a global grid identity and once on the local instance to place them in the correct groups and assign permissions.

### Risks

* Uncertain development schedule – 2 months could turn out to be longer
* Technical challenges exists with implementing the proposed solution on ’s architecture (Grid service running in JBoss 4.0.4 container connecting to EJBs in JBoss 4.0.5 container) Passing credentials between these two containers would require either a change in the proposed architecture for this solution, change in the architecture, or a solution that provides some sort of “trusted communication” between the grid service and the Remote Java API. (More in section 2.2.1 below)

### Notes

At some point, it will probably be necessary for to use a standard solution provided by the caGrid team to stay consistent with other applications. The question is how soon to adopt this solution (or a matured replacement solution). This is currently the only solution that will support true grid identity based logins to the web application.

It is also possible for the team to implement a similar solution on their own to avoid the schedule constraints introduced by waiting on the caGrid team’s implementation. This is a significant LOE however, and would certainly not be feasible for the 2.1 release.

## Candidate 2 – Implement Grid security only on the Grid API

### Description

This is a smaller scale implementation that would most likely be a bridge solution until a full grid security solution is implemented. In this solution, users who access the application through the web interface will continue to use logins provisioned directly through CSM/UPT. However an additional field will be added to the user information allowing it to store both a local userid (which could be LDAP based or CSM based) as well as a grid userid.

As above, access to the grid service will be secured such that users with no grid credentials (anonymous grid query through the caGrid portal, for example) or users with valid grid credentials that do not have a corresponding account on the instance will see only public data. Users with valid grid credentials that have been provisioned on the instance will have the same permissions enforced as they would if they were directly logged in to the instance.

The implementation of this, however, would be that the grid userid is converted to the local userid after authentication and that local userid is used for all authorization actions from that point on. If the grid userid does not have a corresponding local userid, then the user is assumed to have anonymous access and can see only public data.

This solution has an additional load on the administrator to have to enter a grid userid for the user either when provisioning them in UPT or updating it later when a grid credential is obtained. However, if a user has an NIH ID and can be authenticated through their LDAP, the grid id can be automatically determined and populated at the time the user is provisioned in UPT.

There are some code changes necessary for this solution:

* A new DB table linking the local and grid userids must be created and accessed from the Remote Java API code. Adding a column to the existing CSM tables could also do this.
* The Grid service must be modified to accept the user’s grid credentials, extract the grid userid, and pass it through to the Remote Java API over some sort of trusted channel (see below).
* The UPT provisioning screens must allow the administrator to enter a grid userid in addition to the local userid.

In this solution (as well as potentially in candidate 1 above), the Grid service will have a grid userid that has already been authenticated. As such, it will have only a username and no associated password. The current architecture requires that a caller to the Remote Java API pass both a username and password, which the API will authenticate in the local CSM database. In the grid service case, the Remote Java API will receive only a username and must trust that the Grid API has already authenticated this user.

This provides a potential channel for circumventing security if the Remote Java API cannot guarantee that ONLY the Grid API can access the service in that manner. There are several possible solutions for this, including using SSL connections (with client and server authentication) to implement the EJB remote invocation. The simplest, and proposed, solution is to create a unique GUID at install time that is known only by the local instance that will be passed as the user’s password when the Grid API invokes the Remote Java API. Only if this GUID is correct will the authenticated username be accepted. The diagram below shows the interaction when the grid service is invoked.

### Pros

* Easy solution that is achievable for release 2.1
* Provides grid security on calls to the Grid API
* Can be replaced by a full implementation in future releases
* Does not remap any existing user ids or force “double provisioning”

### Cons

* Not a full-scale grid security implementation
* Users will have different local user ids on each instance of caArray, although they will share the same grid id
* Requires some development by the team

### Risks

* It is possible, but remote, that a user could determine the GUID by looking at configuration files on the install machine and use this GUID to fashion an application that can circumvent security. However, any user with this access could also likely directly access the database and see the protected data anyway
* Adding a table or column to CSM may make it difficult to upgrade to new CSM releases

### Notes

This is obviously an interim solution until a full grid security implementation can be done. It is a trade-off analysis for whether this is worth doing for the 2.1 release or just leaving the Grid API restricted to only public access (see 2.3)

## Candidate 3 – No changes to Grid API

### Description

This solution is to leave the Grid API the way that it is and limit grid queries to only public data.

### Pros

* Zero effort for the team
* Can be replaced by full grid security implementation later

### Cons

* No access to private data
* Without grid security implemented, caBIG Gold compliance is unlikely

### Risks

* Delayed compliance with grid initiatives

### Notes

Obviously this is only a short-term decision. Eventually, will have to move to a grid security model. The question here is if we do nothing for release 2.1 and implement solution 1 (or the final caGrid team solution) at a later date.

# Recommendations

TBD pending caArray management review