EUROPEAN MIDDLEWARE INITIATIVE

EMI X509 DELEGATION AGREEMENT

EMI DOCUMENT

Document identifier:	**20121024-emi-delegation-agreement-v1.1.odt**
Date:	**28/01/2013**
Activity:	
Lead Partner:	
Document status:	**Final**
Document link:	

Abstract:
This document provides a high-level overview and description of agreed best practice for EMI project teams when writing software that support interactions between a client that identifies itself with an X509 credential and a server that desires a delegated X509 credential, based on the client's identity.

Document Log

Issue	Date	Comment	Author / Partner
1	15/3/2012	Initial Document	Paul Millar
2	24/10/2012	Update based on feedback	Paul Millar
3			

Document Change Record

Issue	Item	Reason for Change
1		
2		
3		

TABLE OF CONTENTS

1. INTRODUCTION

For most of EMI, X509 certificates—either EEC (Certificate Authority issued certificates) or proxy certificates—are used to identify individuals. The signed attributes of the proxy certificate (such as VOMS FQANs) are often used by services to make authorisation decisions.

A service may need to "act on behalf" a user; that is, the service will interact with some third-party service. This interaction is often (but not always) triggered to satisfy some inbound request from the user. This interaction with the third-party service is identified as being originated by the user and often the process is authorised based on attributes contained within that user's proxy certificate.

One way of solving this problem is to extend trust between two services: between the service with which the user is directly interacting and the third-party service. This would allow the one service simply to state it is making the request on behalf of the user. The third-party service would make any internal auditing and authorisation decisions based on what it was told, trusting the supplied information. In general EMI software does not allow for such a trust relationship between two or more services; moreover, in grid environments, such trust relationships are unusual.

In the absence of shared trust between services, the user must do something that results in the service obtaining some security token that the service uses when interacting with any third-party service. The process of creating this security token on the server is called delegation and the resulting credential is a delegated credential. The delegated credential must contain the end users identity and sufficient attributes to allow the service to use the credential when interacting with some third-party service on behalf of the user.

One approach for delegation is to provide a security token that includes both the identity of the service and that of the end user. Such a delegated credential would assert both that the service is originating the request and that the request has been authorised by (and is on behalf of) the user. Such an approach is used by services that work with SAML tokens, such as UNICORE. However, there is currently no corresponding approach when working with X509 credentials.

Instead, EMI software takes a simpler approach. A delegated credential is simply another X509 proxy credential that identifies the holder as an agent of the user. The proxy credential is located somewhere within the service so it may use this credential when interacting with third-party services.

1.1. PURPOSE OF DOCUMENT

This document provides a high-level view of delegation and describes the recommended approach for teams when writing software that must acquire a delegated X509 credential when the client has an X509 credential. It assumes that such delegation is driven by the service's need for an X509 credential that is used when interactive with 3rd party software.

The agreement is beneficial because having a common approach will improve the efficiency of the teams by avoiding unnecessary or duplicated effort. Another benefit of using a common approach is that security updates involving delegation may be handled in a common fashion, so reducing the impact on each software team.

1.2. NOMENCLATURE USED IN THE DOCUMENT

This document uses certain words, written in capitals, that indicate formal levels of binding of the terms of this agreement. The names are deliberately similar to those described in RFC 2119 and have broadly similar meanings. These are described below, paraphrasing the corresponding descriptions in RFC 2119:

MUST	is an absolute requirement under this agreement.
MUST NOT	means that the definition is an absolute prohibition under this agreement.
SHOULD	means that there may exist valid reasons in particular circumstances for ignoring these items, but the full implementations must be understood and carefully weighted before choosing a different course.
SHOULD NOT	means that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label
MAY	This word means the item is truly optional. One product team may choose to follow the item because of external requirements or because they feel that it enhances the product while another product team may omit this item.

2. GRIDSITE DELEGATION

This section describes the GridSite Delegation protocol and how it is adopted within EMI.

2.1. OVERVIEW

GridSite Delegation is an API for delegation that uses SOAP-over-HTTP. GSD works by identifying a delegator and delegatee. The delegator is the agent that has the credential to be delegated; the delegatee is the agent that will receive the delegated credential. Assuming the client-server model, it is usual for the delegator to be the client and the delegatee to be the server. Although this connection is not required, in the remainder of this document the words "delegator" and "client" are taken as synonyms, as are "delegatee" and "server".

The GSD API v2.0.0 is defined as a SOAP-based API that uses SOAP-over-HTTP. It has nine operations, each of which follows the request/response pattern common to remote-procedure calls.

These operations support a delegated credential life-cycle. It allows client software to:

- trigger the creation of a time-limited delegated credential,
- querying aspects of that credential, and
- requesting the credential's immediate destruction.

GSD also allows the client to discover information about the delegation service itself.

2.2. LIMITATIONS OF GSD

GSD does not provide any mechanism for the service to request delegation. The client must either follow some predetermined pattern (e.g., always delegate) or such support is provided through some application-specific API; for example, by returning a specific error code (e.g., REQUIRES_DELEGATION) the service may indicate to the client software that delegation is required.

GSD does not provide any mechanism for back-delegation. Back-delegation is the process where some third-party requests further delegation. Such a mechanism would allow some trusted third party to obtain a delegated credential on demand. Instead, GSD assumes a tight coupling between the delegation service and the service using these certificates; for example, that they share a common repository for storing delegated credentials.

The GridSite specification states how to calculate the id, which is too implementation-specific. This requirement should be relaxed in future versions of the specification to allow more flexibility with server implementations.

2.3. CLARIFICATION OF GSD

The current description of GSD [GDS] has unfortunately conflated behaviour that should be part of the GSD standard with that of the reference implementation. This mixed description makes it hard to understand whether a server is compliant to GSD unless they implement the exact same functionality in precisely the same fashion. It is possible that the exact behaviour may become a performance bottleneck; therefore, it is desirable that different implementations may adopt different approaches while maintaining sufficient commonality to allow clients to work with any conforming implementation.

To fix this problem, a new document will be written to identify GSD as a standard, independent from the implementations currently available. At time of writing, this document is planned to be written through a new Open Grid Forum (OGF) group within the Security area. This new group will be

established to look into X509 delegation. The OGF formal process will be used, ensuring the high quality of the result document.

2.4. ADOPTION OF GSD API

When writing new software that is to support delegation, the EMI project team SHOULD use the GSD SOAP-over-HTTP-based API for over-the-wire communication. This is irrespective of how that support is implemented (standard library or custom implementation). Using the GSD API is an expression how the delegation process has a common nature. It also allows the software team to take advantage of existing library code, if appropriate.

Existing software that uses an alternative delegation process SHOULD examine whether they can use GSD and an alternative to their existing implementation. This is because using a uniform approach within EMI will help reduce support costs. However, a product team MAY decide not to adopt GSD for a particular software-component because the cost/benefit balance points to keeping the existing implementation. Such a decision MUST be made on the granularity of a software-component or at a finer grain.

If a service needs more functionality that is available from the GSD API then the GSD API MAY be embedded within some large SOAP-based API. Adding support for back-delegation (see above) is an example of additional functionality that a service might need.

Such an embedded use of GSD API SHOULD preserve the namespace of the GSD operations. This is to allow unmodified GSD clients to work. The software SHOULD make use of as many of the GSD API calls as is practical.

2.5. ADOPTION OF EMI'S GSD LIBRARIES

EMI provides client and server implementations as native code and Java libraries. EMI projects SHOULD use these libraries rather than writing their own code. It is possible that EMI projects have sufficiently disparate requirements that no single library implementation is able to satisfy them all; however, if a project team discovers a problem with a library then they SHOULD contact the team supporting that library before writing an alternative implementation.

EMI provides Java and C implementations of the both client and server code. These are available as the following packages:

delegation-api-c	C and C++ libraries for client & server applications; based on gSOAP.
delegation-java	core functionality to satify GSD behaviour.
delegation-service-java	Provides glue code for delegation-java, providing a stand-alone client and a servlet for deploying in a standard servlet container.

Note that the gridsite packages ('shared', 'services', 'service-clients', etc) make use of older GSD v1.1 API, which is incompatible with GSD v2 and so SHOULD NOT be used.

2.6. TRUST RELATIONSHIP BETWEEN USER AND A SERVICE USING GSD

The grid environment is currently using X509 proxy certificates without a mechanism for managing and enforcing a trust relationship between agents. Despite this, if a service uses the GSD API then there is an implicit trust relationship between the delegator (typically, the client) and the delegatee (the service). This section describes how software should operate under this trust relationship.

When a credential is delegated to a service then that service obtains the ability to impersonate the user. The service MUST act responsibly with this information. It SHOULD NOT use delegated credential for any purpose other than the one for which it was delegated. The service MUST take reasonable steps to protect the delegated credentials from being stolen and SHOULD try to limit the time the credential is kept. If the user requests their delegated credential be destroyed then the service MUST do this in a timely fashion.

The service provides a communication channel for the client to send requests. This channel MUST allow the client to identify the service with cryptographically strong assurance. The channel MAY be encrypted; however, note that encryption of the channel is not needed for the integrity and security of the delegation process itself.

3. SUMMARY

Delegation is the process of allowing a service to act on behalf of an agent. GridSite delegation provides one protocol for achieving delegation. This document describes an agreement within EMI to adopt GridSite delegation v2 as a preferred protocol for achieving delegation. Future work may involve extending this protocol in backwards compatible fashion.