



NextGRID Security Profile Use Cases

Editors:	Joris Claessens	EMIC
	Thomas Leonard	IT Innovation

Date	Author	Comments	Version	Status
12 Oct 2006	Joris Claessens	Initial outline	V0.0	Draft
15 Dec 2006	Thomas Leonard	Added intra-org use cases	V0.2	Draft
11 Jan 2007	Joris Claessens	Added references	V0.3	Draft
12 Jan 2007	Thomas Leonard	Minor update to check policy	V0.4	Draft
17 May 2007	Sven v d Berghe	Clean up	V0.5	Draft
14 Sep 2007	Joris Claessens	Minor update to emphasize the use of a federation context	V0.6	Draft
12 Dec 2007	Joris Claessens	Polished references	V0.7	Draft
18 Jan 2008	David Snelling	Final Proof	V1.0	Final



1 INTRODUCTION..... 3

2 USE CASES..... 3

2.1 Cross-organizational security interactions 4

 2.1.1 Create Federation 4

 2.1.2 Manage Federation..... 4

 2.1.3 Retrieve Federation context relevant metadata 5

 2.1.4 Contextualized secure service invocation 5

2.2 Intra-organizational security interactions 5

 2.2.1 Get Token 5

 2.2.2 Validate Token..... 5

 2.2.3 Manage Client and Service Policy 6

 2.2.4 Check Client and Service Policy..... 6

REFERENCES 6

1 Introduction

This document describes use cases motivating the NextGRID Security Profile.

The NextGRID Security Profile is part of the NextGRID Generalized Specifications, which aims to capture NextGRID architectural concepts in a set of composable profiles. These profiles are specified in such a way that they could be implemented in terms of other well known specifications. While overall consistency is achieved at the conceptual level, and captured through the motivating Use Cases accompanying each specification, the implementation in terms of other specifications may not be consistent between different profiles. Thus each profile defines an implementable realisation of the underlying concept, but implementers of the full NextGRID architecture may need to support multiple competing underlying specifications.

2 Use Cases

The NextGRID Security Use Cases are depicted below.

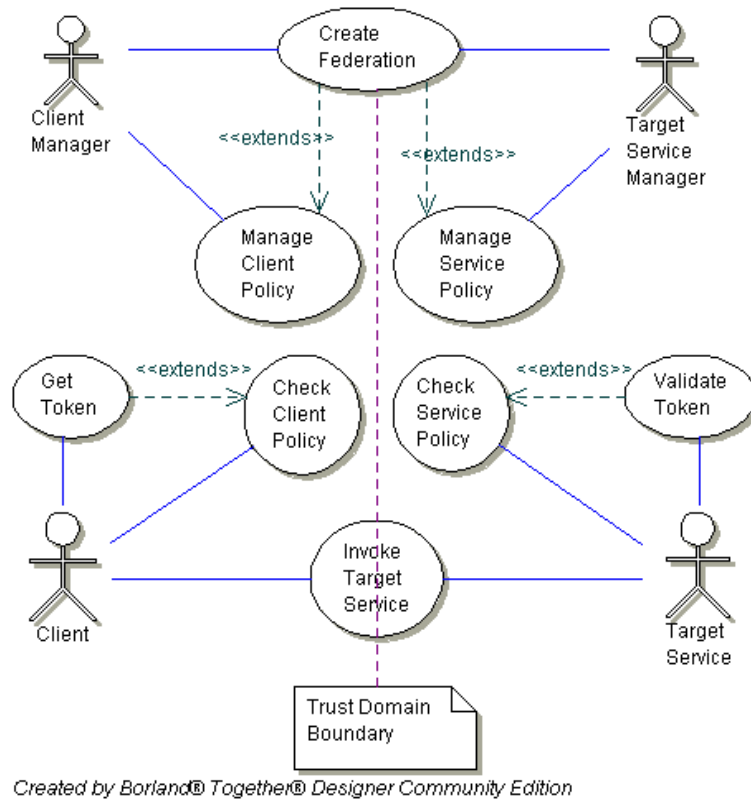


Figure 1 NextGRID Security Use Cases

The overall storyboard goes as follows. A Client in one organization wants to securely invoke a Target Service exposed by another organization. The secure invocation requires the existence of a federated trust relationship between the two organizations. Such a federation is created with the help of the Client and Target Service Managers. In both organizations, specific security policies

can be associated with a defined federation, and managed by the Managers. Furthermore, Client and Target Service Managers may be able to perform specific cross-organizational federation management operations. The secure web service invocation itself needs to happen with a so-called security token which the Client needs to get, and the Target Service needs to be able to validate. Issuance and validation of security tokens may again be subject to specific security policies. Prior to the actual service invocation, the Target Service may support the Client in selecting the appropriate federation context by providing relevant federation context metadata. A federation context identifier should be included in the service request, allowing the Client to indicate in which federation context it wants to access the service. Security policies and service request processing may be different in different federation contexts.

2.1 Cross-organizational security interactions

Interoperability matters most in the cross-organizational interactions discussed below. Moreover, what happens inside the client and service domains can vary; for example, different authorization approaches can be adopted.

An important technical note to make is that we here consider the “U-type” federation model. In this federation model, there is no WS-Trust interaction with a Security Token Service, which crosses organizational domain boundaries. In contrast, the “W-type” federation model, which is typically the default model in the WS-Federation specification, there is a cross-organizational WS-Trust interaction between the client and the service-side token service where the client exchanges its client-side token for a token from the service-side with which it can access the service.

Thus, in case both client and service domains adopt the default WS-Federation approach, there is an additional cross-organizational security interaction in addition to the four outlined below. Minimally, in the “U-type” federation model, and in the case where domains would even choose not to use WS-Trust at all, we claim that at least four interactions will be required for a contextualized dynamic security scenario.

2.1.1 Create Federation

This can happen completely out-of-band and is therefore an optional interaction, but to cater for more dynamic behaviour, an in-band interaction to bootstrap a new federation or security context may be desired.

2.1.2 Manage Federation

The use cases “Manage Client Policy” and “Manage Service Policy” are fully drawn on their respective sides of the trust boundary. Organizations indeed typically manage their own security policies. However, specific client/service policy management operations can cross trust domain boundaries in a federation context.

One specific example of a cross-organizational federation management interaction is a “client policy mapping to federation context”. It enables the client side to configure the mapping from specific client/federation-side security settings (e.g. claims) to specific service-side security settings (e.g. process roles); this interaction is optional.

2.1.3 Retrieve Federation context relevant metadata

Before the Client can get a token to invoke the Target Service in a specific security context, the Client needs to be able to retrieve some security context related information from the Target Service, which will support the discovery and selection of the appropriate security context at the client side. This interaction therefore allows a client to get (any) information from the service which is helpful to discover and select the appropriate security context within which it should access the service; this interaction could happen out-of-band, but is highly recommended for dynamic situations.

2.1.4 Contextualized secure service invocation

This interaction specifies how a client must access a service, in a way that mutual authentication and confidentiality is guaranteed, and with the security/federation context securely attached to this (i.e. the signature on the request message must cover the body together with the federation context identifier); this interaction is mandatory.

2.2 Intra-organizational security interactions

2.2.1 Get Token

Before the client invokes the service, they request a security token from a token service within their own organisation. In order to get this token, the client will have to provide some other security credentials, such as an X.509 identity certificate. If the presented credentials are sufficient, the user is issued a token which they can pass to the service.

Because of the overhead involved in issuing tokens, it should not be necessary for the client to get a new token for each cross-organisational message they send. Instead, the client should be given a token with a limited lifetime (perhaps an hour or so, in line with the specific application security requirements), and this token will be used with multiple messages. When the token expires, the client must ask for another one.

The tokens should not be overly long-lived (weeks or months), as the client organisation's ability to revoke the access of its members depends on the user having to get a new token soon after the token service's policy is modified.

2.2.2 Validate Token

The target service invoked by the client will not typically be able to validate the token itself, since there will be many services run by the service provider and it is not practical to duplicate the rules determining which tokens are acceptable at each one.

Instead, the target service sends a validation request to the service provider's token service, which decides whether the token is acceptable. Different clients will have different levels of access to services, and therefore the client must specify the context (SLA) in which they are invoking the service. The token will be validated within this context.



2.2.3 Manage Client and Service Policy

There are a number of security policies involved in this model. There is the policy within the client organisation which determines whether a particular client's request for a token is granted, and another policy at the service provider to decide whether a presented token is valid.

The client organisation manages the validation policy at the service provider as described in the cross-organisational "manage federation" use case above. How the client and service provider manage policies within their own organisations is out of scope for this specification, but it may be that the same methods will work in these cases too.

2.2.4 Check Client and Service Policy

Clients may be permitted to see some or all of the policy controlling the issuing of tokens. At the very least, they will be able to tell whether they are permitted to get a token simply by asking for one.

The service needs to check its own policy to know what is required from a client. It may also pass some of this information back to the client (e.g. that the client needs to quote their SLA ID in order to use the service, or that they need a SAML token from a particular STS).

References

[WS-Trust] WS-Trust 1.3. OASIS Web Services Secure Exchange (WS-SX) TC. <http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.pdf>, TC page: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-sx

[WS-Federation] WS-Federation 1.2. OASIS Web Services Federation Language (WSFED) TC. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsfed