

nehta

Connectivity

Implementation Guide

Version 1.1 — 30 June 2010

National E-Health Transition Authority Ltd

Level 25

56 Pitt Street

Sydney, NSW, 2000

Australia.

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 © 2010, 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

Table of contents	iii
Document information	iv
Change history	iv
1 Introduction	1
1.1 Document purpose	1
1.2 Intended audience	1
1.3 Definitions, acronyms, abbreviations	1
1.4 Normative references	1
1.5 Overview	1
2 Implementing without the National Infrastructure	3
2.1 Healthcare Identifiers	3
2.1.1 Alternatives to IHI	3
2.1.2 Alternatives to HPI	4
2.2 Authentication	4
2.2.1 Alternatives to NASH	4
2.3 Endpoint Location Service	5
2.3.1 Pre-ELS deployments	5
2.3.2 Alternatives to the Base-ELS	5
3 Performance and reliability	6
3.1 Performance	6
3.1.1 Caching	6
3.2 Reliability	6
3.2.1 Retries	6
3.2.2 Escalation	7
3.3 Auditing and tracking	7
3.3.1 Tracking	7
3.3.2 Auditing	7
4 Security	9
4.1 Service invokers	9
4.1.1 Firewalls	9
4.2 Service providers	9
4.2.1 Firewalls and NAT	9
4.2.2 DMZ	9
5 Deployment	11
5.1 Endpoint Location Service (ELS)	11
5.1.1 Automatic ELS updating	11
5.2 Integration	11
5.2.1 Integration with proprietary systems	11
Appendix A: Change log	13

Document information

Change history

Version	Date	Comments
1.0	2008-12-01	Release
1.1	2010-06-30	Release

1 Introduction

1.1 Document purpose

This document provides guidance on how to design and implement systems which follow the connectivity architecture and its related specifications.

This is a non-normative document. It clarifies certain aspects of the connectivity specification and offers suggested approaches for handling common issues. Implementations can, but do not have to, use 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 NEHTA *Connectivity: Architecture* document [CA2010].

1.3 Definitions, acronyms, abbreviations

ELS	Endpoint Location Service
HPI	Healthcare Provider Identifier
IHI	Individual Healthcare Identifier
NASH	National Authentication Service for Health
NEHTA	National E-Health Transition Authority
TLS	Transport Layer Security

1.4 Normative references

The following NEHTA specifications and other references contain provisions which, through reference in this text, constitute provisions of this Specification. At the time of publication, the editions indicated were valid. All Specification and other references are subject to revision: all users of this Specification are therefore encouraged to investigate the possibility of applying the most recent edition of the Specification and other references listed below.

[CA2010]	NEHTA, <i>Connectivity: Architecture v1.1</i> , 30 June 2010.
[NIF2007]	NEHTA, <i>Interoperability Framework v2.0</i> , 2008.
[QI2010]	NEHTA, <i>Qualified Identifiers v2.0</i> , 5 March 2010.

1.5 Overview

This document provides guidelines on a number of topics. Although they are all related to connectivity, these topics are not necessarily related to each other. For ease of presentation, they are grouped into four chapters:

- Chapter 2 on implementing without the national infrastructure services topics.
- Chapter 3 on performance and reliability topics.
- Chapter 4 on security topics.
- Chapter 5 on deployment topics.

For each topic, the issue, recommendation and tradeoffs are described.

2 Implementing without the National Infrastructure

The national infrastructure services are a set of common services whose implementation is being managed by NEHTA. These services are being deployed according to their own timelines, so there will be an interim period when they are not all available for e-health interactions.

This chapter describes alternatives for the national infrastructure services. These can be used as temporary substitutes until the national service becomes available.

2.1 Healthcare Identifiers

2.1.1 Alternatives to IHI

2.1.1.1 Issue

Implementing connectivity before the NEHTA Individual Healthcare Identifier (IHI) numbers and services are available.

2.1.1.2 Recommendation

Use local individual identifiers which are agreed upon between the communicating parties, or another identifier that is provided by the individual.

The services in the architecture are designed to use qualified identifiers to identify individuals [QI2010]. Systems should be designed to process qualified identifier as opaque URIs—only comparing their values for equality and not being concerned with the format inside them or who issued them.

The architecture depends on the same identifier being used, not where the identifier comes from. Therefore, in the interim, either:

- A qualified identifier can be issued to the individual by one healthcare organisation, and that URI is used by other organisations to identify that individual. An organisation should reuse an existing identifier if one is available.

For example, a general practice can create a qualified identifier from their internal patient identifiers. When a pharmacy receives a prescription from that GP, they should use that qualified identifier to identify the individual.

- The individual chooses their own URI as their qualified identifier, and informs all the organisations they deal with to use that URI to identify them. This is similar to the OpenID concept, where the function identity management is decentralised.

2.1.1.3 Tradeoffs

This interim approach has the risk of multiple qualified identifiers being used to identify the same individual. Care should be taken to avoid this situation—organisations should use an existing qualified identifier (if it exists and they know what it is), in preference to creating their own—and processes should be available to handle it if it arises—allowing multiple qualified identifiers to be mapped to the same individual and making it easy to change and merge records.

2.1.2 Alternatives to HPI

2.1.2.1 Issue

Implementing connectivity before the NEHTA Healthcare Provider Identifier (HPI) numbers and services are available.

2.1.2.2 Recommendation

Use local healthcare provider identifiers which are agreed upon between the communicating parties, or another identifier that is provided by the organisation.

The services in the architecture are designed to use qualified identifiers to identify healthcare providers. See section 2.1.1 for a discussion about how non-centrally assigned qualified identifiers can be used.

For example, a pathology laboratory could create a qualified identifier from their Australian Business Number (ABN) as long as the receiving General Practice can match those numbers to the laboratory.

2.1.2.3 Tradeoffs

See section 2.1.1.3.

2.2 Authentication

2.2.1 Alternatives to NASH

2.2.1.1 Issues

Implementing connectivity before the NEHTA National Authentication for Health (NASH) PKI Authentication for Health (NASH) PKI and services are available.

2.2.1.2 Recommendation

Use certificates from an alternative source with agreement between the communicating parties.

Possible sources of certificates include: Medicare Australia location certificates, commercially available certificate authorities, certificates created by a vendor or community for its members, self-signed certificates.

2.2.1.3 Tradeoffs

This interim solution requires each implementation and/or deployment to perform these tasks through their own efforts instead of relying on NASH to help them:

- Obtain public key certificates given a reference to them.

Instead of retrieving certificates from a single location, certificates will have to be manually obtained from the certificate holder before electronic communications takes place.

- Map certificates to the identity of the certificate's owner.

Instead of examining a known entry inside the certificate, systems will have to deal with many different types of certificates. The simplest approach will be to treat the certificate as opaque, and not try to extract an identifier from it. Instead, the system could itself keep track of which certificate belongs to which owner as a simple mapping. This mapping table will have to be manually populated.

- Validate certificates to see if they have been revoked or not.
Instead of checking a single Certificate Revocation List (CRL) or using a single OCSP service, the system will have to deal with many different mechanisms for revocation checking.

The interim solution requires manual configuration of certificates, but organisations might want to do this even if there were a single certificate provider. If an organisation regularly communicates with only a few parties, security can be improved by manually white-listing certificates from those organisations, instead of simply trusting anyone with a valid certificate.

The ability to manually add certificates and to white-list them can be useful to have, even if there were a centralised certificate authority.

2.3 Endpoint Location Service

2.3.1 Pre-ELS deployments

2.3.1.1 Issues

Implementing the core connectivity interaction without using an Endpoint Location Service (ELS).

2.3.1.2 Recommendation

This situation should not occur, because a deployment of a NEHTA service should also include a deployment of the ELS.

In the absence of a ELS, the values of the ELS must be manually obtained and configured into the service invoker. For example, they could be interchanged as data files which are imported into the service invoker.

2.3.1.3 Tradeoffs

The ELS addresses the problems of locating and using service instances in a distributed and dynamic environment. If ELS is not used, these problems will have to be addressed manually or using proprietary directories.

Although manual methods can work in a small environment, it does not scale to a large environment. It might work during the early stages of deployment where there are a small number of service instances, but it will become impractical as more and more service instances are deployed.

2.3.2 Alternatives to the Base-ELS

2.3.2.1 Issues

The Base-ELS is an instance of an Endpoint Location Service that contains entries for the national infrastructure services.

Service invokers use the Base-ELS to obtain the address and identify the certificates for services such as the IHI service, HPI-I service, HPI-O service and NASH service.

2.3.2.2 Recommendation

In the absence of the Base-ELS, applications should be locally configured with the information they require. Instead of looking up the Base-ELS, they would obtain the information from a local configuration file or local database.

2.3.2.3 Tradeoffs

The local configuration file or local database will have to be manually updated when the information changes.

3 Performance and reliability

3.1 Performance

3.1.1 Caching

3.1.1.1 Issue

Invoking multiple services takes time and places a load on those services.

3.1.1.2 Recommendation

Cache data whenever it is possible.

Caching data means to save data that is obtained from a service for future use. If it is required again, the saved data is used instead of obtaining it from the service.

Technical service specifications should indicate what data can be cached and provide mechanisms to support caching whenever possible.

3.1.1.3 Tradeoffs

Caching consumes local storage space and appropriate policies must be defined to remove stale data from the cache.

Only data that does not change frequently can be successfully cached. It is expected that some data from the IHI, HPI, NASH and ELS can be cached.

When data does change, mechanisms must be in place to ensure that the cached data is not used or does not cause problems if it is used.

3.2 Reliability

3.2.1 Retries

3.2.1.1 Issue

Sometimes a service instance is temporally unavailable.

For example, the service is undergoing maintenance or there is temporary network failure.

3.2.1.2 Recommendation

Service invokers can retry invoking the service instance at a later time. It can repeat this process if the retry also fails.

3.2.1.3 Tradeoffs

This is only suitable for batched or asynchronous processes, where an immediate result is not needed.

The service instance may be permanently unavailable, in which case retries will never succeed. Mechanisms must be in place to handle this situation.

If cached data is being used, mechanisms must be defined to handle the situation where stale cached data is being used.

3.2.2 Escalation

3.2.2.1 Issue

Sometimes the system cannot automatically recover from an error.

3.2.2.2 Recommendation

Systems should be designed to escalate the problem for handling by another process if the system cannot resolve the problem. The escalation process would eventually result in a person handling the problem if it cannot be resolved by the system.

3.2.2.3 Tradeoffs

Manual processes are costly to perform. It is preferable to automatically handle the error, but there will always be situations where a manual process cannot be avoided.

3.3 Auditing and tracking

3.3.1 Tracking

3.3.1.1 Issue

The operation of a system may need to be diagnosed when a problem occurs.

3.3.1.2 Recommendation

Messages that are sent or received should be tracked. Minimally, this will be a log entry that records a timestamp for the message, the message ID, and the source or destination. Additional information, such as the message itself, can also be logged.

It is the responsibility of each system to maintain its own tracking logs.

3.3.1.3 Tradeoffs

Logs incur an overhead to create and store. These overheads increase as more information is stored in a log. Logs may be stored offline or purged to save space.

3.3.2 Auditing

3.3.2.1 Issue

Historical information may need to be kept for auditing purposes.

In this section, a distinction is made between tracking and auditing. Tracking is an optional feature used for technical diagnosis. Auditing is a mandatory feature used to support a business requirement (usually a legal requirement). It is possible to permit an operation to occur if its tracking requirements could not be met, but not if its auditing requirements could not be met.

3.3.2.2 Recommendation

Auditing requirements should be defined for a service. It can either be defined by the service interface specification, by an implementation profile of that specification, or by a mutual service level agreement.

The auditing requirements should always be based upon a real business requirement for it.

Currently, it is the responsibility of the individual systems to maintain their own audit logs. If there is sufficient requirements and demand, services to support auditing may be defined in the future.

3.3.2.3 Tradeoffs

Audit logs incur an overhead to create and to store. These overheads increase as more information is stored in an audit log. Audit logs maybe store offline, but cannot be purged unless permitted by the business rules.

4 Security

4.1 Service invokers

4.1.1 Firewalls

4.1.1.1 Issue

Programs acting as Web services invokers will usually be operating from behind local firewalls.

Most consumer grade firewalls usually block incoming connections, but does not block outgoing connections. These firewalls will not cause any problems for the organisation to act as a service invoker.

Business and enterprise firewalls are capable of blocking outgoing connections too. These firewalls can block outgoing web service invocations.

4.1.1.2 Recommendation

If the firewall is set to block outgoing connections, it needs to be configured to allow the Web services software to initiate connections to the Internet.

4.2 Service providers

4.2.1 Firewalls and NAT

4.2.1.1 Issue

Programs acting as Web services providers should be operating from behind local firewalls. These firewalls may prevent incoming connections, which will prevent connectivity from happening.

The organisation may also be running some form of Network Address Translation (NAT), which prevents external service invokers from directly connecting to the machine that is actually running the Web service.

4.2.1.2 Recommendation

The firewall needs to be configured to allow the Web services software to receive connections from the Internet.

Routers may need to be configured so that the running Web services are contactable by external parties. In the case of NAT, port forwarding will have to be configured.

4.2.2 DMZ

4.2.2.1 Issue

Many organisations deploy servers in a Demilitarized Zone (DMZ). Confidential data may be compromised if it is left exposed in the DMZ.

4.2.2.2 Recommendation

Confidential data should not be stored in the clear in the DMZ. This also applies to decryption keys which can be used to decrypt confidential data.

It is recommended that the decryption keys for confidential data be stored in the local network, and not in the DMZ. The decryption of confidential data will take inside the local network, and not in the DMZ. Therefore, confidential data is not exposed, even if the DMZ was compromised.

This level of protection can be achieved using one of the following techniques:

- If the service uses an XML secured payload that is encrypted, the encrypted payload can be stored in the DMZ. Computers from the local network can initiate a connection to the DMZ to retrieve the secured payload.
- The WS-Security secured SOAP message can be stored in the DMZ. Computers from the inner network can initiate a connection to the DMZ to retrieve it and provide any response data. This is more difficult to implement, since most implementations of SOAP Web services do not support handling of SOAP messages in this manner.

4.2.2.3 Tradeoffs

Web services which do not require a real-time response can be easily deployed in the DMZ. The computer in the DMZ can act as an exchange, and periodically a service inside the local network can connect to the DMZ to retrieve the messages.

If the Web services require real-time response and data that it requires is inside the local network, then the DMZ must be allowed to connect to the local network. For the organisation to be able to offer that service, their security policies must be designed to permit it to function.

5 Deployment

5.1 Endpoint Location Service (ELS)

5.1.1 Automatic ELS updating

5.1.1.1 Issue

The Endpoint Location Service must contain correct entries for deployed service instances.

It is necessary to ensure that an ELS is updated whenever service instances are deployed, modified, or decommissioned.

5.1.1.2 Recommendation

Service implementations should be designed to automatically update the entry in the ELS.

The ELS defines a standard service interface for maintaining ELS entries. A service instance can automatically invoke that interface whenever it is deployed. It can also check if its ELS entry needs updating when the service instance is started.

5.1.1.3 Tradeoffs

The service instance might require additional configurations to perform automatic updating of ELS entries.

5.2 Integration

5.2.1 Integration with proprietary systems

5.2.1.1 Issue

Not all systems will be compliant with the standards. For example, there are existing systems that do not support Web services. Over time, these may be modified to support the standards, but in some situations this may not be practical.

5.2.1.2 Recommendation

Use a gateway that translates between the standard interface and proprietary (non-compliant) system. The gateway connects to external organisations in the national e-health environment using the standard protocols. The gateway connects to the non-compliant system using some proprietary protocol or mechanism.

Using this approach, a non-compliant system can communicate with a compliant system. Two non-compliant systems can communicate with each other through their own gateways. This approach is illustrated in Figure 1.

The fact that a system is using a gateway should be invisible to an external organisation that communicates with it.

The gateway needs to perform the necessary transformations and mapping between the two protocols.

The gateway may be implemented by the owner of the non-complaint system, its vendor, or by a third party.

5.2.1.3 Tradeoffs

The costs of implementing a gateway compared to modifying the non-compliant system needs to be considered.

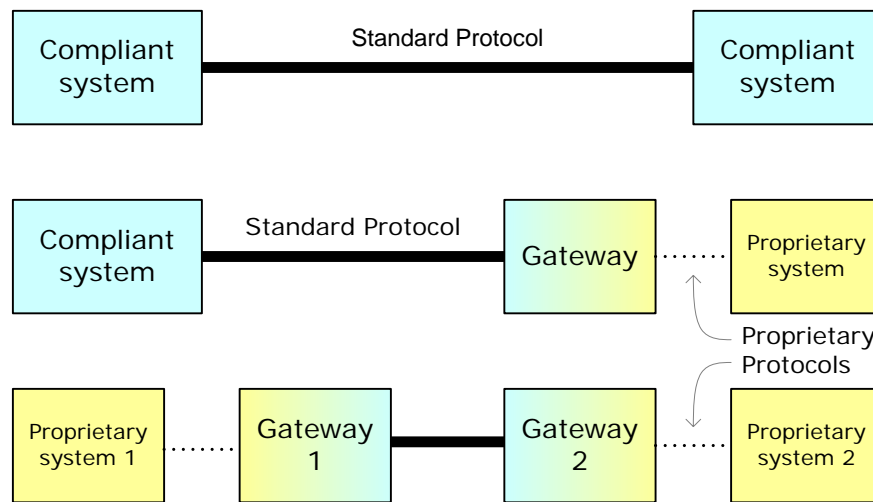


Figure 1: Integration with proprietary systems

Appendix A: Change log

Version 1.1

- Service Instance Locator (SIL) renamed to Endpoint Location Service (ELS).
- Removed discussion about TLS security, since that issue is covered by the technical service specifications.