nehta

Endpoint Location Service

Technical Service Specification

Version 1.3 — 15 November 2010

National E-Health Transition Authority Ltd
Level 25
56 Pitt Street
Sydney, NSW, 2000
Australia.
www.nehta.gov.au

Table of contents

Document information

Change history

Version	Date	Comments
1.3	2010-11-15	First release

1 Introduction

1.1 Background

In a service-oriented e-health environment, tasks are performed by service invokers that invoke operations on service instances. For example, a *client program* making SOAP Web services calls on a *target service instance*. The national e-health environment uses service invocations to facilitate communications between different healthcare organisations.

In a national e-health environment *client programs* in one organisation need to know how to invoke the *target service instances* provided by other organisations. There is a scalability problem, since there are many *target service owners* and many *client programs*. There is also a maintenance problem, since the set of *target service instances* changes over time. It is not practical to inform every *client program* about every *target service instance*. The Endpoint Location Service provides a solution to this problem.

The Endpoint Location Service is a mechanism to allow *client programs* to get the technical information needed to invoke the *target service instance*. This can be done when the *target service instance* is needed, which addresses the scalability and maintenance problems.

1.2 Purpose

This is the technical service specification for the Endpoint Location Service (ELS) using a SOAP Web services interface.

1.3 Scope

This document contains conformance points for *ELS service instances*, *client programs* and *management programs*.

This document must be read in conjunction with the *Endpoint Location Service: WSDL and XML Schema files* [ELSWXS2010]. This document does not duplicate all the information from the WSDL and XML Schema files, so it must be read in conjunction with those files.

As a service interface specification, this document only defines behaviour and does not define how those behaviours have to be implemented.

This document does not cover how ELS is used, which is documented in the *Endpoint Location Service: Solution Design* [ELSSD2010].

1.4 Intended Audience

This document is intended for:

- Developers who are implementing software related to ELS.
- Testers who need to need to test software related to ELS.

1.5 References

[ATS5820—2010] Standards Australia, *ATS 5820—2010 E-Health Web Services Profiles*, Technical Specification, 5 March 2010.

[ELSSD2010] NEHTA, *Endpoint Location Service: Solution Design v1.3*, 15 November 2010.

[ELSWX2010] NEHTA, *Endpoint Location Service: WSDL and XML Schema files v1.3*, 15 November 2010.

[QCR2010] NEHTA, *Qualified Certificate Reference v1.2*, 30 June 2010.

[QI2010] NEHTA, *Qualified Identifiers v2.0*, 30 June 2010.

[RFC2119] IETF, *RFC 2119: Keywords for use in RFCs to Indicate Requirement Levels*, S. Bradner, March 1997, http://ietf.org/rfc/rfc2119.txt

1.6 Definitions

These are terms from the *Endpoint Location Service: Solution Design* [ELSSD2010]:

Certificate use
: Value that is associated with semantics about how a certificate is used when invoking a service interface.

Client program
: Service invoker that wishes to invoke a *target service instance* of a particular *target service owner*. It uses an *ELS service instance* to obtain an *interaction record* that it uses to invoke a *target service instance*.

ELS service instance
: Service instance that makes *interaction records* available to *client programs* and they are maintained by a *management program*.

ELS service operator
: Organisation that is responsible for the technical operation of the *ELS service instance*.

ELS service owner
: Organisation that is responsible for the *ELS service instance*.

Interaction record
: Technical information needed to invoke a *target service instance*.

Management program
: Service invoker that is used by the target service owner (or its delegate) to maintain the *ELS service instance*.

Service category
: Value that is associated with semantics about the business aspects of a service interface.

Service interface
: Value that is associated with semantics about the technical aspects of a service interface.

Target service instance
: Service instance that the *client program* wishes to invoke.

Target service operator
: Organisation that is responsible for the technical operation of the *target service instance*.

Target service owner
: Organisation that is responsible for the *target service instance* and is the organisation that the *client program* wishes to contact.

These are additional terms used in this document:

Current set
: The set of valid *interaction records* associated with an *ELS service instance*.

1.7 Acronyms

ELS	Endpoint Location Service
HTTP	HyperText Transfer Protocol
LDAP	Lightweight Directory Access Protocol
PEM	Privacy Enhanced Mail
TLS	Transport Level Security
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

1.8 Overview

The ELS datatypes are documented in chapter 2. These datatypes are used in the specification of the lookup interface and publish interface.

The lookup interface is documented in chapter 3. This interface is intended to be used by *client programs* to obtain *interaction records* from an *ELS service instance*.

The publish interface is documented in chapter 4. This interface is intended to be used by *management programs* (operated by *target service owners* or their delegates) to add and remove *interaction records* from an *ELS service instance*.

1.9 Conformance

Compliant implementations are required to implement the lookup interface, as defined by chapter 3.

The publish interface, as defined by chapter 4, is optional.

The keywords **SHALL**, **SHALL NOT**, **SHOULD**, **SHOULD NOT**, and **MAY** in this document are to be interpreted as described in IETF's RFC 2119 [RFC2119].

2 XSD: ELS DataTypes

2.1 Introduction

The ELS datatype XML Schema defines the datatypes used by the ELS Lookup interface and the ELS Publish interface.

2.2 Namespace

The ELS datatypes are defined in the following XML namespace:

`http://ns.electronichealth.net.au/els/xsd/DataTypes/2010`

2.3 ComplexTypes

There are three complexTypes defined in the ELS datatype XML Schema:

- `ElsCertRefType`;
- `InteractionType`; and
- `InteractionRequestType`.

2.3.1 ElsCertRefType

The `ElsCertRefType` is used to represent a single certificate in the `InteractionType` (section 2.3.2). It supplements a *qualified certificate reference* with an element that indicates what the certificate is to be used for by the *client program*.

It contains two elements:

- `useQualifier`

  A value that indicates the *certificate use* for the associated certificate as defined in [ELSSD2010].

  These are constants using the URI syntax.

- `qcr:qualifiedCertRef`

  Contains a *qualified certificate reference*. This datatype is defined in [QCR2010] and essentially consists of a qualifier and certificate value (the qualifier is a URI that indicates how to interpret the certificate value, and the certificate value can be a HTTP URL, LDAP URL, or the certificate in PEM format).

Note: Uniform Resource Identifiers (URI) are used in the datatypes of ELS to represent constant values. Instead of using string constants, the use of URIs allows these values to be defined independently while ensuring that the values are globally unique. The most common type of URI is the Uniform Resource Locator (URL), but used in this context they are treated as a unique identifier and not the address of a resource.

2.3.1.1 Equality

Definition of equality: two `ElsCertRefType` elements are equal if their `useQualifier` are equal and their `qcr:qualifiedCertRef` are equal.

2.3.2 InteractionType

The `InteractionType` is used to represent an *interaction record*, which contains the technical information that a service invoker needs to invoke an operation on a service instance.

It contains:

- `target`

  Identifier for the *target service owner*.

  This is represented as a *qualified identifier*. Qualified identifiers are defined in [QI2010], and are essentially identifiers represented as URIs to make them globally unique.

- `serviceCategory`

  A value indicating the *service category* as defined in [ELSSD2010].

  These are constants using the URI syntax.

- `serviceInterface`

  A value indicating the *service interface* as defined in [ELSSD2010].

  These are constants using the URI syntax.

- `serviceEndpoint`

  Address for invoking the *target service instance*.

  The datatype is *anyURI*, but it is expected that its value will be a URL.

  The ELS is designed to support *target service instances* that use SOAP Web services. Therefore the `serviceEndpoint` will usually be a HTTP or HTTPS URL. Other types of services can be supported, as long as their address can be represented as a URI.

- `serviceProvider`

  Identity for the *target service operator*.

  If the target service operator is the same as the target service owner, this element will have the same value as the value of the `target`.

  These are represented as a *qualified identifier* [QI2010].

- `certRef`

  Zero or more elements of type `ElsCertRefType` (section 2.3.1).

  The order of these elements (if present) is not significant.

2.3.2.1 Equality

Definition of equality: two `InteractionType` elements are equal if all values of `target`, `serviceCategory`, `serviceInterface` and `serviceEndpoint` from one is equal to the corresponding element in the other.

2.3.2.2 Service provider

The *target service operator* does not play a formal role in the use of ELS.

The `serviceProvider` element is in the `InteractionType` for informational purposes only. There is no requirement for the *client program* to use this value.

2.3.2.3 Certificate references

The type of service interface being referenced by the interaction determines the number of `certRef` elements that must be present in the *interaction*

record. It also defines the values for the `useQualifier` in those `certRef` elements. This is because the number of certificates required, and where they are used, depends on the particular operation being invoked.

2.3.3 InteractionRequestType

The `InteractionRequestType` is used to represent the matching rules for the *interaction records* that are returned from a `listInteractions` operation (section 3.2.1). It is sent by the *client program* to the *ELS service instance* when invoking the `listInteractions` operation.

It contains:

- `target`

  Identity of the *target service owner*.

  This is represented as a *qualified identifier* [QI2010].

- `serviceCategory`

  One or more *service categories*.

  These are represented as URIs.

- `serviceInterface`

  Zero or more *service interfaces*.

  These are represented as URIs.

2.3.3.1 Matching

Definition of match: an *interaction record* (an instance of the `InteractionType`) matches a request (an instance of the `InteractionRequestType`) element if:

- The `target` elements are equal;
- The `serviceCategory` element in the *interaction record* equals at least one `serviceCategory` element in the request; and
- If there are one or more `serviceInterface` elements in the request, then the `serviceInterface` in the *interaction record* equals at least one of them. But if there are no `serviceInterface` elements in the request then any `serviceInterface` in the *interaction record* is considered to be equal.

Editorial note: A suggestion to only allow one or more serviceInterface values is being considered for a future version of this specification. That is, to remove the option of providing zero serviceInterface values in the InteractionRequestType.

This is because there is no real need for client programs to list interaction records that it does not support. In practical implementations, a client program will only request the serviceInterface values that it knows it supports.

The “list all” function is useful for maintenance, and could be added as an additional operation to the Publish interface (and enhanced to list all available service categories as well as just listing all available service interfaces). This will be more logical, because the Lookup interface is designed for use by the client program; whereas the Publish interface is designed for use by the target service owner.

Note: the ability to list all available service interfaces is an extra feature of this ELS Technical Service Specification. It is not required by the ELS Solution Design.

3 WSDL: Lookup

3.1 Introduction

The lookup interface is provided by *ELS service instances* and invoked by *client programs*.

3.1.1 WSDL

3.1.1.1 Conformance points for ELS service instances

ELS 1 *ELS service instances* **SHALL** be a service provider for the WSDLs with the namespace of http://ns.electronichealth.net.au/els/svc/Lookup/2010 from [ELSWXS2010].

3.1.1.2 Conformance points for client programs

ELS 2 *Client programs* **SHALL** implement a service invoker for the WSDLs with the namespace of http://ns.electronichealth.net.au/els/svc/Lookup/2010 from [ELSWXS2010].

3.2 Operations

The lookup interface defines two operations:

- `listInteractions` and
- `validateInteraction`.

3.2.1 listInteractions

3.2.1.1 Description

The `listInteractions` operation is used by *client programs* to list *interaction records* from an *ELS service instance*.

The input contains:

- An instance of the `InteractionRequestType` (section 2.3.3).

The output contains:

- A list of zero or more `InteractionType` (section 2.3.2) which matches the input.

3.2.1.2 Conformance points for ELS service instances[1]

ELS 3 An *ELS service instance* **SHALL** implement the `listInteractions` operation complying with the *Web Services Base Profile* of [ATS5820—2010].

ELS 4 An *ELS service instance* **SHALL** implement the `listInteractions` operation complying with the *TLS Security Profile* of [ATS5820—2010].

ELS 5 If the requested *target service owner* is not registered with it, an *ELS service implementation* **SHALL** respond with a `lookupError` SOAP fault containing `unknownTargetId`.

[1] The term “ELS service instance” refers to a service instance (a deployment of an implementation). These conformance points can also be applied to “service implementations”, but the term “ELS service instance” will be used to avoid introducing “ELS service implementation” as a new term.

ELS 6 If the requested *target service owner* is registered with it, an *ELS service instance* shall respond with a list of *interaction records* that:

- **SHALL** include all *interaction records* that has ever been successfully added to the *current set* and matches the `interactionRequest` element, but excluding all *interaction records* where the most recent invocation of `validateInteraction` for that *interaction record* had returned an `isValid` status of false and it has not been successfully added to the *current set* after that invocation of `validateInteraction`.
- **SHOULD** exclude *interaction records* which are currently not in the *current set*.

3.2.1.3 Conformance points for client programs

ELS 7 *Client programs* **SHALL** invoke the `listInteractions` operation complying with the *Web Services Base Profile* of [ATS5820—2010].

ELS 8 *Client programs* **SHALL** invoke the `listInteractions` operation complying with the *TLS Security Profile* of [ATS5820—2010].

3.2.1.4 Notes (non-normative)

3.2.1.4.1 Match

The matching of an *interaction record* to an interaction request is defined in section 2.3.3.1.

3.2.1.4.2 Current set

The concept of the *current set* is defined in section 4.2.1.4.2.

3.2.1.4.3 Valid vs stale

Invocation records are either valid or stale. The term “stale” is used since the most common reason for that state is due to a stale cache entry.

3.2.1.4.4 Result list of interaction records

The conformance point for the list has been precisely defined to allow for caching and proxy implementations of *ELS service instances*.

The mandatory clause does not make reference to the current contents of the *current set*—only the optional clause does. The list returned will definitely contain all the *interaction records* in the *current set* (subject to matching the `interactionRequest`). But it can also contain additional stale *interaction records* that are not in the *current set* (but still match the `interactionRequest`).

The `validateInteraction` operation is used to determine whether an *interaction record* is stale or valid. The *validateInteraction* operation also prevents a stale *interaction record* from being returned in subsequent invocations of `listInteractions` (as long as they have not been subsequently added back into the *current set*).

A simple implementation of an *ELS service instance* can just return all the *interaction records* from the *current set* (subject to matching the `interactionRequest`).

A more complex implementation (such as a caching or proxy implementation) has the option to additionally return stale *interaction records*. These more complex implementations usually do so to improve efficiency, by avoiding expensive validation checking unless it is explicitly required.

3.2.1.4.5 Zero interaction records in result

When the `listInteractions` operation returns zero *interaction records*, this indicates that the *target service owner* is registered with that *ELS service instance*, but there are no *interaction records* that match the request.

3.2.1.4.6 Ordering

The conformance points imply that the order of the `serviceCategory` values or the order of the `serviceInterface` values is not significant. Also, duplicates in those lists are not significant and are not treated as incorrect.

3.2.1.4.7 unknownTargetId

A SOAP fault with `unknownTargetId` indicates that the *client program* has invoked the wrong *ELS service instance*, since the *target service owner* is not registered with it.

3.2.2 validateInteraction

3.2.2.1 Description

The `validateInteraction` operation is used by *client programs* to determine whether an *interaction record* is valid or stale. If it is stale, it also prevents it from being returned in subsequent invocations of the list interaction operations.

The input contains:

- One *interaction record*—the *interaction record* to validate.

The output contains:

- An `isValid` status: true or false.

3.2.2.2 Compliance points for ELS service instances

ELS 9 *ELS service instances* **SHALL** implement the `validateInteraction` operation complying with the *Web Services Base Profile* of [ATS5820—2010].

ELS 10 *ELS service instances* **SHALL** implement the `validateInteraction` operation complying with the *TLS Security Profile* of [ATS5820—2010].

ELS 11 If the `target` in the request is not registered with it, an *ELS service instance* **SHALL** respond with a `lookupError` SOAP fault containing `unknownTargetId`.

ELS 12 If the target in the request is registered with it, an *ELS service instance* **SHALL** respond with an `isValid` status of:

- True if the *interaction record* in the request is in the *current set*; and
- False otherwise.

3.2.2.3 Compliance points for client programs

ELS 13 *Client programs* **SHALL** invoke the `validateInteraction` operation complying with the *Web Services Base Profile* of [ATS5820—2010].

ELS 14 *Client programs* **SHALL** invoke the `validateInteraction` operation complying with the *TLS Security Profile* of [ATS5820—2010].

3.2.2.4 Notes (non-normative)

3.2.2.4.1 Design goals

This operation was introduced to allow `listInteractions` to be efficiently implemented as an *ELS service instance* proxy or cache. If

`listInteractions` was designed to guarantee its results were valid, any performance benefits from using a proxy or cache will be lost, since it will have to check every interaction record before returning it. Instead, `listInteractions` was given the option to return stale results, and this `validateInteraction` operation was then necessary to handle them.

This `validateInteraction` operation only returns a Boolean status, instead of being an operation that also returned a valid *interaction record*. Instead, if the *client program* discovers the *interaction record* was stale, it will have to make another invocation of `listInteractions` to attempt to get a valid *interaction record* (in very rare situations, that subsequent `listInteractions` could also return another stale *interaction record*). If there was an operation that always returned valid *interaction records*, programmers could mistakenly use it all the time (accidently or intentionally) instead of using the more efficient `listInteractions`, and therefore eliminating any efficiency gains a caching or proxying *ELS service instance* might have had.

3.2.2.4.2 Common mistakes

A *client program* needs to be carefully implemented to avoid misinterpreting the results from the `validateInteraction` operation and/or errors when using an *interaction record*. For example, here are some possible incorrect assumptions:

- It is wrong to assume that if `validateInteraction` returns an `isValid` of false then the next invocation to `listInteractions` will not have that same *interaction record* in the list returned. It is possible that the *interaction record* was not in the *current set* when `validateInteraction` was invoked, but was then added to the *current set* before the invocation of `listInteractions`.
- It is wrong to assume that if an *interaction record* was returned from `listInteractions` then `validateInteraction` will return true for it. It might return `isValid` of false because it was subsequently removed from the *current set* (removed after the invocation of `listInteractions` but before the invocation of `validateInteraction`).

  The *ELS service instance* could have been returned that *interaction record* from its internal cache, so it was already stale to begin with. This is permitted, because of how `listInteraction` is defined.
- It is wrong to assume that if invoking the *target service instance* using the information in the *interaction record* fails, then the *interaction record* is stale. The *interaction record* might be correct, but the *target service instance* was not functioning properly.
- If there are multiple ELS service instances, it is wrong to assume that if `validateInteraction` returns `isValid` of true then the *interaction record* is valid. The *target service owner* might have changed to a different *ELS service instanc*e without updating the old one.

A *client program* needs to also be careful not to get into infinite loops. For example, here are some possible infinite loop scenarios:

- Attempting to re-invoke the *target service instance* indicated by the *interaction record* (when it keeps failing) without sometime validating the *interaction record*. The interaction record could be valid.

3.2.2.4.3 unknownTargetId

A SOAP fault with `unknownTargetId` indicates that the *client program* has invoked the wrong *ELS service instance*, since the *target service owner* is not registered with it.

4 WSDL: Publish

4.1 Introduction

The publish interface is provided by *ELS service instances* and invoked by *management programs* (usually by *target service owners* or their delegate).

4.1.1 WSDL

4.1.1.1 Conformance points for ELS service instances

ELS 15 *ELS service instances* **SHALL** be a service provider for the WSDLs with the namespace of http://ns.electronichealth.net.au/els/svc/Publish/2010 from [ELSWXS2010].

4.1.1.2 Conformance points for management programs

ELS 16 *Management programs* **SHALL** be a service invoker for the WSDLs with the namespace of http://ns.electronichealth.net.au/els/svc/Publish/2010 from [ELSWXS2010].

4.2 Operations

The publish interface defines two operations:

- `addInteraction` and
- `removeInteraction`.

4.2.1 addInteraction

4.2.1.1 Description

The `addInteraction` operation is used by *management programs* to add an *interaction record* to the *current set*.

The input contains:

- Exactly one *interaction record*.

The output contains:

- An enumerated type of `ELSPublishReturnCode`.

4.2.1.2 Conformance points for ELS service instances

ELS 17 An *ELS service instance* **SHALL** implement the `addInteraction` operation complying with the *Web Services Base Profile* of [ATS5820—2010].

ELS 18 An *ELS service instance* **SHALL** implement the `addInteraction` operation complying with the *TLS Security Profile* of [ATS5820—2010].

ELS 19 If the `target` is not registered, the *ELS service instance* **SHALL** send back a `publishError` SOAP fault with `unknownTargetId`.

ELS 20 If the `target` is registered and the *interaction record* is in the *current set*, the *ELS service instance* **SHALL** respond with a `returnCode` of `duplicate`.

ELS 21 If the `target` is registered and the *interaction record* is not in the *current set*, the *ELS service instance* **SHALL** add it to that *current set* and respond with a `returnCode` of `ok`.

4.2.1.3 Conformance points for management programs

ELS 22 *Management programs* **SHALL** invoke the `addInteraction` operation complying with the *Web Services Base Profile* of [ATS5820—2010].

ELS 23 *Management programs* **SHALL** invoke the `addInteraction` operation complying with the *TLS Security Profile* of [ATS5820—2010].

4.2.1.4 Notes (non-normative)

4.2.1.4.1 Equality

The equality of *interaction records* is defined in section 2.3.2.1.

4.2.1.4.2 Current set

The concept of a *current set* is introduced to describe the state and behaviour of *ELS service instances*.

Each *ELS service instance* has exactly one *current set* associated with it. The *current set* contains a set (i.e. unordered collection with no duplicates) of *interaction records*.

When an `addInteraction` operation succeeds, the *interaction record* is added to the *current set*.

When a `removeInteraction` operation succeeds (section 4.2.2), the *interaction record* is removed from the *current set*.

This specification does not specify the behaviour of the *current set* when target service owners are registered or unregistered from an ELS service instance.

4.2.1.4.3 Duplicate

It is not an error to receive a `duplicate` response code, if this is a retry of a previously failed invocation. The previous invocation might have succeeded on the *ELS service instance*, but the *management program* did not know it had succeeded.

4.2.2 removeInteraction

4.2.2.1 Description

The `removeInteraction` operation is used by *management programs* to remove an *interaction record* from the *current set*.

The input contains:

- The *interaction record* to remove.

The output contains:

- An enumerated type `ELSPublishReturnCode`.

4.2.2.2 Conformance points for ELS service instances

ELS 24 An *ELS service instance* **SHALL** implement the `removeInteraction` operation complying with the *Web Services Base Profile* of [ATS5820—2010].

ELS 25 An *ELS service instance* **SHALL** implement the `removeInteraction` operation complying with the *TLS Security Profile* of [ATS5820—2010].

ELS 26 If the `target` is not registered, the *ELS service instance* **SHALL** send back a `publishError` SOAP fault with `unknownTargetId`.

ELS 27 If the `target` is registered and an equal *interaction record* is in the *current set*, the *ELS service instance* **SHALL** remove it from the *current set* and respond with a `returnCode` of `ok`.

ELS 28 If the `target` is registered and the *interaction record* is not in the *current set*, the *ELS service instance* **SHALL** respond with a `returnCode` of `notFound`.

4.2.2.3 Conformance points for management programs

ELS 29 *Management programs* **SHALL** invoke the `removeInteraction` operation complying with the *Web Services Base Profile* of [ATS5820—2010].

ELS 30 *Management programs* **SHALL** invoke the `removeInteraction` operation complying with the *TLS Security Profile* of [ATS5820—2010].

4.2.2.4 Notes (non-normative)

4.2.2.4.1 Equality

The equality of *interaction records* is defined in section 2.3.2.1.

4.2.2.4.2 NotFound

It is not an error to receive a `notFound` response code, if this is a retry of a previously failed invocation. The previous invocation might have succeeded on the *ELS service instance*, but the *management program* did not know it had succeeded.

4.3 Security

4.3.1 Authorisation

The *ELS service instance* is responsible for determining and implementing its own authorisation policies.

It is expected that the *target service owner*, or delegates operating on their behalf, are permitted to add or remove *interaction records* from the *current set*. For example, the *target service provider* could outsource the maintenance of *interaction records* in the *ELS service instance* to their *target service operator*.

The mechanism for determining delegation is implementation specific and is outside the scope for this specification to define.

Appendix A: Change log

Version 1.3

- First release.

  The first release of this technical service specification was aligned with the *Endpoint Location Service: Solution Design v1.3* document, so it has the same version number.