EbXML Registry Security Proposal

Technical Architecture Security Team

May 10, 2001

Status of this Document

There are three categories of ebXML deliverables:

- Technical Specifications conform to the ebXML Requirements document.
- Technical Reports are either guidelines or catalogues.
- White Papers constitute a snapshot of on-going work within a Project Team.

This White Paper represents a report that has been approved by the Technical Architecture Security Team and has been accepted by the ebXML Steering Committee.

The material in this document constitutes a snapshot of on-going work within this Project Team.

Distribution of this document is unlimited.

This version:

http://www.ebxml.org/specs/secREG.pdf

Latest version:

http://www.ebxml.org/specs/secREG.pdf

Authors

- Krishna Sankar [ksankar@cisco.com]
- Farrukh Najmi [Farrukh.Najmi@east.sun.com]

Contributors

- o Munter, Joel D [joel.d.munter@intel.com]
- o Maryann Hondo [mhondo@us.ibm.com]
- o Nieman, Scott [Scott.Nieman@NorstanConsulting.com]

Abstract

This document is a draft proposal whose purpose is to solicit additional input and convey the security aspects of the ebXML Registry.

Referenced Documents

EbXML Technical Architecture Risk Assessment [secRISK]

Table of Contents

1. Business Problem(s)

Note: This version (0.003) is a very preliminary version. It is more an aggregation of the ideas and has an engineering bias. It needs a lot of rewrite to make it into a specification. Let us start from the ideas and …

I am expecting comments from all with improvements and ideas.

How can we make this simpler yet extensible and secure ?

1.1. Authentication

The ebXML Registry is being used by businesses for various activities including publishing information, discovery, ad-hoc query, drill down etc. Authentication is required to identify the ownership of content as well as for identifying what “privileges” an entity can be assigned to with respect to the objects in the registry.

In addition, organizations might want to create private spaces for their partners and the access to these private spaces needs the authentication of users as well.

1.2. Integrity

The ebXML Registry is global and distributed, which contains information about capabilities, business process definitions and other XML documents. The integrity of the registry content is of great importance to those who refer to and use these documents for mission-critical business applications.

It is expected that most business registries do not have the resources to validate the voracity of the content submitted to them. The minimal integrity that the registry must provide is to ensure that content submitted by a Submitting Organization (SO) is maintained in the registry without any tampering en-route or within the registry. Furthermore, the registry should make it possible to identify the SO for any registry content unambiguously.

1.3. Confidentiality

The registry should provide capabilities for organizations to publish information, which are seen only by their partners. We cannot assume that all published information is public.

There should be capabilities to publish information to be viewed by a subset of users – for example the organization’s partners.

There are two types of confidentiality needs.

1. “On the wire” confidentiality that ensures that content cannot be read on its way to the registry
2. “In registry” confidentiality that ensures that content is only visible to authorized parties (e.g. the partners of the SO)

1.4. Authorization

An issue related to the confidentiality and integrity is the appropriate access to the data, or authorization. The information publishers should be able to define who can access and do what with their data. The registry should provide authorization mechanisms to achieve this.

General

There need to be security around the registry as well as individual security around the documents.

2. Requirements

The ebXML Registry security requirements are derived from the business problems in the previous section:

1. The registry security system should have user level security
2. The registry also should have document level authorization security
3. The registry must support a set of default document level authorization security policies
4. The registry should allow the default document level authorization security policies to be customized by publisher of that document
5. The authorization policies (for example role based access control) should be granular to specify and limit access at the content (or object) level as well as at the operation (or method) level
6. The Registry Service should enforce access control policies when servicing client requests
7. All users who access the registry should be authenticated using standard schemes
    a. This does not preclude a guest level access which could be used by users who are not authenticated
    b. The guest level access, if present, should be the least secure mode
    c. The guest level access, if present, should not get any privileges by default, which means the default privilege should be no access to the guest level.
8. The main function of the authenticator is to ensure that only known entities can access the registry
9. The registry authentication service should be able to be boot-strapped (including adding credentials, profiles et al) in a secure way
10. The Registry authorization scheme should be able to provide, at a minimum, the following roles (REF : ISO/IEC 11179):
    a. RegistrationAuthority(RA) – Organization authorized to register data; usually the owner of the registry
    b. ResponsibleOrganization(RO) – Organization Responsible for the contents ; usually the one which signd the content
    c. Submitting Organization (SO) – One which submits content incl update, delete etc – ie one that has content submission and content life cycle management authorization ; this could be many entities including individuals and departments inside an organization
    d. Guest – a user who has some set of minimum capabilities

11. The authorization scheme should be flexible enough to have public and private areas within the registry

12. The security system should not prevent the registry from being a completely private registry

13. In order to avoid authentication for every message/interaction, a session based security scheme could be used

    a. If a session-based scheme is used, the session should not be permanent.

    b. It is RECOMMENDEED that the session time-out be configurable by the Registry Administrator

14. The security system should be able to prevent registry spoofing i.e. prevent an entity from posing as the intended registry when its not the intended registry

15. The security mechanism should be able to prevent the so-called “man-in-the-middle” attack, the “replay” attack and denial of service attack.

16. Messages between Registry clients and service need to be confidential

17. Registry content may be confidential and disclosed only to authorized parties

18. Contents may not be visible to registry if registry is not trusted or there is no need for the registry to see the contents.

    For example, if the content contains sensitive information like user names and passwords, the SO can encrypt the contents. They can still be kept in the registry but the registry would not be able to "see" them

    Meta data is always visible to the registry.

19.

3. ebXML Registry Security

3.1. Security rules

Release 1 will employ credential-based authentication (digital certificates and signatures), simple default role based access control and message level confidentiality and encryption.

These are the security rules, which will be implemented in Release 1.

- Authentication is required on a per request basis

  Which means from a security point of view, all messages are independent; there is no concept of a session or a long-standing conversation ; there is the concept of a multi-message conversation

- Default Access Control Policies
  - For Release 1, the philosophy is "Any known entity can publish and anyone can view"
  - So, the following roles will be built-in the registry:

Role	Default Permissions	ISO 11179 Cross Reference
ContentOwner	* implying all methods on ONE ManagedObject (full permissions to ONE object – the one the entity created)	Submitting Organization (SO)
RegistryAdministrator	* implying all methods on ALL ManagedObjects (full permissions to **ALL** objects in the Registry)	RegistrationAuthority(RA)
RegistryGuest	All getXXX methods on ALL ManagedObjects (read-only access to all content)	Guest
		ResponsibleOrganization(RO) This is derived from the signature of the content. There are no specific registry permissions for the ResponsibleOrganization

- At the time of content submission, the registry will assign the default ContentOwner role to the Submitting Organization (SO) as authenticated by the credentials in the submission message
    - In Release 1 it will be the DN as identified by the certificate
- All requests performing sensitive operations are signed
    - Which means all non-getXXXX messages will need signature
- All content must be signed
- For Release 1, clients need not use certificates and will have the default RegistryGuest privileges
- Furthermore, in Release 1, the role based access control and access control policies are not visible outside the registry
    - Which means the clients will not be able to submit custom access control policies
    - In short, for Release 1 :
        - The Registry Service by default establishes the access policies
        - Only the SO and the Registry administrator have access to all methods and the clients can access the getXXX methods
        - Anyone can publish content, but needs authentication
        - Anyone can access the content and no authentication is required

- Release 1 will rely on TRP for message level authentication, confidentiality & integrity
- Registry is trusted to see all content
- There are no negative access control attributes

3.2. Interaction with ebXML TRP

The ebXML Registry security involves interactions with the message layer.

In case of ebXML TRP, the following interactions are involved:

a) Authentication

The TRP has the semantics and syntax for signing the message header. The registry will use the certificate DN from the signature to authenticate the user.

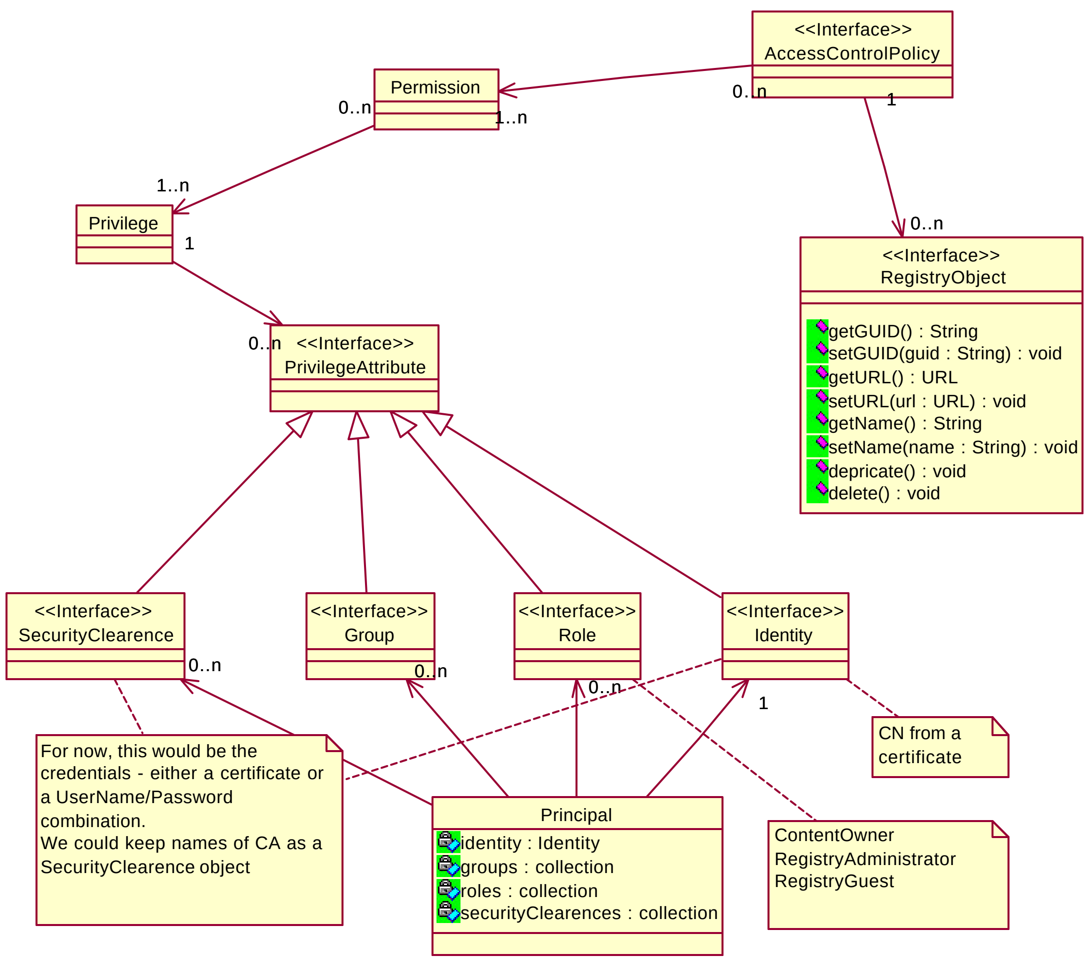
b) Integrity

The TRP has the semantics and syntax for signing the message payload. All submitted contents should be signed (as defined in TRP) and the Registry will store the signature as a part of the content. When a client requests a content, the registry will also send the signature. This way, the client can verify the integrity of the content.

3.3. Security Info Model

The security model is based on two goals – simplicity from a client’s point of view and extensibility for future enhancements.

The following figure shows the info model, which contains the security related objects. The figure is for reference only. For more detail, please refer to the Registry Information Model document.



The AccessControlPolicy is the the top-level security object. It ties together the permission object with an instance of a Registry object. The permission object also contains the methods (of the RegistryObject), which the privilege object can access.

Notes:

The actual method names are static and well known.

One permission 0bject is associated with one infoObject. However, an InforObject will be associated with many permission objects.

For example, each infoobject will be associated with three permission objects which have the attributes

{Role = RegistryAdministrator, methods = *},

(Identity = <the DN of the SO>, methods = *},

{Role = RegistryGuest, methods = “getGUID”,”getName”,getURL”}

A privilege object contains many Privilege Attributes. A Privilege Attribute can be a Security Clearence, a group, a role, or an identity. This association enables one, the flexibility to have object access control policies based on a role, an identity or a group or a securityclearence or even better all of the above !

While privileges deal with groups, roles et al, the permissions deal with the methods of an object and tie them to privileges. The permission is an “and” operation (or a cumulative) . i.e. an entity can access the method of a RegistryObject only if it has all the privileges as detailed by the privilage object.

On the other hand, the AccessPolicy is an “or” operation. If an entity has “any” of the permissions, it can perform the method as detailed by the permission object.

An Identity usually is the DN in a certificate. It could be username/password as well.

The SecurityClearence object could keep the CA names, root certificates, et al. A SecurityClearence could be the traditional operations like Read, Create, Update, and Delete.

The group object is not used for now.

The role names are ContentOwner, RegistryAdministrator, RegistryGuest.

The Principal object is an entity, which has an identity, and optionally a set of role memberships, group memberships or security clearances. The authenticator will work against a principal.

3.4. Security Processing

This section provides a blueprint for how security processing may be implemented in the registry. It is meant to be illustrative not prescriptive. Registries may choose to have different implementations as long as they support the default security roles and authorization rules described in this document.

3.4.1. Authentication

1. As soon as a message is received, the first work is the authentication. A principal object is created.
2. If the message is signed, it is verified (including the validity of the certificate) and the DN of the certificate becomes the identity of the principal. Then the Registry is searched for the principal and if found, the roles, groups and the securityclearences are filled in.
3. If the message is not signed, an empty principal is created with the role RegistryGuest. This step is for symmetry and to decouple the rest of the processing.
4. Then the message is processed for the command and the objects it will act on

3.4.2. Examine Transaction Rights on Object Request (Authorization)

For every object, the access controller will iterate thru all the AccessControlPolicy objects with the object and see if there is a chain thru the permossion objects to verify that requested method is permitted for the Principal. If any of the permission objects which the object is associated with has a common role, or identity, or group with the principal, the action is permitted.

3.4.3. Registry Bootstrap

When a registry is newly created, a default Principal object should be created with the identity of the Registry Admin’s certificate DN with a role RegistryAdmin. This way, any message signed by the Registry Admin will get all the privileges.

3.4.4. Content Submission – processing done by the Registry Client

The Registry client has to sign the contents before submission – otherwise the content will be rejected.

3.4.5. Content Submission – processing done Registry Service

1. Like any other request, the client will be first authenticated. In this case, the Principal object will get the DN from the certificate.
2. As per the request in the message, the info Object will be created.
3. The next step is to create the default permission objects
    a. If required, a permission object is created associating the RegistryObject methods with the Privilege object pointing to the RegistryAdministrator role with * as the method name
    b. An AccessControlPolicy object is created with the permission and the GUID of the new content.
    c. If a principal with the identity of the SO is not available, an identity object with the SO's DN is created
    d. A principal with this identity is created
    e. A second permission object is created associating this identity with the with * as the method name
    f. A third permission object is created associating the RegistryGuest role with the with the getName, getURL and getUID as the method names
    g. Then two more AccessControlObjects are created tying in all the permission objects with the GUID of the newly created object

3.4.6. Content Delete/Deprecate – processing done by the Registry Client

The Registry client has to sign the payload (not entire message) before submission, for authentication purposes; otherwise, the request will be rejected

3.4.7. Content Delete/Deprecate – processing done Registry Service

1. Like any other request, the client will be first authenticated. In this case, the Principal object will get the DN from the certificate. As there will be a principal with this identity in the Registry, the principal obj will get all the roles from that object
2. As per the request in the message (delete or deprecate), the appropriate method in the info Object will be accessed.
3. The access controller performs the authorization by iterating thru the permission objects associated with this object
4. As the Registry had created an AccesssControlPolicy object which has the permission object associating this identity and with the method names *, the action will be permitted.

5. Issues & Ideas

5.1. Issues

- Trust relationship between distributed registries – Not on Release 1
- Session and auth tokens exchange – Not in Release 1
    - Session based interaction
    - Sessions as short-lived certificates (?)
- Do we need a userid/password based authentication or can a certificate based authentication suffice - No
- Should we allow Object retrieval via HTTP GET?
- How to deal with expiration of a certificate associated with submitted content
- What objects are persistent and which are transient. It is hard to grasp when the security objects, like permissions or principals are created and when they go away (which can be a security issue in itself).
- Develop a CPP for this. The CPP could define the different roles and also demonstrate the security needed at each level....for example the "reader" role would not need any security on its request message, as opposed to the "document owner" role needing authentication. Then we will abstract the security interactions to different roles and provide a CPP for it.

5.2. Phase 2

- Define interface to submit custom Access Control Policies

  Identity and Role based authorization

- Registry may not be trusted to view all content
- Trust relationship between distributed registries
- Session and auth tokens exchange
    - Session based interaction
    - Sessions as short-lived certificates
- Do we need a userid/password based authentication or can a certificate based authentication suffice?
-

Copyright Statement