



---

## **Endpoint Location Service**

### **Implementation Guide**

Version 1.2 — 30 June 2009

---

**National E-Health Transition Authority Ltd**

Level 25

56 Pitt Street

Sydney, NSW, 2000

Australia.

[www.nehta.gov.au](http://www.nehta.gov.au)**Disclaimer**

NEHTA makes the information and other material (“Information”) in this document available in good faith but without any representation or warranty as to its accuracy or completeness. NEHTA cannot accept any responsibility for the consequences of any use of the Information. As the Information is of a general nature only, it is up to any person using or relying on the Information to ensure that it is accurate, complete and suitable for the circumstances of its use.

**Document Control**

This document is maintained in electronic form. The current revision of this document is located on the NEHTA Web site and is uncontrolled in printed form. It is the responsibility of the user to verify that this copy is of the latest revision.

**Copyright © 2009, NEHTA.**

This document contains information which is protected by copyright. All Rights Reserved. No part of this work may be reproduced or used in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems—without the permission of NEHTA. All copies of this document must include the copyright and other information contained on this page.

# Table of contents

<b>Table of contents</b> .....	<b>iii</b>
<b>Document information</b> .....	<b>v</b>
Change history .....	v
<b>1 Introduction</b> .....	<b>7</b>
1.1 Document purpose .....	7
1.2 Intended audience .....	7
1.3 Definitions, acronyms, abbreviations .....	7
1.4 Normative references .....	8
1.5 Overview .....	9
1.5.1 Alternate and Complementary Interfaces .....	9
1.5.2 Web Service Environments .....	9
1.5.3 ELS Deployment .....	9
1.5.4 Data Stores .....	10
<b>2 National Infrastructure</b> .....	<b>11</b>
2.1 Unique Healthcare Identifier Service .....	11
2.1.1 Identifiers .....	11
2.1.2 ELS Endpoints .....	11
2.2 NASH .....	12
2.2.1 Certification Authority .....	12
2.2.2 Certificate References .....	12
<b>3 Data Model</b> .....	<b>13</b>
3.1 Database Schema .....	13
3.1.1 Entities .....	13
3.1.2 Constant URIs .....	13
3.2 DAO or Alternative .....	13
3.3 Query for listInteractions .....	14
3.4 Standard Update Operations .....	14
3.4.1 addInteraction .....	14
3.4.2 removeInteraction .....	15
<b>4 Trust and Security</b> .....	<b>16</b>
4.1 ELS Certificates .....	16
4.1.1 ELS Associations .....	16
4.1.2 Identity Mapping Function .....	16
4.2 Lookup Operations .....	16
4.3 Update Operations .....	16
4.3.1 Updates by Owners .....	16
4.3.2 Updates by Non-owners .....	17
<b>5 Maintenance</b> .....	<b>19</b>
5.1 Extensions to the Standard Interface .....	19
5.1.1 Associating Primary Identifiers .....	19
5.1.2 Lower-level Updates .....	19
5.1.3 Bulk Interfaces .....	19
5.1.4 Non-Web service Updates .....	20
5.2 Update Responsibility .....	20
5.2.1 Owner and ELS Updates .....	20
5.2.2 Changes Affecting HPIO .....	21
5.3 Alternate Directories .....	22
5.3.1 Service Provider Directories .....	23
5.3.2 Healthcare Provider Directories & ELS .....	24
<b>6 Reliability</b> .....	<b>25</b>

---

6.1	System Down-time .....	25
6.1.1	Clustered Application Server Environment .....	25
6.1.2	Failover Databases .....	25
<b>7</b>	<b>Network Configuration .....</b>	<b>26</b>
<b>8</b>	<b>Existing Provider Directories .....</b>	<b>27</b>
	<b>Appendix A: Informative references .....</b>	<b>28</b>

# Document information

## Change history

Version	Date	Comments
1.1	2008-12-01	Release
1.2	2009-06-30	Renamed as Endpoint Location Service and updated to reflect modified data structures.

This page is intentionally left blank.

# 1 Introduction

## 1.1 Document purpose

This document provides guidance on how to design and implement an endpoint location service (ELS). Some of the issues that may arise through deployment of an ELS are also covered.

This is a non-normative document. While implementers are obliged to conform to the ELS specification, they are not required to follow the advice contained in this document.

## 1.2 Intended audience

This is a technical document.

This document is intended for:

- Enterprise architects who develop strategic plans for solutions that involve connectivity.
- Solution architects who develop solutions that implement or interact with the connectivity architecture.
- Software developers who implement the solutions designed by the solution architects.

The reader is expected to be familiar with the ELS Architecture .

## 1.3 Definitions, acronyms, abbreviations

Definitions are reproduced from [ELSA2009] with a few additions.

CA	Certification Authority—a trusted entity that establishes healthcare provider membership in the e-health community by signing the providers X.509 certificate with their own (CA) private key. The certificate containing the corresponding public key would be stored by e-health clients.
CRL	Certificate Revocation List
CRUD	Create, read, update and delete
CUD	Create, update and delete
Document	A clinical document or ancillary message, such as a notification or acknowledgement for a clinical document. Documents are usually represented in XML. Elements within such documents may contain non-XML data, e.g. formatted according to HL7.
Endpoint	A URI including network protocol and address. It provides the binding of an interface to an implementation.
ELS	Endpoint location service
FTPS	File transfer protocol over SSL (Secure sockets layer)
HPII	Healthcare Provider Identifier for an Individual
HPIO	Healthcare Provider Identifier for an Organisation

HTTP	Hypertext Transfer Protocol
IHI	Individual Healthcare Identifier
Interaction	Pattern of communication whereby a target obtains its document – corresponds to the patterns outlined in [CPIS2008].
Interaction Role	Function within an interaction. Roles are played by healthcare providers or intermediaries.
LDAP	Lightweight Directory Access Protocol – a directory predicated on X.500 standards and using TCP protocols for transport
NASH	National Authentication Service for Health—NASH will be a CA for e-health technical service providers.
OCA	Organisational Certificate Authority – an intermediate CA. An OCA certificate is signed by a root or self-signed CA certificate.
OCSP	Online Certificate Status Protocol
Service	In this document the term service generally refers to a technical service as per [IF2007]. A technical service is usually a Web service.
Service Category	Service types encompassed by a medical realm. Categories are initially expected to be document types specified by the NEHTA work packages. Each service category will be identified by a URI.
Service Interface	URI-based definition of a service offered by a role. Initially these interfaces will describe a Web service.
Service Provider	An organisation that hosts a Web service. This could be the target, source, or a third party.
SFTP	File transfer protocol using secure shell (SSH)
Source	Document suppliers / compilers. For clinical documents a source is a healthcare organisation, e.g. pathology laboratory.
Target	The final destination (intended recipient) of a document. For clinical documents a target is a healthcare provider organisation, e.g. medical clinic.
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UHI	Unique Healthcare Identifier (HPIO, HPII, or IHI)
UHI Service	Proposed National UHI Web service

## 1.4 Normative references

The following NEHTA specifications and other references, through not referenced in this text, inform certain elements of this document. At the time of publication, the editions indicated were valid. All Specification and other references are subject to revision: all users of this document are therefore

encouraged to investigate the possibility of applying the most recent edition of the specifications and other references listed below.

[IF2007] NEHTA, *Interoperability Framework v2.0*, 17 August 2007.

[CPIS2008] NEHTA, *Concepts and Patterns for Implementing Services, V2.0*, 1 December 2008.

[WSP2008] NEHTA, *Web Services Profile v3.0*, 1 December 2008.

[SA2008] NEHTA, *ELS Architecture v1.1*, 1 December 2008.

## 1.5 Overview

ELS implementations are responsible for maintaining the interaction records containing associated endpoints as described in [ELSA2009]. Some of the issues dealt with in this document are:

- Data store requirements,
- Tie-in with HPIO records,
- HPIO Identifiers,
- Certificates,
- Trust of/by clients,
- Semantics Identifiers.

Note that ELS must be implemented as a Web service. Details of how to create and deploy Web service applications are not covered.

### 1.5.1 Alternate and Complementary Interfaces

ELS instances must provide the interfaces dictated by [SA2008]. However the specification does not preclude implementing other interfaces. As outlined in 5.1, alternate mechanisms, particularly for updates, may be more convenient and easier to implement. One such approach is to use Web applications.

[SA2008] does not specify operations for all the updates which an ELS implementation will have to encompass. Therefore, many design decisions, even at the interface level, are left to the discretion of implementers. For ease of use, particularly for staff of healthcare organisations, it is highly desirable that a user-friendly GUI is provided, at least for updates.

### 1.5.2 Web Service Environments

There are several choices for building Web service applications, the most popular of which are Microsoft's .NET environment and various Java application servers. Both choices allow for integration of Web service and Web applications if this is desired.

### 1.5.3 ELS Deployment

According to [ELSR2009] the HPIO: ELS implementation relationship has a cardinality of 1:N. It is therefore possible for a single, centrally administered ELS to be associated with every HPIO. However, that would require significant resources from one provider agent while preventing others from value adding their own features. Consequently, such a solution may be unacceptable to most healthcare providers in the long term.

Other deployment scenarios include distributing several large implementations based on regions, or distributing a large number of small implementations, where each is associated with only a few (or one) organisations. In all cases it is anticipated that a HPIO record will contain an ELS address and associated

X.509 certificate(s). Those records will eventually be obtained from the UHI service.

#### **1.5.4 Data Stores**

Depending on the number of associated HPIOs, different ELS implementations will need data stores of widely varied capability. Except in cases where the backing data store is very small, a relational database will be required. This document contains suggestions for implementing the data model and base access/update statements. Actual database access will most likely be accomplished through the libraries and/or transaction managers available in the deployment environment.

## 2 National Infrastructure

### 2.1 Unique Healthcare Identifier Service

#### 2.1.1 Identifiers

ELS lookups require a URI representing a target's HPIO as input. This HPIO identifies the target healthcare provider to whom a source transmits a document. It is envisaged that sources will locate correct ELS endpoints by first consulting the UHI service.

It is likely that ELS implementations will be needed before the UHI service is fully specified and before any HPIOs are issued. If so, ELS clients will need an identifier to substitute for the HPIO in the short term. According to [ELSA2009] this identifier can be any URI. NEHTA is currently examining options for alternative identifying schemes.

For ELS deployments with a large number of associated healthcare providers, it will be beneficial if a consistent scheme were used. One such scheme would be to formulate a qualified identifier, according to [QI2008], using a Medicare Australia Registration Authority identifier (RA) in place of an HPIO. The result would be similar to:

- <http://ns.nehta.gov.au/Id/RA/1.0#5300408561>

Such a scheme would ensure uniqueness of identifiers, although it does not allow for full automation. Note that a qualified identifier corresponding to a future HPIO would be similar to:

- <http://ns.nehta.gov.au/Id/HPIO/1.0#9036034300408561>

#### 2.1.2 ELS Endpoints

Lack of a UHI service raises the issue of how to resolve ELS endpoints. The most likely short term scenario will be for cooperating source and target pairs to obtain addresses through out-of-band means (e.g. by telephone or email).

Although the ELS specifications discuss document exchange scenarios in terms of healthcare providers and clinical documents, the interfaces themselves are generic. This means it would be possible to operate an ELS to function as a directory of ELS endpoints for healthcare providers. In effect, such an ELS would replace the UHI service for the purposes of obtaining ELS endpoints.

##### 2.1.2.1 ELS Serving ELS Interactions

In order to substitute for the UHI service, an Interaction data structure would contain values similar to the example below.

Interaction:

```
Interaction: "Retrieve"  
Target: <HPIO>  
Service Category: "ELS"  
Service Interface: <ELS lookup interface>  
Service Provider: <HPIO of ELS provider>  
Service Endpoint: <ELS service endpoint>  
Qualified Cert Ref: <ref to ELS response signing cert>
```

## 2.2 NASH

### 2.2.1 Certification Authority

In the medium term, NEHTA will arrange for an OCA to issue trusted certificates associated with HPIOs. It is likely that ELS implementations will be needed before HPIOs become available or any trusted certificates have been issued. However, early implementations may choose to trust certificates issued by Medicare Australia PKI.

### 2.2.2 Certificate References

In the absence of NASH, there is no standard way of resolving certificate references. However, ELS clients need to obtain certificates to use for services returned by the ELS for the purposes of encrypting symmetric keys and checking response signatures.

To get around this problem, the qualified certificate reference data structure is flexible enough to contain the certificate itself. To accomplish this, values for the `QualifiedCertRef` attributes could be used similar to those shown below.

`useQualifier:`

```
http://ns.nehta.gov.au/Qcr/Use/PayloadTransport/KeyEnc/1.0
```

`typeQualifier:`

```
http://ns.nehta.gov.au/Qcr/Direct/PEM/1.0
```

`value:`

```
-----BEGIN CERTIFICATE-----
```

```
MIIDHjCCAtugAwIBAgIESOxRITALBgcqhkJ00AQDBQAwc jELMAkGA1UEBhMCQVUx
DDAKBgNVBAGTA1FMRDERMA8GA1UEBxMIQnJpc2JhbmUxDjAMBgNVBAoTBU5FSFRB
MRkwFwYDVQQLExBTZWZWN1cmUgTWVzc2Fnaw5nMRcwFQYDVQQDEw5TYW1wbGUU2VY
dmljZTAeFw0wODEwMDgwNjIwMTdaFw0wOTAxMDYwNjIwMTdaMHExCzAJBgNVBAYT
AkFVMQwwCgYDVQQIEwNRTEQxETAPBgNVBAcTCEJyaXNiYXN1YXN1YXN1YXN1YXN1
RUhUQTEZMBCGA1UECXMQU2VjdXJlIE1lc3NhZ2luZzEXMBUGA1UEAxMOU2FtcGxl
IFN1cnZpY2UwggG3MIIBLAYHKOZIZjgEATCCAR8CgYEA/X9TgR11Ei1S30qcLuzk
5/YRt1I870QAwX4/gLZRJm1FXUAiUftZPY1Y+r/F9bow9subVWzXgTuAHTRv8mZg
t2uZUKWkn5/obHsQIsJPu6nX/rfGG/g7V+fgqKYVDwT7g/bTxR7DAjVUE1oWkTL2
dfOuK2HXKu/yIgmZndFIaccCFQCXYFCPFSMLzLkSuYKi64QL8Fgc9QKBgQD34aCF
1ps93su8q1w2uFe5eZSvu/o66oL5V0wLPQeCZ1FZV4661F1P5nEHEIGAtEkWcSPo
TCgWE7fPCTKMyKbhPBZ6i1R8jsjgo64eK7OmdZFuo38L+ie1YvH7YnoBJDvMpPG+
qFGQiaid3+Fa5Z8GkotmXoB7VSVkAUw7/s9JKgOBhAACgYAvaKwhiCES/MuDBAm0c
8JEypfVBB17ptjtEQU42P/3ILOsRyHbG2+yNg3tteQmyMDDfm44+zFiNL4aPMWx4
/WZm0u8URfLgHkgFaM9q5Lb1jYs7SWeponYeGz5okoBzF4D3oAxXjIrMRTEt34nQ
5kZvOcmr6WDrLKBppjkrqVBA1DALBgcqhkJ00AQDBQADMAAwLQIURep/PTYiwMf
p6WjUgiuV5nrWhYCFQCPy85CQOrPDnAWM6D80MJAE4p0nQ==
```

```
-----END CERTIFICATE-----
```

## 3 Data Model

ELS data structures follow a relatively straightforward hierarchy which is only slightly more complicated to represent using an Entity-Relationship model. The previous version of ELS (Service Instance Locator 1.1) required a more involved set of data structures. The ELS specification allows for substantially simplified data relationships (See [ELSA2009].)

An entity-relationship diagram has been removed from this guide. For detailed design information pertinent to a specific implementation, see [ELSDD2009].

### 3.1 Database Schema

For a detailed table definitions, see [ELSDD2009].

#### 3.1.1 Entities

A table will be required to hold organisational identifiers. These are qualified identifiers, URIs with embedded HPIOs (or other unique identifier). Entities associated with an ELS are healthcare providers, but the table could also contain organisations that provide technical services, e.g. a *deliver* Web service. It is a requirement of [ELSA2009] that standard operations return an *ELSError* fault if an entity referenced by an input *InteractionRequest* or *Interaction* contains an identifier with no ELS association. A column of type Boolean may be employed to indicate whether the row describes an associated healthcare provider or a technical service provider. If the value of the row entry is false (or NULL) a healthcare organisation is implied, otherwise a technical provider is implied.

#### 3.1.2 Constant URIs

The ELS model consists of several constant URIs. These are:

- Interaction URIs;
- Service Interface URIs;
- Service Category URIs.

These URIs are used for identification. There are relatively few in each category, and queries will typically begin by comparing them against the input.

Certain interfaces may correspond to particular service categories, as specified by a NEHTA collaboration project. It is possible to model this relationship to provide a compatibility check of service interface to service category. It could also be used in internal queries to determine what interfaces align with a particular category.

### 3.2 DAO or Alternative

It is expected that implementers will provide or choose their own data access object (DAO) layer as a façade to the database, which is relevant to the platform and programming language of choice. The most obvious way of doing this is to use ODBC/JDBC libraries.

There are alternative means to accomplish data access from code such as object-relational mapping technologies (e.g. hibernate/nhibernate, TopLink, Kodo, etc).

Insertions, updates, and deletions may require substantial consistency checking, which is much easier to implement in a programming language like Java, rather than exclusively through stored procedures, etc. No code

examples are provided in this document, but some SQL examples are given in the following sections to provide some idea of the complexities involved.

ELS is a relatively simple specification in its own right, but implementers may choose to extend the base functionality at the expense of additional complexity. Subsequent sections outline some of the issues that could come into play for early adopters.

### 3.3 Query for listInteractions

Operation *listInteractions* returns a set of interaction data structures, which in turn contain a set of qualified certificate references. This is a tree structure which can be built from a data model by obtaining the high level interactions, then obtaining any associated certificate references.

Input to *listInteractions* is a target organisation, a set of service category URIs, and optionally, a set of service interface URIs. The starting point is to match all the interaction rows by comparing the input target, input service categories and interaction URIs, if present.

If a set of service interfaces were present, the input(s) would have to be checked against the URIs in the table containing supported service interfaces. It is expected that most queries will specify an HPIO and one category URI.

### 3.4 Standard Update Operations

There are two standard operations to modify ELS data: *addInteraction* and *removeInteraction*. Both are coarse-grained, consistent with recommended practice for distributed service interfaces, but lacking the requisite control to fully create and update *Interaction* record components.

The semantics for publish operations are not specified, so implementers have considerable flexibility. It is recommended that standard operations disallow implicit updates for atomic components. The easiest approach would be to check that URIs corresponding to the service categories and service interfaces exist in the database. If any are absent, the standard error *badParam* should be returned to the client.

Disallowing implicit insertion of the various URIs implies that there must be an alternate means for low-level updates, not necessarily a Web service. Some suggestions are given in 5.1.

According to [ELSA2009] if the organisational identifier URI does not exist, *ELSError* should be returned with the enumeration set to *unknownTargetID*, which tells the client there is no HPIO associated with this ELS.

#### 3.4.1 addInteraction

Insertion of an *Interaction* requires several checks to be made. Each of the constant URIs must be determined to be present, as well as any referenced service provider entities and certificate references. Note that an implementation could allow implicit insertion of certificate references or require them to pre-exist in the database. Since all fields have to be present in the input, implicit insertion would require additional SQL insert statements.

When the NASH architecture is finalized, it may be a reasonable expectation that the ELS checks to ensure the referenced certificates have actually been issued to the entity hosting a service. Until then, there is no standard way to accomplish this, so the ELS may simply store the certificate. It is up to the communicating parties to ensure they trust one another. If the certificate reference(s) are invalid or not trusted, there will be problems when the client attempts to use the service. If it becomes possible for the ELS instance to verify the certificate references, standard error *badParam* should be returned to the client as a fault.

### 3.4.2 removeInteraction

This operation is the inverse of `addInteraction`. It requires the same checks for consistency of input. The ELS Architecture [ELSA2009] leaves the exact semantics unspecified, so the implementer can choose the extent to which implicit deletes will be executed. There are two options.

1. Delete the *Interaction* only, i.e. no implicit deletes.
2. Delete the *Interaction* and associated *QualifiedCertRefs*.

It is important that `removeInteraction` semantics are consistent with those of `addInteraction`. For example, interactions may be added but if certificate references do not pre-exist, this would be considered an error. This is perhaps the easiest approach, since certificates may be reused for numerous services. If this approach is adopted for `addInteraction`, then `removeInteraction` should delete the *interaction* but it should not delete certificate references, even if removal of an *interaction* would leave the certificate reference(s) dangling.

# 4 Trust and Security

## 4.1 ELS Certificates

The ELS provider must obtain its own certificate which the client will use to encrypt the request and which will also be used to sign the response (a list of interactions) (See [WSP2008].)

From the client point of view, there must be a mechanism to obtain the certificate to be used with any ELS instance. If the UHI service is not available, this may be done by out-of-band means. The ELS service provider will be responsible to define some mechanism and communicate it to potential clients. Some examples are:

- Download the certificate from a Web site.
- Attach the certificate to emails to clients.
- Incorporate ELS address and corresponding certificates in the proposed national infrastructure ELS.

### 4.1.1 ELS Associations

There are no official guidelines about associating ELS services with corresponding certificates or HPIOs. These issues should be addressed in the near future. The preferred solution is for the UHI record to contain a mapping between HPIO and corresponding implementation (See 1.5.3).

If there is one central implementation, the problem of associating HPIOs will be moot since every HPIO would be associated with the same ELS. Similarly, since there would only be one certificate, for signing responses, this could be obtained quite eaELSy, e.g. from a well-known Web site.

If there is to be a distributed set of ELSs, ad-hoc mechanisms for setting and retrieving associations are less acceptable, because synchronization and consistency issues could eaELSy occur. Before the UHI service is operational, it is recommended that NEHTA take responsibility by publishing ELS associations on an external Web site.

### 4.1.2 Identity Mapping Function

It should be possible to authenticate an organisation using a function that extracts an HPIO or other identifier from a certificate presented on connection to the ELS. For example, a proposed NASH certificate profile incorporates a URI in a subject alternate name attribute. If this URI contains

## 4.2 Lookup Operations

Any entity with a certificate signed by the trusted CA(s) is allowed to perform a lookup operation (see [ELSR2009] and 2.2.1). It is a core requirement for an ELS not to restrict information to any e-health community member.

## 4.3 Update Operations

### 4.3.1 Updates by Owners

According to [ELSR2009] organisations must be able to update their own records. ELS implementations need to check request certificates to determine whether an update operation is being attempted by the logical owner of the information. To accomplish this, the HPIO (or applicable identifier) in the

interaction has to match a certificate issued to the corresponding organisation by a trusted CA.

### 4.3.2 Updates by Non-owners

In general, an organisation should not be allowed to update information on behalf of some other organisation. However, existing relationships exist between many healthcare providers and entities which administer local service directory entities. In such a relationship the directory stores service location information on behalf of more than one healthcare provider. It is likely that in some circumstances the directory provider will also host Web services to transfer clinical documents and other messages.

Where such a relationship exists, a HPIO entity should be able to delegate update operations on its ELS entries. In the absence of an official e-health authorisation strategy, an ELS instance must still allow such updates to occur. Currently, there is no standard strategy to accomplish this. Below are a few suggestions.

#### 4.3.2.1 Role Based Authorisation

Define authorisation roles (not to be confused with *Interaction Roles*) corresponding to capabilities. An example set may include:

- Owner– Corresponds to the HPIO whose ELS records may be updated. Owners should have all CUD privilege for records (to be) associated with their HPIO.
- Agent– Corresponds to the HPIO(s) that an owner authorizes to perform its standard inserts and deletions.
- Agent Insert– Corresponds to the HPIO(s) that an owner authorizes to insert interactions.
- Agent Delete– Corresponds to the HPIO(s) that an owner authorizes to delete interactions.
- Agent Update– Corresponds to the HPIO(s) that an owner authorizes to update interactions. There is no update operation in the ELS specification, so any updates would constitute extra operations implemented by an ELS provider.
- Certificate Administrator – Allowed to insert and remove qualified certificate references.
- ELS Owner – Corresponds to the ELS implementer. May be responsible for adding HPIO URIs to the ELS data set.

Unless the ELS implementer required a fine degree of authorisation granularity, there is no need for all the roles listed above. Minimally, *ELS Owner* and *Owner* would be needed.

If particular directory provider HPIOs were allowed full CUD privileges, an *Agent* role would be needed. There would have to be an operation of some kind for an *ELS Owner* or *Owner* to grant the privilege.

#### 4.3.2.2 Explicit v Implicit Roles

It is possible to define roles and associated privileges explicitly. This would require configuration files, or more likely, tables in the database containing roles, associated HPIOs, and privileges.

The *Owner* role can easily be made implicit. CUD permissions would be contingent on the X.509 certificate matching the private key used to sign the operation request.

It may not be possible for an *Agent* role to be implicit. However, if the suggested data model is used (see 3.1), it is easy to delegate CUD access with the addition of another table. The table could be named

*DELEGATED\_AGENTS* and would require two foreign keys on table entity (*TARGET, AGENT*). If an agent organisation attempted to update a record for a target organisation, this table would be checked. Again, a non-standard operation would be required for a target organisation to insert a record for its agent. In this way, support for roles would be implicit in the logic of the operations; there would be no need for tables of roles and privileges. Similarly, tables such as *DELEGATED\_INSERT\_AGENTS* or *DELEGATED\_DELETE\_AGENTS* could be created for finer granularity. Such a scheme could simplify the logic of CUD operations.

If transitive privileges are required, the simple schemes outlined above may not be sufficient. In that case defining explicit tables for roles, delegated access control and associated entities may be required.

Unauthorized Web service requests should always return the standard error fault - `notAuthorised`.

#### 4.3.2.3 Certificate Based Authorisation

Role-based permissions are probably easiest to implement when the underlying data model uses relational databases. However, there are alternatives to role-based updates. One of these is to grant CUD privileges based on certificates.

If a particular entity signs a Web service request with a private key corresponding to a supported certificate (or uses a supported certificate in a SSL handshake using mutual authentication), that entity would be granted permission to update the records for an associated target entity. For this to succeed, it would be necessary to ensure a representative of an HPIO entity contacts the ELS provider with a list of certificates authorized to update ELS records on its behalf. This could be done by:

- providing a Web service operation whereby the HPIO entity can submit certificates authorizing update operations on its behalf, or
- providing a Web form allowing an HPIO entity to submit certificates that can be used for update requests on its behalf.

Internally, these solutions rely on the ELS provider maintaining a set of certificates which are allowed to sign update operations for each HPIO. If the list is empty, only certificates issued to the organisation corresponding to the HPIO itself would be allowed to execute the operation.

Unauthorized Web service requests should always return the standard error fault `notAuthorised`. This is a coarse-grained approach, but it could be refined by maintaining separate lists for varying levels of access control.

# 5 Maintenance

## 5.1 Extensions to the Standard Interface

Standard ELS update operations are inadequate even for storing all the required associations. For example there is no operation to associate an ELS with a HPIO or any other target identifier. This means that ELS implementers are free to choose their own methods to create the associations.

### 5.1.1 Associating Primary Identifiers

Depending on how many HPIOs or other identifiers are intended to be associated with a particular ELS, operations to add and remove them may not be necessary. If there are only a few associated primary identifiers, updating a configuration file may be sufficient for the purpose. If many hundreds or thousands were anticipated, a relational database would be more appropriate.

It is up to the implementer to decide the form, if any, of operations to associate/disassociate top level identifiers. An obvious choice would be to provide a Web form for a user to complete. Of course, an ELS cannot use the certificate of some entity trying to associate its own identifier unless prior arrangements have been made to trust the certificate used to sign requests.

### 5.1.2 Lower-level Updates

Update operations of the ELS specification are limited to adding and deleting an entire interaction. The semantics of this are unspecified but the implementers need to decide on a strategy for updating individual tables.

URIs for providers, interactions, service categories, roles, and interfaces should be able to be updated directly. The most practical strategy would be to disallow implicit insertions of URIs on insertion of an interaction. If a URI contained in an input interaction of some kind cannot be located, the implementation should return the standard error fault `badParam` (or `badlyFormedMsg` if the URI is malformed). Similarly, URIs should not be implicitly removed by a delete operation.

It should be possible to add qualified certificate references independently of a standard insert request since different services may reference the same certificates. Similarly, it should be possible to add services independently, since different interactions can refer to the same service. However, if services are not present in the database, they should be added with a new interaction.

When an interaction is added and its referenced services and/or certificates are already present in the database, the insert operation boils down to linking the pre-existing elements through updates to tables *interact*, *interact\_svc* and *svc\_qcr*.

### 5.1.3 Bulk Interfaces

If a provider directory has a large number of existing interactions and wants to transfer those to an ELS, a bulk interface of some kind may be in order. Usually, there are two methods of bulk loading a database.

1. Source a SQL file containing insert and update statements, possibly constructed by adapting a dump of the first database. This could entail high overhead if the schema of the first database is significantly different from the schema of the target database.
2. Load the database tables from comma separated value (CSV) files. It is probably necessary to write a script (e.g. sh, bash, Python, Perl, PHP) to

coalesce all the CSV files around the defined database schema and sequence calls to the database's load utilities.

If adopting a schema similar to that suggested in this document, both these approaches may be difficult due to the reliance on foreign keys. It may be advisable to create stored procedures to link individual records through the intersection tables to be repeatedly called by the scripts.

#### 5.1.4 Non-Web service Updates

It is not necessary that non-standard updates are Web services and it is envisaged that the most maintenance will be carried out by means of a custom application or, more likely, a browser-based interface. If Web services are not being used, the Web service profile does not apply and there must be an alternate means of securing messages.

In using a Web server application, communications should be secured using HTTPS with mutual authentication. Mutual authentication allows the server to authenticate the client (user) through their X.509 certificate. The certificate itself could be used for access decisions if certificate-based authorisation was being employed (see 4.3.2.3).

If role-based authorisation was employed, as is more likely, the entity would be resolved using the certificate and the specific access checked using the appropriate *DELEGATED\_\** table (see 4.3.2.2). (There is no need to use the schemes described in 4.3.2.2 - permissions may be stored in specific privilege tables or configured through access control lists).

Alternatively, the entity could be authenticated at the application level over a HTTPS session, most likely through username and password. However, for an ELS associated with a large number of HPIOs there would be the additional overhead of maintaining identifying information, either in database tables or in a separate repository such as an LDAP directory. For a small ELS it would not make much difference which approach was taken. For a large ELS, the development, maintenance and storage overheads could become quite high.

## 5.2 Update Responsibility

If a source entity cannot locate a target's supported services through its ELS there is a possibility that one or more patients could be temporarily deprived of a necessary healthcare service. It is therefore important that all provider ELS instances remain up to date. Typically, service owners are responsible for ensuring a smooth maintenance cycle.

### 5.2.1 Owner and ELS Updates

An organisation is referred to as a *service owner* when it hosts some service for document exchange or such a service is hosted on its behalf. It is possible for a service to be redeployed as a fresh implementation without impacting its metadata, i.e. details stored in the ELS. Whenever any detail of a service is modified, including its service interface, supported category, role, endpoint, or certificates used in service invocation, it is the responsibility of the owner organisation to ensure the ELS is modified. Similarly, it is the responsibility of the owner to add or remove interactions when required.

If all services are outsourced, the outsourcing entity may find it more expedient to perform ELS updates. In that case, the owner must at least ensure that the outsourcer does in fact keep the ELS up to date. This implies a situation similar to that discussed in 4.3.2. Some strategy is required to deal with this situation by the ELS implementation. The service owner would need some process whereby it could delegate its maintenance responsibilities. At some point in the future, maintenance privilege may have to be removed from the outsourcer, for example because the owner enters into a different contract for services.

## 5.2.2 Changes Affecting HPIO

A healthcare provider organisation will be allotted a single HPIO, by which it will be identified in the E-health community. Some organisational changes will impact this identifier and its associated ELS entries, while others will have no effect. Listed below are some common circumstances and maintenance expectations.

Circumstance	Action
Healthcare provider ceases to practice/trade.	Unless another organisation takes over the old operations, associated ELS entries must be removed. Since the owner has ceased to do business, it will be incumbent on the ELS provider to delete the records. The HPIO must also be removed from the HPIO service. When this happens, reference to the organisation's ELS will also be deleted.
Provider is split into two (or more) entities.	Details could vary here. If the original HPIO continues to be associated with a provider that maintains its old operations no ELS updates may be necessary. The new organisation will set up its own ELS and add interactions as necessary. If the old services are to be divided between the new entities, the original entity must remove records that have become invalid. It is the new entity's responsibility to add its own ELS records. If existing services are to be maintained under the new entity's HPIO, there is no standard operation to accomplish the switch. The most suitable course of action would be to contact the ELS administrator (possibly the original organisation) to effect the change. In addition, a new HPIO record, including ELS details, must be made available from the HPIO service.
Provider is merged with another entity.	Assuming that both organisations have existing services, it is clearly the responsibility of the new provider entity to ensure the services of the old provider become associated with its HPIO. Because there is no standard operation (or authorisation strategy) to simply switch services to a new provider, it is probably a good idea to have the ELS administrator carry this out in the period leading up to the merge date. If records of the entity to be merged (whose HPIO is to be deleted) need to be concurrently maintained until the actual merge date, the updates should be scheduled to take place at the last moment. If required, the two entities could continue to exist at the HPIO level for some time but refer to a common service. This would require different <i>service</i> entries to refer to the same endpoint.
Service provider(s) are changed.	The service owner must remove the old records and add or arrange for the new provider to add the new ones.

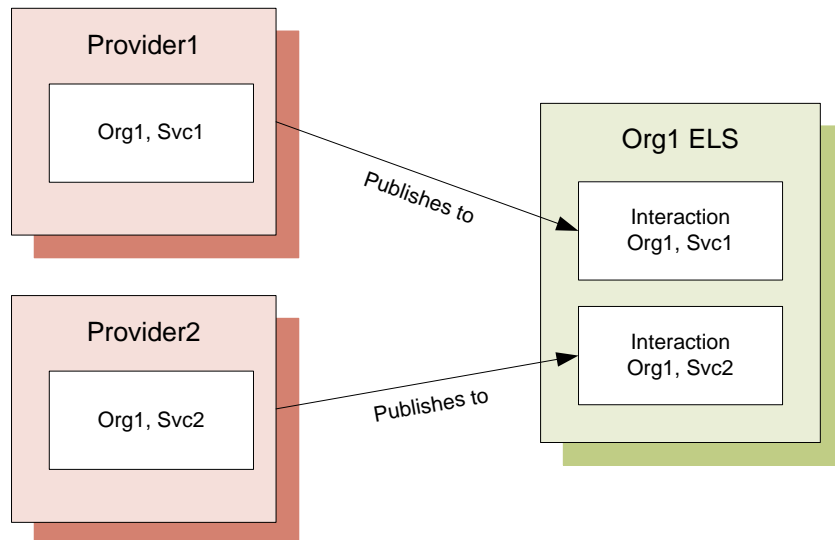
Provider changes its ELS.	Provider is responsible for duplicating the entries for the new ELS and updating its HPIO record. If the old ELS instance is not being decommissioned, the provider should also be responsible for removing its records and arranging for its HPIO to be disassociated.
Core business changes.	This is unlikely, but would seem to entail old services being removed and new services being added. If the ELS instance changes the HPIO record must reflect that.
Organisation incorporates.	HPIO may change, in which case the new HPIO record must be arranged and the old one removed. Assuming the same ELS is used, the records must be switched over to the new HPIO. There is no standard operation for that, so it would be best to have the ELS administrator carry it out. If the HPIO remains unchanged, it is unlikely any change is required.
Acquisition occurs.	Similar to the situation where existing providers are merged. If the acquired organisation has no ELS and/or existing services, no action would be required in the short term.
X.509 certificate associated with a service is superseded, e.g. because the old certificate expires or is revoked.	As usual, it is the owner's responsibility to perform the update, in this case of the certificate references. If the service in question is provided by a third party, it may be better for the service provider to do the task. That would only be possible if the owner had delegated the necessary update privileges to the provider. There is no standard operation to change certificate references. An alternative interface (e.g. Web application) could be used if provided. Alternatively, out of band arrangements could be made with the ELS implementer. An almost standard way would be to remove the interaction entirely, remove the certificate reference, insert a new certificate reference (e.g. through a Web application) and finally re-insert the interaction and service(s). The number of steps this requires underlines the desirability of extending the update mechanisms.

### 5.3 Alternate Directories

Some directories may be maintained by service provider organisations that host document exchange services on behalf of healthcare provider clients. Some directories may be maintained to allow healthcare organisations to list the services of other healthcare providers to whom documents are to be sent.

ELS is organized from the perspective of the healthcare organisations that actually own the services. Responsibility for advertising and maintaining service endpoints also resides with those healthcare entities. However, an ELS implementation can co-exist with the other directory types.

### 5.3.1 Service Provider Directories

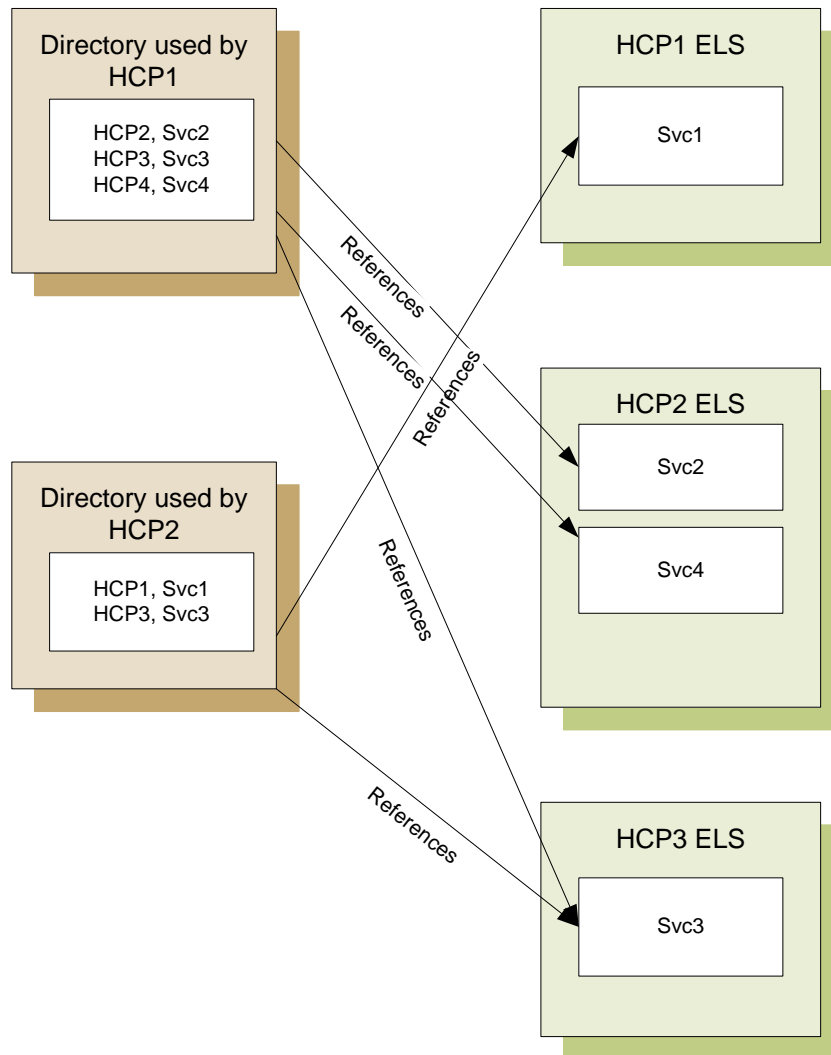


**Figure 1: Service Provider Directories & ELS**

In Figure 1 two directories are shown representing service providers called *Provider1* and *Provider2*. In this case healthcare provider *Org1* is a client of both service providers, meaning that each provider maintains a service on behalf of *Org1*. However *Org1* may only have one associated ELS, and both services should appear there. (The other directories may or may not be accessible by the healthcare community.)

It is up to *Org1* to arrange for each service endpoint to be published in an ELS interaction. Probably the best way to accomplish that is for *Org1* to delegate update permissions to both *Provider1* and *Provider2* in its ELS instance.

### 5.3.2 Healthcare Provider Directories & ELS



**Figure 2: Healthcare Provider Directories & ELS**

In Figure 2 there are two non-ELS directories. These directories are used by healthcare organisations designated as *HCP1* and *HCP2*. These directories either duplicate or reference ELS entries for healthcare providers *HCP1*, *HCP2*, *HCP3* and *HCP4*.

Traditional directory use is represented here. Senders use the directory to determine the addresses of recipients. Suppose *HCP1* wants to send a document to *HCP4*. It consults its directory and discovers it should use service *Svc4*.

As the figure indicates, service endpoints may have to be duplicated in more than one directory. This inevitably leads to maintenance problems. Therefore, it is better if such directories contain references to ELS interactions rather than copies of the endpoints. If that is not possible, frequent trawling of all applicable ELS instances is recommended.

# 6 Reliability

## 6.1 System Down-time

Like all systems, it will be necessary to bring down the ELS instance from time to time for server maintenance, software updates, etc. Despite care in administering application and database servers, etc., it is likely that system degradation or suspensions will occasionally take place, necessitating a system reset. Since there is only one ELS instance per healthcare provider, downtime implies a potential to impede or delay timely provision of healthcare services. It is therefore recommended to determine a strategy for mitigating down-time.

### 6.1.1 Clustered Application Server Environment

In a clustered environment there are two or more application servers running the ELS application, generally on different hosts. Typically both would connect to the same database. Each server in the cluster is delivered a request through a pass-through proxy. In normal mode, the proxy would route requests equally among the servers.

While the use of a configurable proxy is advisable, other solutions may be acceptable, e.g. using DNS round robin.

#### 6.1.1.1 Rolling Maintenance Strategy

When the ELS application requires updating, the proxy would be temporarily configured to discontinue routing to one of the servers. After the appropriate maintenance is performed, e.g. a new version of the ELS service is deployed, the server re-enters the cluster and the proxy is set to direct fresh requests to the new service. The process is repeated for every server in the cluster. Since an ELS is not a session-based application, there should be no client-side issues in bringing down an instance. There should be no appreciable difference in response times if the absent server's load can be substantially carried by a different instance.

### 6.1.2 Failover Databases

A running ELS instance is little use if it cannot rely on its data stores. For ELSs associated with a large number of HPIOs it may be beneficial to select a database with failover capabilities. For ELSs associated with only a few HPIOs, the data store may be implemented in files, in which case only corruption of the file system (a rare occurrence) would cause a problem.

# 7 Network Configuration

Similar considerations apply to those documented in the Security section of [CIG2008]. See that document for more information.

## 8 Existing Provider Directories

As outlined in the ELS Architecture [SA2008], certain technical service providers have built directories for healthcare clients that have similar capabilities to an ELS. They list services and endpoints for healthcare providers and in addition may coordinate message exchange. Messaging often takes the form of secure email, where documents are exchanged as S/MIME attachments.

Some of these provider organisations may wish to extend their capabilities by also implementing an ELS. However, if the provider directory coordinates document exchange in one particular category, there could be issues if the healthcare client participates in exchanges for other categories. This is because a healthcare provider may only be associated with one ELS, whereas there is no such restriction for third-party directories.

If a healthcare provider is referenced by more than one directory and both directory providers wish to implement an ELS, there would have to be negotiation as to which ELS the healthcare provider would be associated with. The decision would likely be based on business considerations. Association with more than one directory can continue as before, but lack of consistency may be the outcome of several agencies maintaining related data.

Directory providers must ensure that any services they refer to are also referenced by the ELS. It may be necessary for the service owners to register their directory provider organisations as trusted entities in their ELS (see 4.3.2) if that feature is supported.

# Appendix A: Informative references

- [CIG2008] NEHTA, *Connectivity: Implementation Guide v1.0*, 1 December 2008.
- [EIIWSJAXWS] NEHTA, *Example Implementation of Interoperable Web Services: JAX-WS v2.0*, 1 December 2008.
- [EIIWSWCF] NEHTA, *Example Implementation of Interoperable Web Services: WCF v2.0*, 1 December 2008.
- [EIIWSWSE] NEHTA, *Example Implementation of Interoperable Web Services: WSE v2.0*, 1 December 2008.
- [IF2007] NEHTA, *Interoperability Framework v2.0*, 17 August 2007.
- [ELSR2009] NEHTA, *ELS Requirements v1.2*, 30 June 2009.
- [ELSA2009] NEHTA, *ELS Architecture v1.2*, 30 June 2009.
- [QI2008] NEHTA, *Qualified Identifiers v1.0*, 1 December 2008.
- [ELSDD2009] NEHTA, *ELS Detailed Design v1.0*, 30 June 2009.